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	4MHz
Connectivity	SCI, SPI
Peripherals	POR, WDT
Number of I/O	30
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	4.75V ~ 5.25V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-QFP
Supplier Device Package	80-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc11f1cpu4

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Paragraph Title Page

SECTION 9 TIMING SYSTEM

SECTION 10 ANALOG-TO-DIGITAL CONVERTER

2.2 Reset ($\overline{\text{RESET}}$)

An active low bidirectional control signal, $\overline{\text{RESET}}$, acts as an input to initialize the MCU to a known start-up state. It also acts as an open-drain output to indicate that an internal failure has been detected in either the clock monitor or COP watchdog circuit. The CPU distinguishes between internal and external reset conditions by sensing whether the reset pin rises to a logic one in less than two E-clock cycles after a reset has occurred. It is not advisable to connect an external resistor-capacitor (RC) power-up delay circuit to the reset pin of M68HC11 devices because the circuit charge time constant can cause the device to misinterpret the type of reset that occurred. Refer to **SECTION 5 RESETS AND INTERRUPTS** for further information.

Figure 2-3 illustrates a reset circuit that uses an external switch. Other circuits can be used, however, it is important to incorporate a low voltage interrupt (LVI) circuit to prevent operation at insufficient voltage levels which could result in erratic behavior or corruption of RAM.

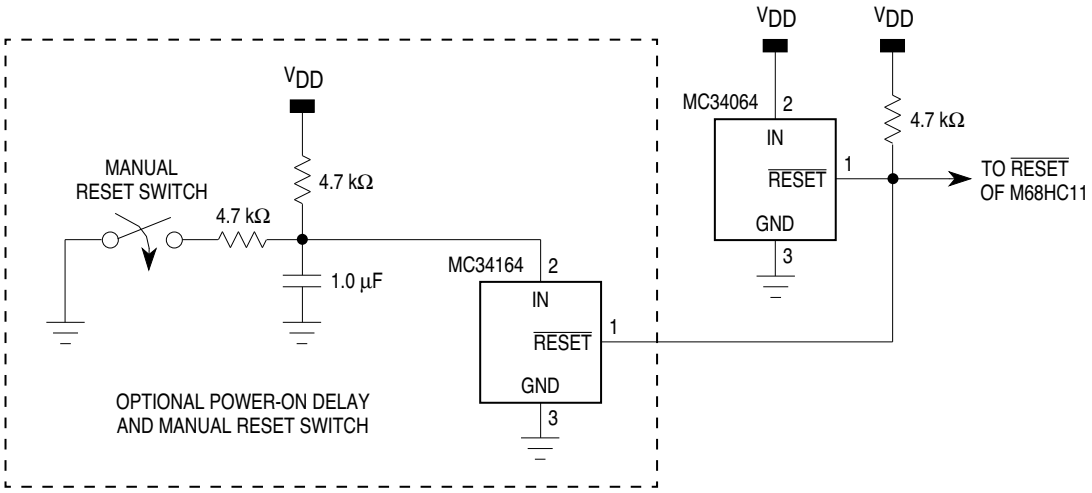


Figure 2-3 External Reset Circuit

2.3 E-Clock Output (E)

E is the output connection for the internally generated E clock. The signal from E is used as a timing reference. The frequency of the E-clock output is one fourth that of the input frequency at the EXTAL pin. When E-clock output is low, an internal process is taking place. When it is high, data is being accessed. All clocks, including the E clock, are halted when the MCU is in STOP mode. The E clock can be turned off in single-chip modes to reduce the effects of radio frequency interference (RFI). Refer to **SECTION 9 TIMING SYSTEM**.

2.4 Crystal Driver and External Clock Input (XTAL, EXTAL)

These two pins provide the interface for either a crystal or a CMOS-compatible clock to control the internal clock generator circuitry. Either a crystal oscillator or a CMOS compatible clock can be used. The resulting E-clock rate is the input frequency divided by four.

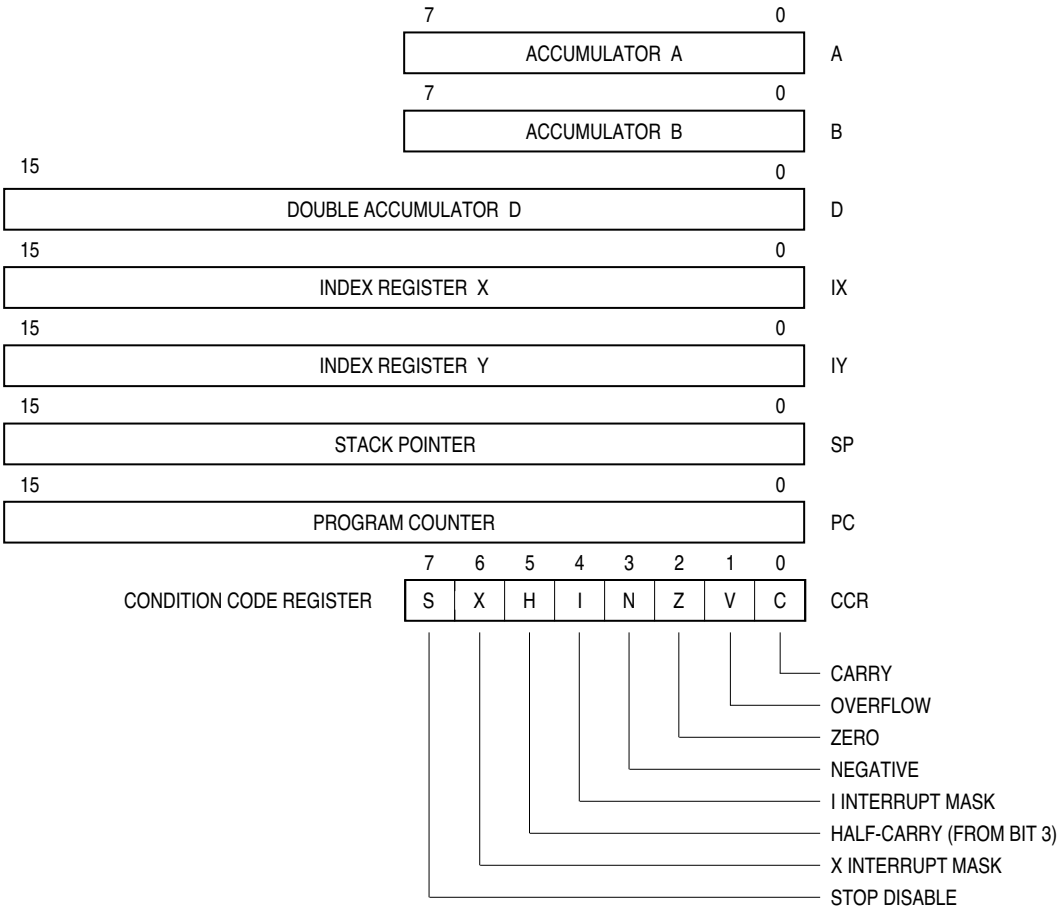


Figure 3-1 Programming Model

3.1.1 Accumulators A, B, and D

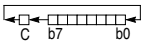
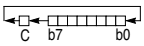
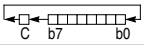
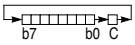
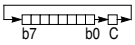
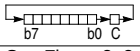
Accumulators A and B are general-purpose 8-bit registers that hold operands and results of arithmetic calculations or data manipulations. For some instructions, these two accumulators are treated as a single double-byte (16-bit) accumulator called accumulator D. Although most instructions can use accumulators A or B interchangeably, the following exceptions apply:

The ABX and ABY instructions add the contents of 8-bit accumulator B to the contents of 16-bit register X or Y, but there are no equivalent instructions that use A instead of B.

The TAP and TPA instructions transfer data from accumulator A to the condition code register, or from the condition code register to accumulator A, however, there are no equivalent instructions that use B rather than A.

The decimal adjust accumulator A (DAA) instruction is used after binary-coded decimal (BCD) arithmetic operations, but there is no equivalent BCD instruction to adjust accumulator B.

Table 3-2 Instruction Set (Sheet 5 of 6)

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Operand	Cycles	Condition Codes S	X	H	I	N	Z	V	C
NOP	No operation	No Operation	INH	01	—	2	—	—	—	—	—	—	—	—
ORAA (opr)	OR Accumulator A (Inclusive)	A + M ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	8A 9A BA AA 18 AA	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
ORAB (opr)	OR Accumulator B (Inclusive)	B + M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	CA DA FA EA 18 EA	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
PSHA	Push A onto Stack	A ⇒ Stk,SP = SP – 1	A INH	36	—	3	—	—	—	—	—	—	—	—
PSHB	Push B onto Stack	B ⇒ Stk,SP = SP – 1	B INH	37	—	3	—	—	—	—	—	—	—	—
PSHX	Push X onto Stack (Lo First)	IX ⇒ Stk,SP = SP – 2	INH	3C	—	4	—	—	—	—	—	—	—	—
PSHY	Push Y onto Stack (Lo First)	IY ⇒ Stk,SP = SP – 2	INH	18 3C	—	5	—	—	—	—	—	—	—	—
PULA	Pull A from Stack	SP = SP + 1, A ⇐ Stk	A INH	32	—	4	—	—	—	—	—	—	—	—
PULB	Pull B from Stack	SP = SP + 1, B ⇐ Stk	B INH	33	—	4	—	—	—	—	—	—	—	—
PULX	Pull X From Stack (Hi First)	SP = SP + 2, IX ⇐ Stk	INH	38	—	5	—	—	—	—	—	—	—	—
PULY	Pull Y from Stack (Hi First)	SP = SP + 2, IY ⇐ Stk	INH	18 38	—	6	—	—	—	—	—	—	—	—
ROL (opr)	Rotate Left		EXT IND,X IND,Y	79 69 18 69	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
ROLA	Rotate Left A		A INH	49	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLB	Rotate Left B		B INH	59	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROR (opr)	Rotate Right		EXT IND,X IND,Y	76 66 18 66	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
RORA	Rotate Right A		A INH	46	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORB	Rotate Right B		B INH	56	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RTI	Return from Interrupt	See Figure 3–2	INH	3B	—	12	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ
RTS	Return from Subroutine	See Figure 3–2	INH	39	—	5	—	—	—	—	—	—	—	—
SBA	Subtract B from A	A – B ⇒ A	INH	10	—	2	—	—	—	—	Δ	Δ	Δ	Δ
SBCA (opr)	Subtract with Carry from A	A – M – C ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	82 92 B2 A2 18 A2	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	Δ	Δ
SBCB (opr)	Subtract with Carry from B	B – M – C ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	C2 D2 F2 E2 18 E2	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	Δ	Δ
SEC	Set Carry	1 ⇒ C	INH	0D	—	2	—	—	—	—	—	—	—	1
SEI	Set Interrupt Mask	1 ⇒ I	INH	0F	—	2	—	—	—	1	—	—	—	—
SEV	Set Overflow Flag	1 ⇒ V	INH	0B	—	2	—	—	—	—	—	—	1	—
STAA (opr)	Store Accumulator A	A ⇒ M	A DIR A EXT A IND,X A IND,Y	97 B7 A7 18 A7	dd hh ll ff ff	3 4 4 5	—	—	—	—	Δ	Δ	0	—

4.3.2 Initialization

Because bits in the following registers control the basic configuration of the MCU, an accidental change of their values could cause serious system problems. The protection mechanism, overridden in special operating modes, requires a write to the protected bits only within the first 64 bus cycles after any reset, or only once after each reset. **Table 4-2** summarizes the write access limited registers.

4.3.2.1 CONFIG Register

CONFIG controls the presence and position of the EEPROM in the memory map. CONFIG also enables the COP watchdog timer.

CONFIG — System Configuration Register **$103F**

	Bit 7	6	5	4	3	2	1	Bit 0	
	EE3	EE2	EE1	EE0	—	NOCOP	—	EEON	
RESET:	1	1	1	1	1	P	1	1	Single Chip
	1	1	1	1	1	P(L)	1	1	Bootstrap
	P	P	P	P	1	P	1	P	Expanded
	P	P	P	P	1	P(L)	1	0	Special Test

P indicates a previously programmed bit. P(L) indicates that the bit resets to the logic level held in the latch prior to reset, but the function of COP is controlled by DISR bit in TEST1 register.

The CONFIG register consists of an EEPROM byte and static latches that control the start-up configuration of the MCU. The contents of the EEPROM byte are transferred into static working latches during reset sequences. The operation of the MCU is controlled directly by these latches and not by CONFIG itself. In normal modes, changes to CONFIG do not affect operation of the MCU until after the next reset sequence. When programming, the CONFIG register itself is accessed. When the CONFIG register is read, the static latches are accessed.

These bits can be read at any time. The value read is the one latched into the register from the EEPROM cells during the last reset sequence. A new value programmed into this register cannot be read until after a subsequent reset sequence. Unused bits always read as ones.

In special test mode, the static latches can be written directly at any time. In all modes, CONFIG bits can only be programmed using the EEPROM programming sequence, and are neither readable nor active until latched via the next reset. Refer to **4.4.3 CONFIG Register Programming**.

EE[3:0] — EEPROM Mapping Control

EE[3:0] select the upper four bits of the EEPROM base address. In single-chip and bootstrap modes, EEPROM is forced to $FE00–$FFFF regardless of the value of EE[3:0].

```
ROWE    LDAB    #$0E        ROW=1, ERASE=1, EELAT=1, EEPGM=0
        STAB    $103B       Set to ROW erase mode
        STAB    0,X         Store any data to any address in ROW
        LDAB    #$0F        ROW=1, ERASE=1, EELAT=1, EEPGM=1
        STAB    $103B       Turn on high voltage
        JSR     DLY10       Delay 10 ms
        CLR     $103B       Turn off high voltage and set to READ mode
```

4.4.1.4 EEPROM Byte Erase

The following is an example of how to erase a single byte of EEPROM and assumes that index register X contains the address of the byte to be erased.

```
BYTEE   LDAB    #$16        BYTE=1, ROW=0, ERASE=1, EELAT=1, EEPGM=0
        STAB    $103B       Set to BYTE erase mode
        STAB    0,X         Store any data to address to be erased
        LDAB    #$17        BYTE=1, ROW=0, ERASE=1, EELAT=1, EEPGM=1
        STAB    $103B       Turn on high voltage
        JSR     DLY10       Delay 10 ms
        CLR     $103B       Turn off high voltage and set to READ mode
```

4.4.2 PPROG EEPROM Programming Control Register

Bits in PPROG register control parameters associated with EEPROM programming.

PPROG — EEPROM Programming Control **$103B**

	Bit 7	6	5	4	3	2	1	Bit 0
	ODD	EVEN	—	BYTE	ROW	ERASE	EELAT	EEPGM
RESET:	0	0	0	0	0	0	0	0

ODD — Program Odd Rows in Half of EEPROM (TEST)

EVEN — Program Even Rows in Half of EEPROM (TEST)

Bit 5 — Not implemented
Always reads zero

BYTE — Byte/Other EEPROM Erase Mode
0 = Row or bulk erase mode used
1 = Erase only one byte of EEPROM

ROW — Row/All EEPROM Erase Mode (only valid when BYTE = 0)
0 = All 512 bytes of EEPROM erased
1 = Erase only one 16-byte row of EEPROM

CR[1:0] — COP Timer Rate Select

The internal E clock is first divided by 2^{15} before it enters the COP watchdog system. These control bits determine a scaling factor for the watchdog timer. Refer to **Table 5-1**.

5.1.6 CONFIG Register

CONFIG — System Configuration Register **$103F**

	Bit 7	6	5	4	3	2	1	Bit 0	
	EE3	EE2	EE1	EE0	—	NOCOP	—	EEON	
RESET:	1	1	1	1	1	P	1	1	Single Chip
	1	1	1	1	1	P(L)	1	1	Bootstrap
	P	P	P	P	1	P	1	P	Expanded
	P	P	P	P	1	P(L)	1	0	Special Test

P indicates a previously programmed bit. P(L) indicates that the bit resets to the logic level held in the latch prior to reset, but the function of COP is controlled by DISR in TEST1 register.

EE[3:0] — EEPROM Mapping Control

Refer to **SECTION 4 OPERATING MODES AND ON-CHIP MEMORY**.

Bit 3 — Not implemented

Always reads one

NOCOP — COP System Disable

0 = COP system enabled (forces reset on time-out)

1 = COP system disabled

Bit 1 — Not implemented

Always reads one

EEON — EEPROM Enable

Refer to **SECTION 4 OPERATING MODES AND ON-CHIP MEMORY**.

5.2 Effects of Reset

When a reset condition is recognized, the internal registers and control bits are forced to an initial state. Depending on the cause of the reset and the operating mode, the reset vector can be fetched from any of six possible locations. Refer to **Table 5-2**.

Table 5-2 Reset Cause, Operating Mode, and Reset Vector

Cause of Reset	Normal Mode Vector	Special Test or Bootstrap
POR or RESET Pin	$FFFE, FFFF	$BFFE, $BFFF
Clock Monitor Failure	$FFFC, FFFD	$BFFC, $BFFD
COP Watchdog Time-out	$FFFA, FFFB	$BFFA, $BFFB

These initial states then control on-chip peripheral systems to force them to known start-up states, as follows:

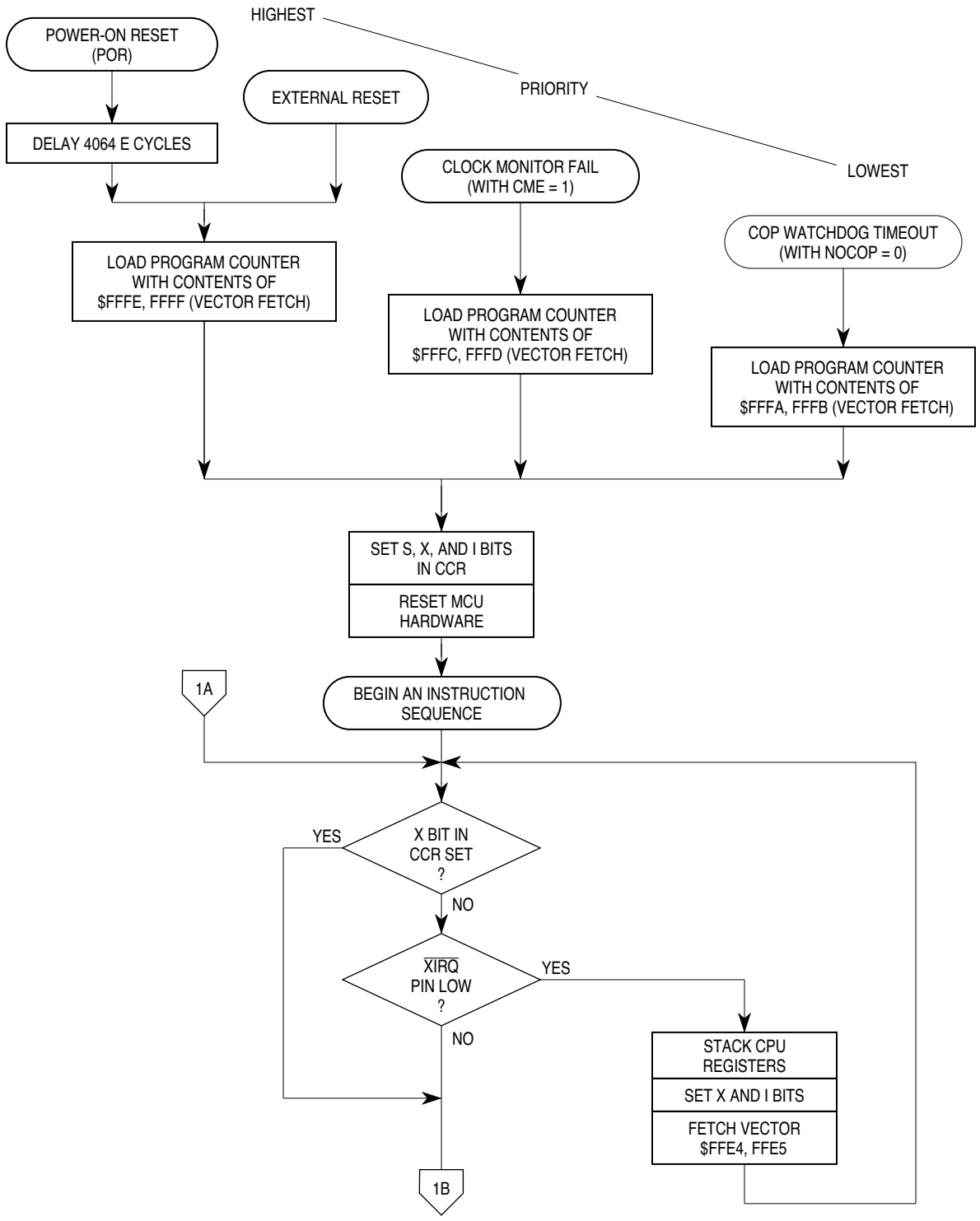


Figure 5-1 Processing Flow Out of Reset (1 of 2)

SECTION 7 SERIAL COMMUNICATIONS INTERFACE

The serial communications interface (SCI) is a universal asynchronous receiver transmitter (UART), one of two independent serial I/O subsystems in the MC68HC11F1 MCU. It has a standard nonreturn to zero (NRZ) format (one start bit, eight or nine data bits, and one stop bit). Several baud rates are available. The SCI transmitter and receiver are independent, but use the same data format and bit rate.

7.1 Data Format

The serial data format requires the following conditions:

1. An idle-line in the high state before transmission or reception of a message.
2. A start bit, logic zero, transmitted or received, that indicates the start of each character.
3. Data that is transmitted and received least significant bit (LSB) first.
4. A stop bit, logic one, used to indicate the end of a frame. (A frame consists of a start bit, a character of eight or nine data bits, and a stop bit.)
5. A break (defined as the transmission or reception of a logic zero for some multiple number of frames).

Selection of the word length is controlled by the M bit of SCI control register SCCR1.

7.2 Transmit Operation

The SCI transmitter includes a parallel transmit data register (SCDR) and a serial shift register. The contents of the serial shift register can only be written through the SCDR. This double buffered operation allows a character to be shifted out serially while another character is waiting in the SCDR to be transferred into the serial shift register. The output of the serial shift register is applied to TxD as long as transmission is in progress or the transmit enable (TE) bit of serial communication control register 2 (SCCR2) is set. The block diagram, **Figure 7-1**, shows the transmit serial shift register and the buffer logic at the top of the figure.

OR — Overrun Error Flag

OR is set if a new character is received before a previously received character is read from SCDR. Clear the OR flag by reading SCSR and then reading SCDR.

0 = No overrun
1 = Overrun detected

NF — Noise Error Flag

NF is set if majority sample logic detects anything other than a unanimous decision. Clear NF by reading SCSR and then reading SCDR.

0 = Unanimous decision
1 = Noise detected

FE — Framing Error

FE is set when a zero is detected where a stop bit was expected. Clear the FE flag by reading SCSR and then reading SCDR.

0 = Stop bit detected
1 = Zero detected

Bit 0 — Not implemented

Always reads zero

7.6.5 Baud Rate Register

Use this register to select different baud rates for the SCI system. The SCP[1:0] bits select the prescaler rate for the SCR[2:0] bits. Together, these five bits provide multiple baud rate combinations for a given crystal frequency. Normally, this register is written once during initialization. The prescaler is set to its fastest rate by default out of reset, and can be changed at any time. Refer to **Table 7-1** and **Table 7-2** for normal baud rate selections.

BAUD — Baud Rate **$102B**

	Bit 7	6	5	4	3	2	1	Bit 0
	TCLR	—	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0
RESET:	0	0	0	0	0	U	U	U

TCLR — Clear Baud Rate Counters (Test)

SCP[1:0] — SCI Baud Rate Prescaler Selects

Refer to the SCI baud rate generator block diagram.

Table 7-1 Baud Rate Prescaler Selection

Prescaler		Divide Internal	Crystal Frequency (MHz)					
SCP1	SCP0	Clock By	4.0	4.9152	8.0	12.0	16.0	20.0
0	0	1	62500	76800	125000	187500	25000	312500
0	1	3	20833	25600	41667	62500	83332	104165
1	0	4	15625	19200	31250	46875	62500	78125
1	1	13	4800	5907	9600	14423	19200	24000

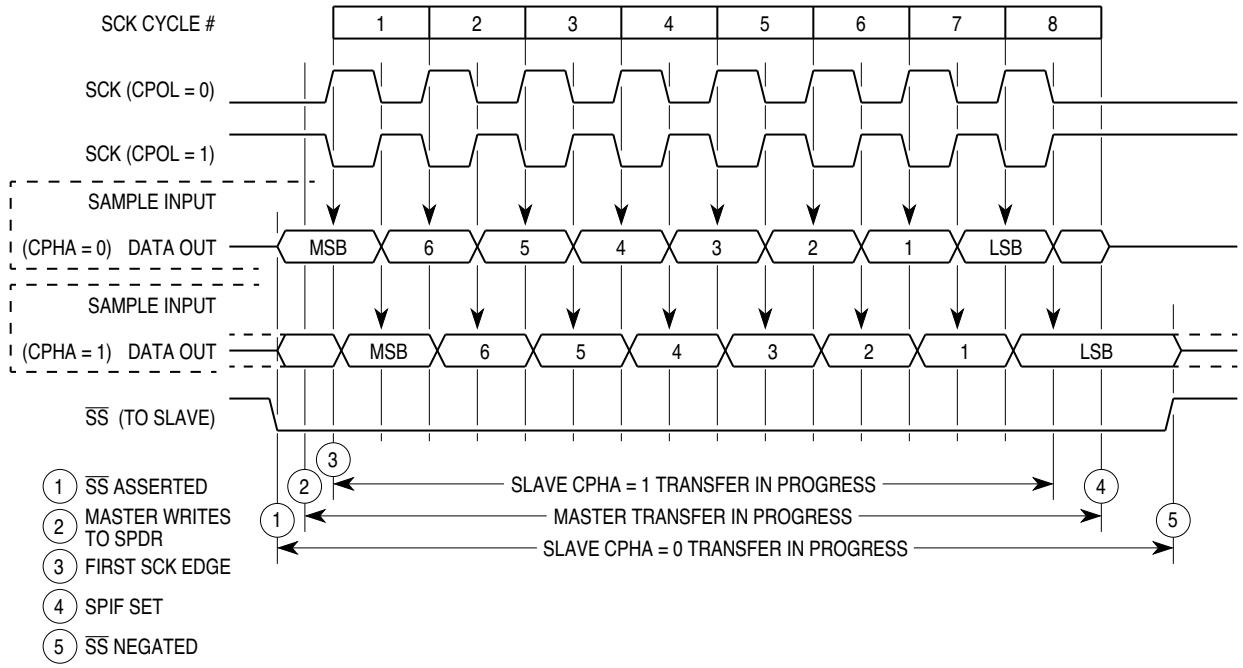


Figure 8-2 SPI Transfer Format

8.2.1 Clock Phase and Polarity Controls

Software can select one of four combinations of serial clock phase and polarity using two bits in the SPI control register (SPCR). The clock polarity is specified by the CPOL control bit, which selects an active high or active low clock, and has no significant effect on the transfer format. The clock phase (CPHA) control bit selects one of two different transfer formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transfers to allow a master device to communicate with peripheral slaves having different requirements.

When CPHA equals zero, the $\overline{SS}$ line must be negated and reasserted between each successive serial byte. Also, if the slave writes data to the SPI data register (SPDR) while $\overline{SS}$ is low, a write collision error results.

When CPHA equals one, the $\overline{SS}$ line can remain low between successive transfers.

8.3 SPI Signals

The following paragraphs contain descriptions of the four SPI signals: master in slave out (MISO), master out slave in (MOSI), serial clock (SCK), and slave select ($\overline{SS}$).

Any SPI output line must have its corresponding data direction bit in DDRD register set. If the DDR bit is clear, that line is disconnected from the SPI logic and becomes a general-purpose input. All SPI input lines are forced to act as inputs regardless of the state of the corresponding DDR bits in DDRD register.

SPIE — Serial Peripheral Interrupt Enable

Set the SPE bit to one to request a hardware interrupt sequence each time the SPIF or MODF status flag is set. SPI interrupts are inhibited if this bit is clear or if the I bit in the condition code register is one.

0 = SPI system interrupts disabled
1 = SPI system interrupts enabled

SPE — Serial Peripheral System Enable

When the SPE bit is set, the port D bit 2, 3, 4, and 5 pins are dedicated to the SPI function. If the SPI is in the master mode and DDRD bit 5 is set, then the port D bit 5 pin becomes a general-purpose output instead of the SS input.

0 = SPI system disabled
1 = SPI system enabled

DWOM — Port D Wired-OR Mode

DWOM affects all port D pins.

0 = Normal CMOS outputs
1 = Open-drain outputs

MSTR — Master Mode Select

0 = Slave mode
1 = Master mode

CPOL — Clock Polarity

When the clock polarity bit is cleared and data is not being transferred, the SCK pin of the master device has a steady state low value. When CPOL is set, SCK idles high. Refer to **Figure 8-2** and **8.2.1 Clock Phase and Polarity Controls**.

CPHA — Clock Phase

The clock phase bit, in conjunction with the CPOL bit, controls the clock-data relationship between master and slave. The CPHA bit selects one of two different clocking protocols. Refer to **Figure 8-2** and **8.2.1 Clock Phase and Polarity Controls**.

SPR[1:0] — SPI Clock Rate Selects

These two bits select the SPI clock (SCK) rate when the device is configured as master. When the device is configured as slave, these bits have no effect. Refer to **Table 8-1**.

Table 8-1 SPI Clock Rates

SPR[1:0]	E Clock Divide By	Frequency at E = 2 MHz	Frequency at E = 3 MHz	Frequency at E = 4 MHz	Frequency at E = 5 MHz
0 0	2	1.0 MHz	1.5 MHz	2.0 MHz	2.5 MHz
0 1	4	500 kHz	750 kHz	1.0 MHz	625 kHz
1 0	16	125 kHz	187.5 kHz	250 kHz	156.25 kHz
1 1	32	62.5 kHz	93.7 kHz	125 kHz	78.125 kHz

8.5.2 Serial Peripheral Status

SPSR — Serial Peripheral Status Register **$1029**

	Bit 7	6	5	4	3	2	1	Bit 0
	SPIF	WCOL	—	MODF	—	—	—	—
RESET:	0	0	0	0	0	0	0	0

SPIF — SPI Interrupt Complete Flag

SPIF is set upon completion of data transfer between the processor and the external device. If SPIF goes high, and if SPIE is set, a serial peripheral interrupt is generated. To clear the SPIF bit, read the SPSR then access the SPDR. Unless SPSR is read (with SPIF set) first, attempts to write SPDR are inhibited.

WCOL — Write Collision

Clearing the WCOL bit is accomplished by reading the SPSR followed by an access of SPDR. Refer to **8.3.4 Slave Select** and **8.4 SPI System Errors**.

0 = No write collision
1 = Write collision

Bit 5 — Not implemented

Always reads zero

MODF — Mode Fault

To clear the MODF bit, read the SPSR then write to the SPCR. Refer to **8.3.4 Slave Select** and **8.4 SPI System Errors**.

0 = No mode fault
1 = Mode fault

Bits [3:0] — Not implemented

Always read zero

8.5.3 Serial Peripheral Data Register

The SPDR is used when transmitting or receiving data on the serial bus. Only a write to this register initiates transmission or reception of a byte, and this only occurs in the master device. At the completion of transferring a byte of data, the SPIF status bit is set in both the master and slave devices.

A read of the SPDR is actually a read of a buffer. To prevent an overrun and the loss of the byte that caused the overrun, the first SPIF must be cleared by the time a second transfer of data from the shift register to the read buffer is initiated.

SPDR — SPI Data Register **$102A**

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

SPI is double buffered in and single buffered out.

Table 9-1 Timer Summary

Control Bits PR1, PR0	XTAL Frequencies				
	4.0 MHz	**8.0 MHz**	**12.0 MHz**	**16.0 MHz**	**Other Rates**
	1.0 MHz	**2.0 MHz**	**3.0 MHz**	**4.0 MHz**	**(E)**
	1000 ns	**500 ns**	**333 ns**	**250 ns**	**(1/E)**
	Main Timer Count Rates				
0 0 1 count — overflow —	1.0 µs 65.536 ms	500 ns 32.768 ms	333 ns 21.845 ms	250 ns 16.384 ms	(1/E) (2^{16}/E)
0 1 1 count — overflow —	4.0 µs 262.14 ms	2.0 µs 131.07 ms	1.333 µs 87.381 ms	1.0 µs 65.536 ms	(4/E) (2^{18}/E)
1 0 1 count — overflow —	8.0 µs 524.29 ms	4.0 µs 262.14 ms	2.667 µs 174.76 ms	2.0 µs 131.07 ms	(8/E) (2^{19}/E)
1 1 1 count — overflow —	16.0 µs 1.049 s	8.0 µs 524.29 ms	5.333 µs 349.52 ms	4.0 µs 262.14 ms	(16/E) (2^{20}/E)

9.1 Timer Structure

Figure 9-2 shows the capture/compare system block diagram. The port A pin control block includes logic for timer functions and for general-purpose I/O. For pins PA3, PA2, PA1, and PA0, this block contains both the edge-detection logic and the control logic that enables the selection of which edge triggers an input capture. The digital level on PA[3:0] can be read at any time (read PORTA register), even if the pin is being used for the input capture function. Pins PA[6:3] are used for either general-purpose I/O, or as output compare pins. When one of these pins is being used for an output compare function, it cannot be written directly as if it were a general-purpose output. Each of the output compare functions (OC[5:2]) is related to one of the port A output pins. Output compare one (OC1) has extra control logic, allowing it optional control of any combination of the PA[7:3] pins. The PA7 pin can be used as a general-purpose I/O pin, as an input to the pulse accumulator, or as an OC1 output pin.

PACTL — Pulse Accumulator Control **$1026**

	Bit 7	6	5	4	3	2	1	Bit 0
	—	PAEN	PAMOD	PEDGE	—	I4/O5	RTR1	RTR0
RESET:	0	0	0	0	0	0	0	0

Bit 7 — Not implemented
Always reads zero

PAEN — Pulse Accumulator System Enable
0 = Pulse accumulator disabled
1 = Pulse accumulator enabled

PAMOD — Pulse Accumulator Mode
0 = Event counter
1 = Gated time accumulation

PEDGE — Pulse Accumulator Edge Control
This bit has different meanings depending on the state of the PAMOD bit, as shown in **Table 9-6**.

Table 9-6 Pulse Accumulator Edge Detection Control

PAMOD	PEDGE	Action on Clock
0	0	PAI falling edge increments the counter.
0	1	PAI rising edge increments the counter.
1	0	A zero on PAI inhibits counting.
1	1	A one on PAI inhibits counting.

Bit 3 — Not implemented
Always reads zero

I4/O5 — Input Capture 4/Output Compare 5
0 = Output compare 5 function enable (No IC4)
1 = Input capture 4 function enable (No OC5)

RTR[1:0] — RTI Interrupt Rate Selects
Refer to **9.4 Real-Time Interrupt**.

9.6.2 Pulse Accumulator Count Register

This 8-bit read/write register contains the count of external input events at the PAI input, or the accumulated count. The PACNT is readable even if PAI is not active in gated time accumulation mode. The counter is not affected by reset and can be read or written at any time. Counting is synchronized to the internal PH2 clock so that incrementing and reading occur during opposite half cycles.

PACNT — Pulse Accumulator Count **$1027**

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

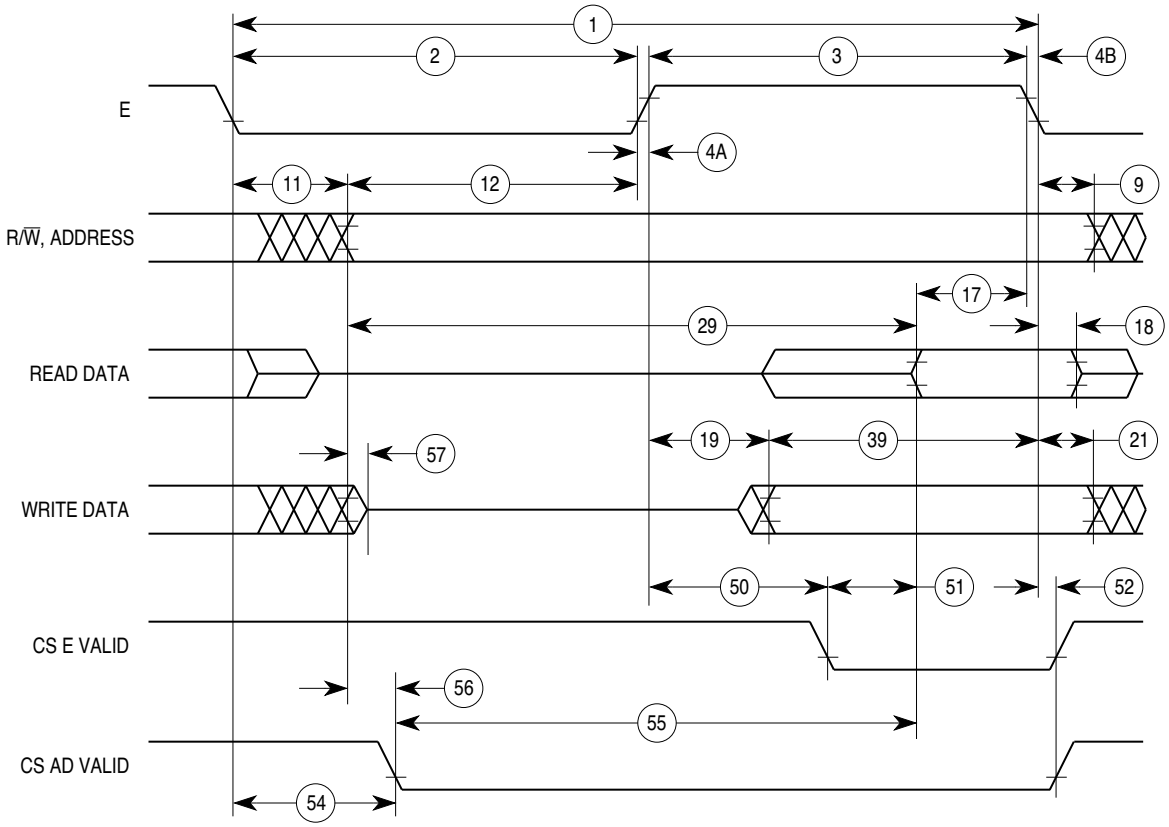
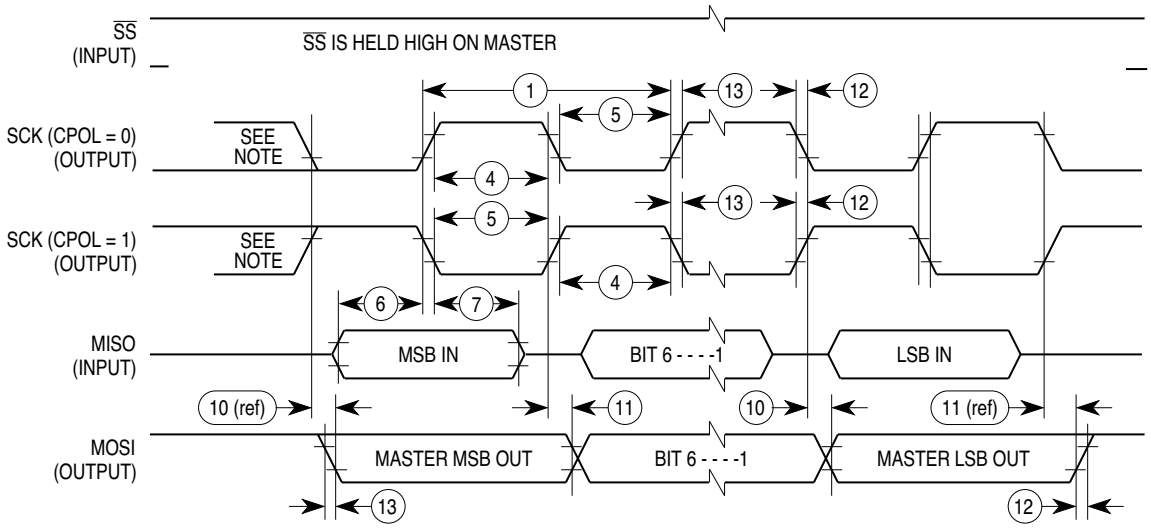
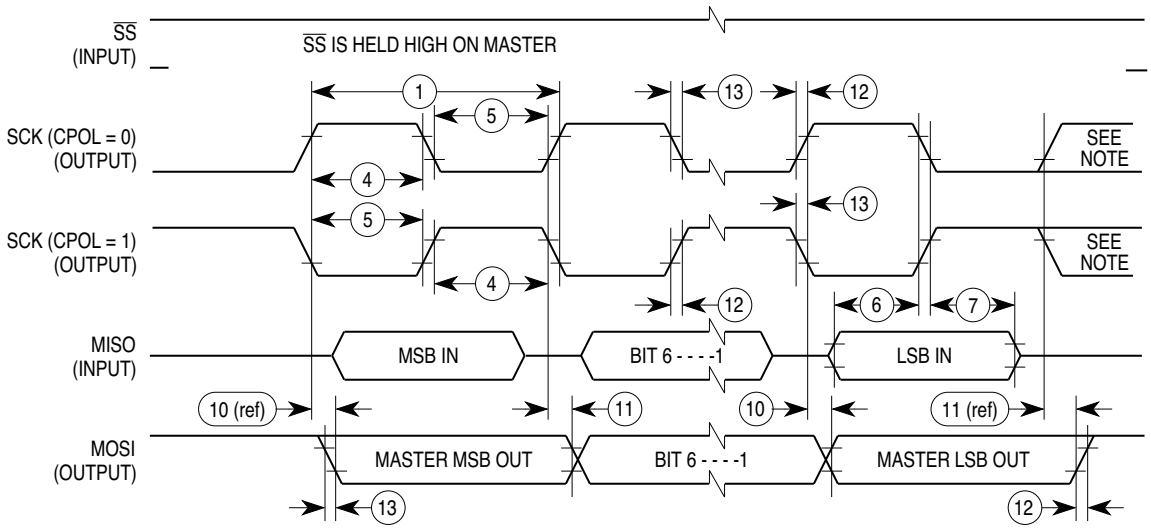


Figure A-9 Expansion Bus Timing



NOTE: This first clock edge is generated internally but is not seen at the SCK pin.

Figure A-10 SPI Master Timing (CPHA = 0)



NOTE: This last clock edge is generated internally but is not seen at the SCK pin.

Figure A-11 SPI Master Timing (CPHA = 1)

APPENDIX CDEVELOPMENT SUPPORT

C.1 MC68HC11F1 Development Tools

The following table and text provide a reference to development tools for the MC68HC11F1 microcontrollers. For more complete information refer to the appropriate manual for each system.

Table C-1 MC68HC11F1 Development Tools

Device	Evaluation Systems	Modular Development Systems
MC68HC11F1	M68HC11F1EVS	MMDS11*

* For MC68HC11F1 support, the MMDS11 must be used with an MC68HC11F1 emulator module.

C.2 MC68HC11EVS — Evaluation System

The EVS is an economical tool for designing, debugging, and evaluating target systems based on the MC68HC11F1 MCU. The two printed circuit boards that comprise the EVS are the MC68HC11F1EM emulator module (EM) and the M68HC11PFB platform board (PFB).

- Monitor/debugger firmware
- One-line assembler/disassembler
- Host computer download capability
- Dual memory maps:
  - 64 Kbyte monitor map that includes 16 Kbytes of monitor EPROM
  - MC68HC11F1 user map that includes 64 Kbytes of emulation RAM
- OTPROM, EPROM, and EEPROM MCU programmer
- MCU extension I/O port for single-chip, expanded, and special-test operation modes
- RS-232C terminal and host I/O ports
- Logic analyzer connector

C.3 M68MMDS11 — Modular Development System for M68HC11 Devices

The M68MMDS11 Freescale Modular Development System (MMDS11) is a tool for developing embedded systems based on M68HC11 MCUs. The MMDS11 is an emulator system that provides an on-screen bus state analyzer and real-time memory monitoring windows. An integrated design environment includes an editor, an assembler, the user interface, and source-level debug capability. These features significantly reduce the time necessary to develop and debug an embedded MCU system. The compact unit requires minimum space.

- Real-time, non-intrusive, in-circuit emulation
- Assembly-language source-level debugging