Welcome to E-XFL.COM

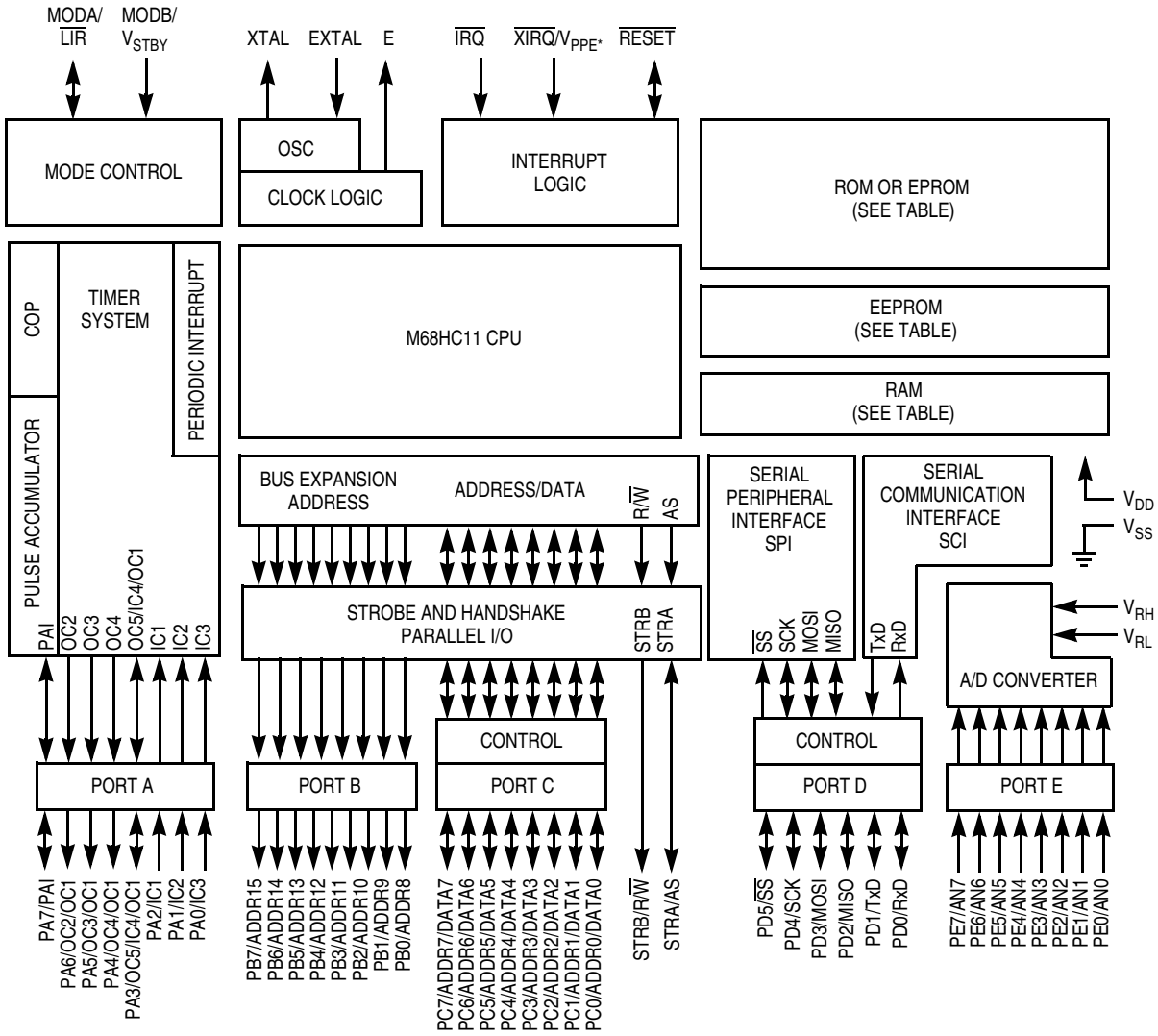
What is "Embedded - Microcontrollers"?

"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	Active
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	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LCC (J-Lead)
Supplier Device Package	52-PLCC (19.13x19.13)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68hc11e0cfne3r

List of Chapters



DEVICE	RAM	ROM	EPROM	EEPROM
MC68HC11E0	512	—	—	—
MC68HC11E1	512	—	—	512
MC68HC11E9	512	12 K	—	512
MC68HC711E9	512	—	12 K	512
MC68HC11E20	768	20 K	—	512
MC68HC711E20	768	—	20 K	512
MC68HC811E2	256	—	—	2048

* V_{PPE} applies only to devices with EPROM/OTPROM.

Figure 1-1. M68HC11 E-Series Block Diagram

1.4.9 STRA/AS

The strobe A (STRA) and address strobe (AS) pin performs either of two separate functions, depending on the operating mode:

- In single-chip mode, STRA performs an input handshake (strobe input) function.
- In the expanded multiplexed mode, AS provides an address strobe function.

AS can be used to demultiplex the address and data signals at port C. Refer to Chapter 2 Operating Modes and On-Chip Memory.

1.4.10 STRB/R/$\overline{W}$

The strobe B (STRB) and read/write (R/$\overline{W}$) pin act as either an output strobe or as a data bus direction indicator, depending on the operating mode.

In single-chip operating mode, STRB acts as a programmable strobe for handshake with other parallel devices. Refer to Chapter 6 Parallel Input/Output (I/O) Ports for further information.

In expanded multiplexed operating mode, R/$\overline{W}$ is used to indicate the direction of transfers on the external data bus. A low on the R/$\overline{W}$ pin indicates data is being written to the external data bus. A high on this pin indicates that a read cycle is in progress. R/$\overline{W}$ stays low during consecutive data bus write cycles, such as a double-byte store. It is possible for data to be driven out of port C, if internal read visibility (IRV) is enabled and an internal address is read, even though R/$\overline{W}$ is in a high-impedance state. Refer to Chapter 2 Operating Modes and On-Chip Memory for more information about IRVNE (internal read visibility not E).

1.4.11 Port Signals

Port pins have different functions in different operating modes. Pin functions for port A, port D, and port E are independent of operating modes. Port B and port C, however, are affected by operating mode. Port B provides eight general-purpose output signals in single-chip operating modes. When the microcontroller is in expanded multiplexed operating mode, port B pins are the eight high-order address lines.

Port C provides eight general-purpose input/output signals when the MCU is in the single-chip operating mode. When the microcontroller is in the expanded multiplexed operating mode, port C pins are a multiplexed address/data bus.

Refer to Table 1-1 for a functional description of the 40 port signals within different operating modes. Terminate unused inputs and input/output (I/O) pins configured as inputs high or low.

1.4.12 Port A

In all operating modes, port A can be configured for three timer input capture (IC) functions and four timer output compare (OC) functions. An additional pin can be configured as either the fourth IC or the fifth OC. Any port A pin that is not currently being used for a timer function can be used as either a general-purpose input or output line. Only port A pins PA7 and PA3 have an associated data direction control bit that allows the pin to be selectively configured as input or output. Bits DDRA7 and DDRA3 located in PACTL register control data direction for PA7 and PA3, respectively. All other port A pins are fixed as either input or output.

PA7 can function as general-purpose I/O or as timer output compare for OC1. PA7 is also the input to the pulse accumulator, even while functioning as a general-purpose I/O or an OC1 output.

Table 2-1. Hardware Mode Select Summary

Input Levels at Reset		Mode	Control Bits in HPRIO (Latched at Reset)		
MODB	MODA		RBOOT	SMOD	MDA
1	0	Single chip	0	0	0
1	1	Expanded	0	0	1
0	0	Bootstrap	1	1	0
0	1	Special test	0	1	1

A normal mode is selected when MODB is logic 1 during reset. One of three reset vectors is fetched from address $FFFA–$FFFF, and program execution begins from the address indicated by this vector. If MODB is logic 0 during reset, the special mode reset vector is fetched from addresses $BFFA–$BFFF, and software has access to special test features. Refer to Chapter 5 Resets and Interrupts.

Address: $103C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RBOOT(1)	SMOD(1)	MDA(1)	IRV(NE)(1)	PSEL3	PSEL2	PSEL1	PSEL0
Resets:								
Single chip:	0	0	0	0	0	1	1	0
Expanded:	0	0	1	0	0	1	1	0
Bootstrap:	1	1	0	0	0	1	1	0
Test:	0	1	1	1	0	1	1	0

1. The reset values depend on the mode selected at the RESET pin rising edge.

Figure 2-9. Highest Priority I-Bit Interrupt and Miscellaneous Register (HPRIO)

RBOOT — Read Bootstrap ROM Bit

Valid only when SMOD is set (bootstrap or special test mode); can be written only in special modes

0 = Bootloader ROM disabled and not in map

1 = Bootloader ROM enabled and in map at $BE00–$BFFF

SMOD and MDA — Special Mode Select and Mode Select A Bits

The initial value of SMOD is the inverse of the logic level present on the MODB pin at the rising edge of reset. The initial value of MDA equals the logic level present on the MODA pin at the rising edge of reset. These two bits can be read at any time. They can be written anytime in special modes. MDA can be written only once in normal modes. SMOD cannot be set once it has been cleared.

Input		Mode	Latched at Reset	
MODB	MODA		SMOD	MDA
1	0	Single chip	0	0
1	1	Expanded	0	1

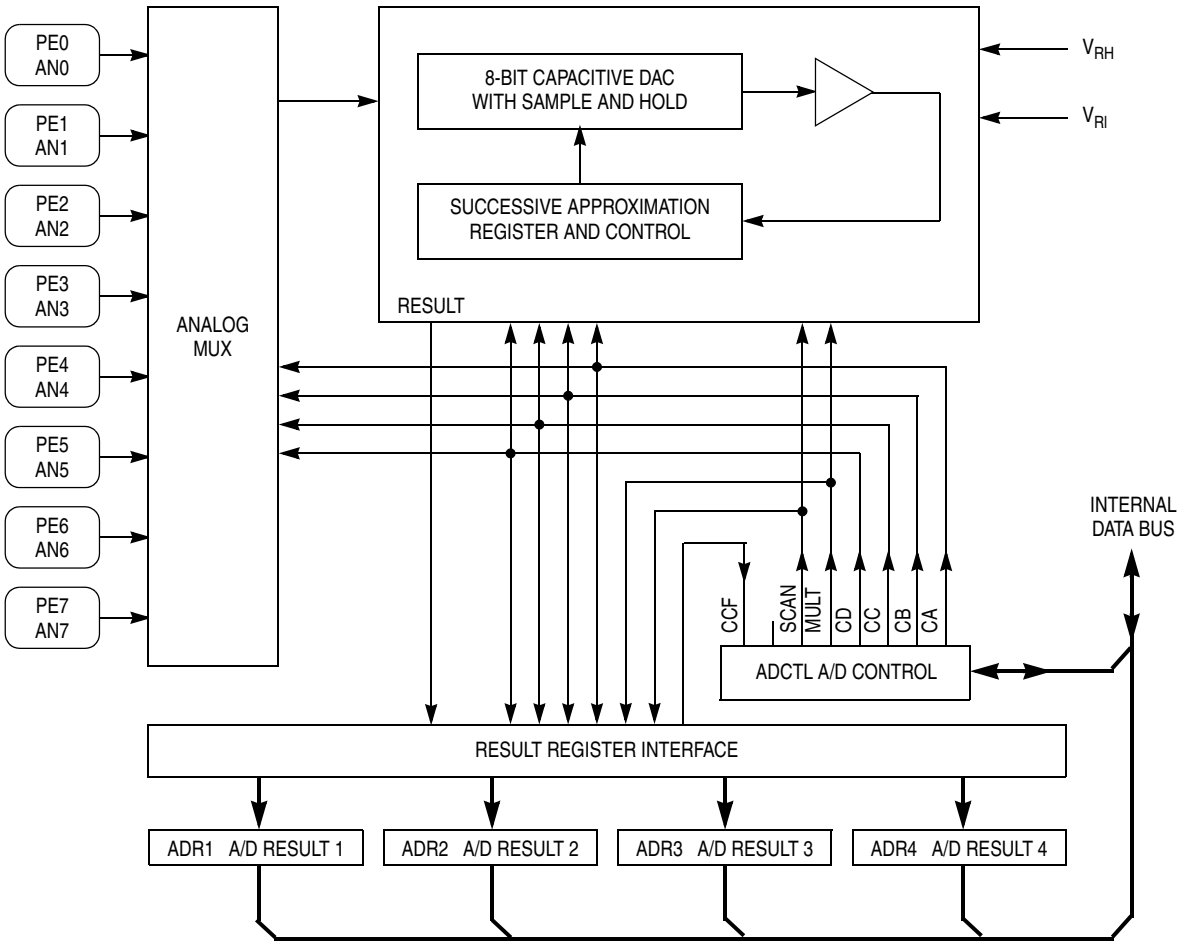
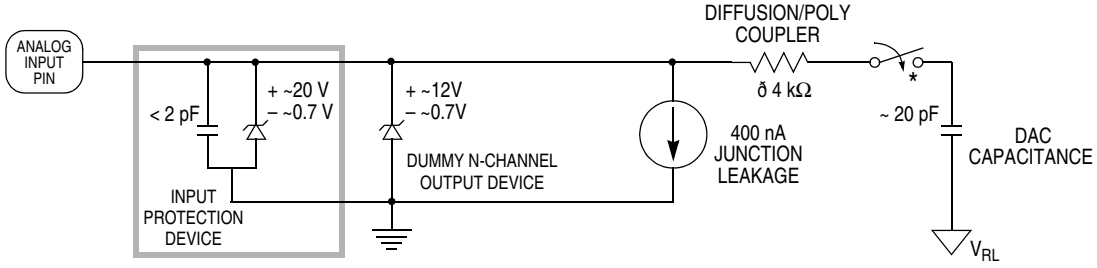


Figure 3-1. A/D Converter Block Diagram

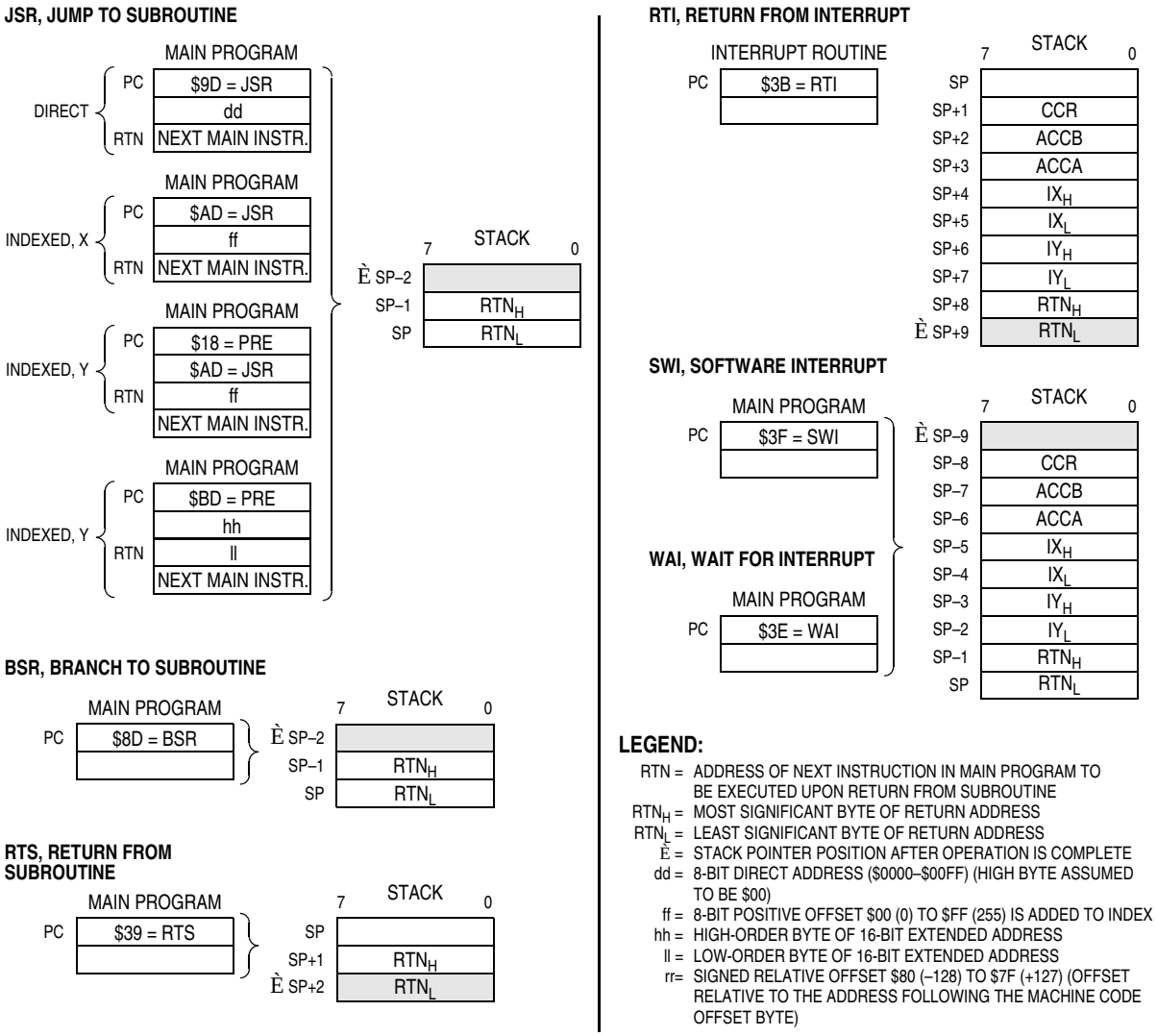


* THIS ANALOG SWITCH IS CLOSED ONLY DURING THE 12-CYCLE SAMPLE TIME.

Figure 3-2. Electrical Model of an A/D Input Pin (Sample Mode)

At the end of the interrupt service routine, an return-from interrupt (RTI) instruction is executed. The RTI instruction causes the saved registers to be pulled off the stack in reverse order. Program execution resumes at the return address.

Certain instructions push and pull the A and B accumulators and the X and Y index registers and are often used to preserve program context. For example, pushing accumulator A onto the stack when entering a subroutine that uses accumulator A and then pulling accumulator A off the stack just before leaving the subroutine ensures that the contents of a register will be the same after returning from the subroutine as it was before starting the subroutine.



LEGEND:

RTN = ADDRESS OF NEXT INSTRUCTION IN MAIN PROGRAM TO BE EXECUTED UPON RETURN FROM SUBROUTINE
RTN_H = MOST SIGNIFICANT BYTE OF RETURN ADDRESS
RTN_L = LEAST SIGNIFICANT BYTE OF RETURN ADDRESS
È = STACK POINTER POSITION AFTER OPERATION IS COMPLETE
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
ll = LOW-ORDER BYTE OF 16-BIT EXTENDED ADDRESS
rr= SIGNED RELATIVE OFFSET $80 (–128) TO $7F (+127) (OFFSET RELATIVE TO THE ADDRESS FOLLOWING THE MACHINE CODE OFFSET BYTE)

Figure 4-2. Stacking Operations

Address	$103A							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Reset:	0	0	0	0	0	0	0	0

Figure 5-1. Arm/Reset COP Timer Circuitry Register (COPRST)

Complete this 2-step reset sequence to service the COP timer:

1. Write $55 to COPRST to arm the COP timer clearing mechanism.
2. Write $AA to COPRST to clear the COP timer.

Performing instructions between these two steps is possible as long as both steps are completed in the correct sequence before the timer times out.

5.2.4 Clock Monitor Reset

The clock monitor circuit is based on an internal resistor capacitor (RC) time delay. If no MCU clock edges are detected within this RC time delay, the clock monitor can optionally generate a system reset. The clock monitor function is enabled or disabled by the CME control bit in the OPTION register. The presence of a timeout is determined by the RC delay, which allows the clock monitor to operate without any MCU clocks.

Clock monitor is used as a backup for the COP system. Because the COP needs a clock to function, it is disabled when the clock stops. Therefore, the clock monitor system can detect clock failures not detected by the COP system.

Semiconductor wafer processing causes variations of the RC timeout values between individual devices. An E-clock frequency below 10 kHz is detected as a clock monitor error. An E-clock frequency of 200 kHz or more prevents clock monitor errors. Using the clock monitor function when the E-clock is below 200 kHz is not recommended.

Special considerations are needed when a STOP instruction is executed and the clock monitor is enabled. Because the STOP function causes the clocks to be halted, the clock monitor function generates a reset sequence if it is enabled at the time the stop mode was initiated. Before executing a STOP instruction, clear the CME bit in the OPTION register to 0 to disable the clock monitor. After recovery from STOP, set the CME bit to logic 1 to enable the clock monitor. Alternatively, executing a STOP instruction with the CME bit set to logic 1 can be used as a software initiated reset.

Table 5-3. Highest Priority Interrupt Selection

PSEL[3:0]	Interrupt Source Promoted
0 0 0 0	Timer overflow
0 0 0 1	Pulse accumulator overflow
0 0 1 0	Pulse accumulator input edge
0 0 1 1	SPI serial transfer complete
0 1 0 0	SCI serial system
0 1 0 1	Reserved (default to $\overline{\text{IRQ}}$)
0 1 1 0	$\overline{\text{IRQ}}$ (external pin or parallel I/O)
0 1 1 1	Real-time interrupt
1 0 0 0	Timer input capture 1
1 0 0 1	Timer input capture 2
1 0 1 0	Timer input capture 3
1 0 1 1	Timer output compare 1
1 1 0 0	Timer output compare 2
1 1 0 1	Timer output compare 3
1 1 1 0	Timer output compare 4
1 1 1 1	Timer input capture 4/output compare 5

5.5 Interrupts

The MCU has 18 interrupt vectors that support 22 interrupt sources. The 15 maskable interrupts are generated by on-chip peripheral systems. These interrupts are recognized when the global interrupt mask bit (I) in the condition code register (CCR) is clear. The three non-maskable interrupt sources are illegal opcode trap, software interrupt, and $\overline{\text{XIRQ}}$ pin. Refer to Table 5-4, which shows the interrupt sources and vector assignments for each source.

For some interrupt sources, such as the SCI interrupts, the flags are automatically cleared during the normal course of responding to the interrupt requests. For example, the RDRF flag in the SCI system is cleared by the automatic clearing mechanism consisting of a read of the SCI status register while RDRF is set, followed by a read of the SCI data register. The normal response to an RDRF interrupt request would be to read the SCI status register to check for receive errors, then to read the received data from the SCI data register. These steps satisfy the automatic clearing mechanism without requiring special instructions.

5.5.4 Software Interrupt (SWI)

SWI is an instruction, and thus cannot be interrupted until complete. SWI is not inhibited by the global mask bits in the CCR. Because execution of SWI sets the I mask bit, once an SWI interrupt begins, other interrupts are inhibited until SWI is complete, or until user software clears the I bit in the CCR.

5.5.5 Maskable Interrupts

The maskable interrupt structure of the MCU can be extended to include additional external interrupt sources through the $\overline{\text{IRQ}}$ pin. The default configuration of this pin is a low-level sensitive wired-OR network. When an event triggers an interrupt, a software accessible interrupt flag is set. When enabled, this flag causes a constant request for interrupt service. After the flag is cleared, the service request is released.

5.5.6 Reset and Interrupt Processing

Figure 5-5 and Figure 5-6 illustrate the reset and interrupt process. Figure 5-5 illustrates how the CPU begins from a reset and how interrupt detection relates to normal opcode fetches. Figure 5-6 is an expansion of a block in Figure 5-5 and illustrates interrupt priorities. Figure 5-7 shows the resolution of interrupt sources within the SCI subsystem.

5.6 Low-Power Operation

Both stop mode and wait mode suspend CPU operation until a reset or interrupt occurs. Wait mode suspends processing and reduces power consumption to an intermediate level. Stop mode turns off all on-chip clocks and reduces power consumption to an absolute minimum while retaining the contents of the entire RAM array.

5.6.1 Wait Mode

The WAI opcode places the MCU in wait mode, during which the CPU registers are stacked and CPU processing is suspended until a qualified interrupt is detected. The interrupt can be an external $\overline{\text{IRQ}}$, an $\overline{\text{XIRQ}}$, or any of the internally generated interrupts, such as the timer or serial interrupts. The on-chip crystal oscillator remains active throughout the wait standby period.

The reduction of power in the wait condition depends on how many internal clock signals driving on-chip peripheral functions can be shut down. The CPU is always shut down during wait. While in the wait state, the address/data bus repeatedly runs read cycles to the address where the CCR contents were stacked. The MCU leaves the wait state when it senses any interrupt that has not been masked.

The free-running timer system is shut down only if the I bit is set to 1 and the COP system is disabled by NOCOP being set to 1. Several other systems also can be in a reduced power-consumption state depending on the state of software-controlled configuration control bits. Power consumption by the analog-to-digital (A/D) converter is not affected significantly by the wait condition. However, the A/D converter current can be eliminated by writing the ADPU bit to 0. The SPI system is enabled or disabled by the SPE control bit. The SCI transmitter is enabled or disabled by the TE bit, and the SCI receiver is enabled or disabled by the RE bit. Therefore, the power consumption in wait is dependent on the particular application.

SCR[2:0] — SCI Baud Rate Select Bits

Selects receiver and transmitter bit rate based on output from baud rate prescaler stage. Refer to Figure 7-8 and Figure 7-9.

The prescaler bits, SCP[2:0], determine the highest baud rate, and the SCR[2:0] bits select an additional binary submultiple (÷1, ÷2, ÷4, through ÷128) of this highest baud rate. The result of these two dividers in series is the 16X receiver baud rate clock. The SCR[2:0] bits are not affected by reset and can be changed at any time, although they should not be changed when any SCI transfer is in progress.

Figure 7-8 and Figure 7-9 illustrate the SCI baud rate timing chain. The prescaler select bits determine the highest baud rate. The rate select bits determine additional divide by two stages to arrive at the receiver timing (RT) clock rate. The baud rate clock is the result of dividing the RT clock by 16.

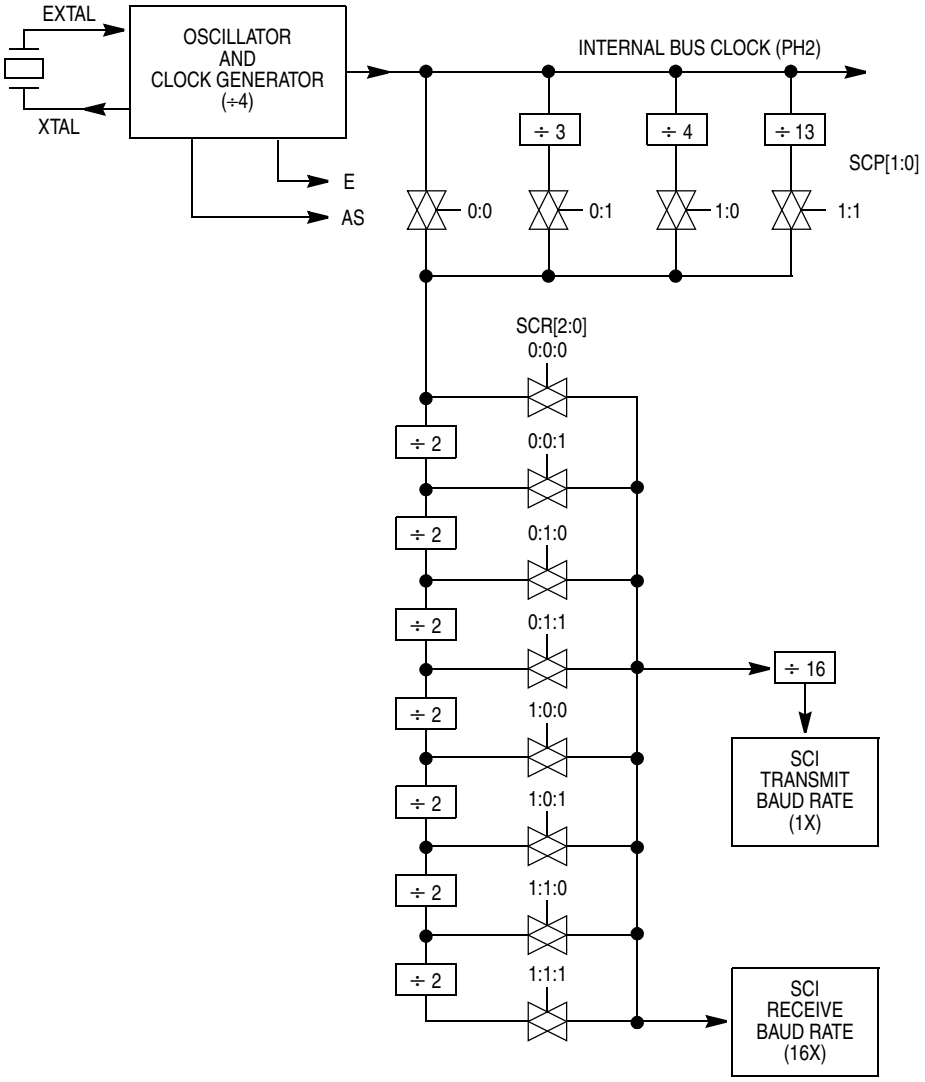
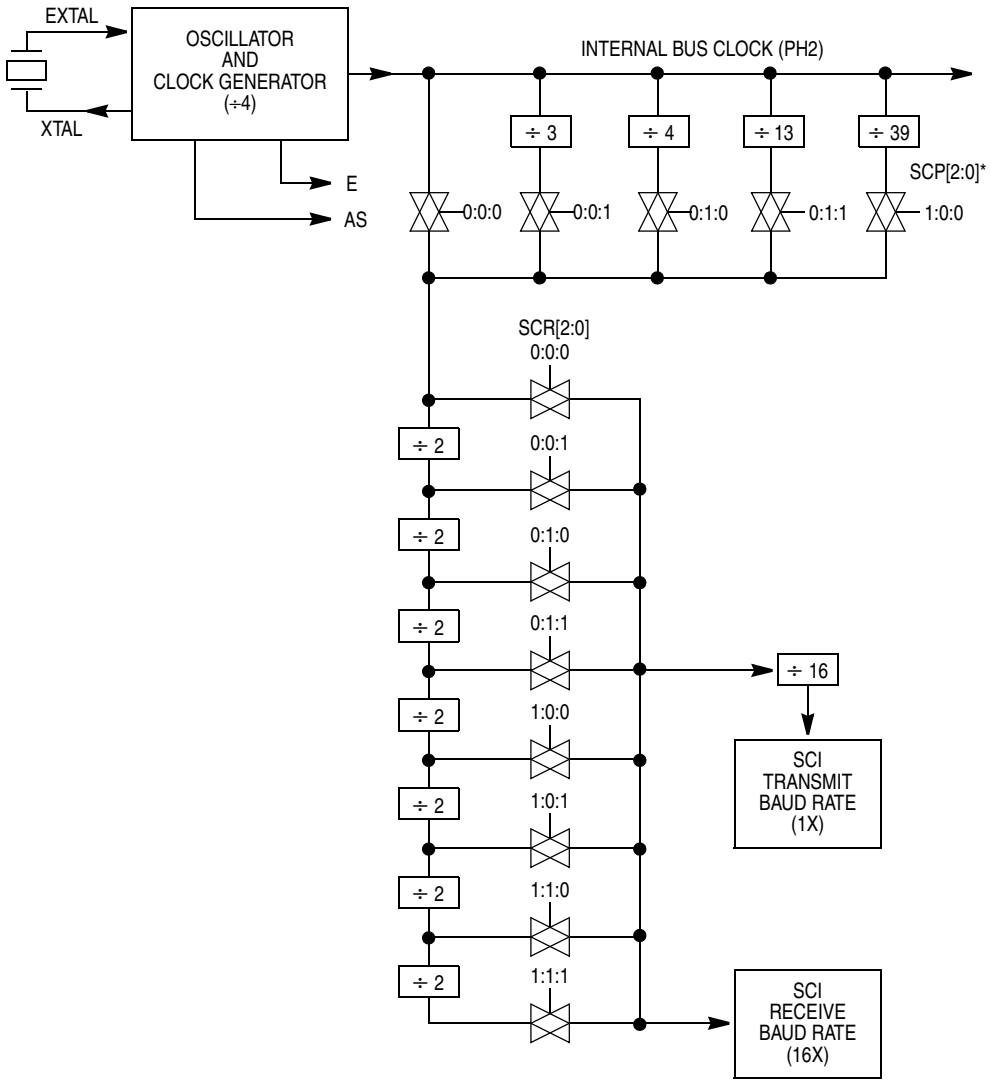


Figure 7-8. SCI Baud Rate Generator Block Diagram



*SCP2 is present only on MC68HC(7)11E20.

Figure 7-9. MC68HC(7)11E20 SCI Baud Rate Generator Block Diagram

7.8 Status Flags and Interrupts

The SCI transmitter has two status flags. These status flags can be read by software (polled) to tell when the corresponding condition exists. Alternatively, a local interrupt enable bit can be set to enable each of these status conditions to generate interrupt requests when the corresponding condition is present. Status flags are automatically set by hardware logic conditions, but must be cleared by software, which provides an interlock mechanism that enables logic to know when software has noticed the status indication. The software clearing sequence for these flags is automatic. Functions that are normally performed in response to the status flags also satisfy the conditions of the clearing sequence.

A write collision is normally a slave error because a slave has no control over when a master initiates a transfer. A master knows when a transfer is in progress, so there is no reason for a master to generate a write-collision error, although the SPI logic can detect write collisions in both master and slave devices.

The SPI configuration determines the characteristics of a transfer in progress. For a master, a transfer begins when data is written to SPDR and ends when SPIF is set. For a slave with CPHA equal to 0, a transfer starts when $\overline{SS}$ goes low and ends when $\overline{SS}$ returns high. In this case, SPIF is set at the middle of the eighth SCK cycle when data is transferred from the shifter to the parallel data register, but the transfer is still in progress until $\overline{SS}$ goes high. For a slave with CPHA equal to 1, transfer begins when the SCK line goes to its active level, which is the edge at the beginning of the first SCK cycle. The transfer ends in a slave in which CPHA equals 1 when SPIF is set.

8.7 SPI Registers

The three SPI registers are:

- Serial peripheral control register (SPCR)
- Serial peripheral status register (SPSR)
- Serial peripheral data register (SPDR)

These registers provide control, status, and data storage functions.

8.7.1 Serial Peripheral Control Register

Address: $1028

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0
Reset:	0	0	0	0	0	1	U	U

U = Unaffected

Figure 8-3. Serial Peripheral Control Register (SPCR)

SPIE — Serial Peripheral Interrupt Enable Bit

Set the SPE bit to 1 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 1.

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

SPE — Serial Peripheral System Enable Bit

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 $\overline{SS}$ input.

0 = SPI system disabled
1 = SPI system enabled

DWOM — Port D Wired-OR Mode Bit

DWOM affects all port D pins.

0 = Normal CMOS outputs
1 = Open-drain outputs

10.11 Peripheral Port Timing

Characteristic[1] [2]	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of operation E-clock frequency	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	333	—	ns
Peripheral data setup time MCU read of ports A, C, D, and E	t_{PDSU}	100	—	100	—	100	—	ns
Peripheral data hold time MCU read of ports A, C, D, and E	t_{PDH}	50	—	50	—	50	—	ns
Delay time, peripheral data write t_{PWD} = 1/4 t_{CYC}+ 100 ns MCU writes to port A MCU writes to ports B, C, and D	t_{PWD}	 — —	 200 350	 — —	 200 225	 — —	 200 183	ns
Port C input data setup time	t_{IS}	60	—	60	—	60	—	ns
Port C input data hold time	t_{IH}	100	—	100	—	100	—	ns
Delay time, E fall to STRB t_{DEB} = 1/4 t_{CYC}+ 100 ns	t_{DEB}	—	350	—	225	—	183	ns
Setup time, STRA asserted to E fall[3]	t_{AES}	0	—	0	—	0	—	ns
Delay time, STRA asserted to port C data output valid	t_{PCD}	—	100	—	100	—	100	ns
Hold time, STRA negated to port C data	t_{PCH}	10	—	10	—	10	—	ns
3-state hold time	t_{PCZ}	—	150	—	150	—	150	ns

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Ports C and D timing is valid for active drive. (CWOM and DWOM bits are not set in PIOC and SPCR registers, respectively.)
3. If this setup time is met, STRB acknowledges in the next cycle. If it is not met, the response may be delayed one more cycle.

10.15 Expansion Bus Timing Characteristics

Num	Characteristic[1]	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
			Min	Max	Min	Max	Min	Max	
	Frequency of operation (E-clock frequency)	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
1	Cycle time	t_{CYC}	1000	—	500	—	333	—	ns
2	Pulse width, E low[2], PW_{EL} = 1/2 t_{CYC}–23 ns	PW_{EL}	477	—	227	—	146	—	ns
3	Pulse width, E high[2], PW_{EH} = 1/2 t_{CYC}–28 ns	PW_{EH}	472	—	222	—	141	—	ns
4a	E and AS rise time	t_r	—	20	—	20	—	20	ns
4b	E and AS fall time	t_f	—	20	—	20	—	15	ns
9	Address hold time[2] [3]a, t_{AH} = 1/8 t_{CYC}–29.5 ns	t_{AH}	95.5	—	33	—	26	—	ns
12	Non-multiplexed address valid time to E rise $t_{AV} = PW_{EL} – (t_{ASD} + 80\ ns)$[2] [3]a	t_{AV}	281.5	—	94	—	54	—	ns
17	Read data setup time	t_{DSR}	30	—	30	—	30	—	ns
18	Read data hold time, max = t_{MAD}	t_{DHR}	0	145.5	0	83	0	51	ns
19	Write data delay time, t_{DDW} = 1/8 t_{CYC}+ 65.5 ns[2] [3]a	t_{DDW}	—	190.5	—	128		71	ns
21	Write data hold time, t_{DHW} = 1/8 t_{CYC}–29.5 ns[2] [3]a	t_{DHW}	95.5	—	33	—	26	—	ns
22	Multiplexed address valid time to E rise $t_{AVM} = PW_{EL} – (t_{ASD} + 90\ ns)$[2] [3]a	t_{AVM}	271.5	—	84	—	54	—	ns
24	Multiplexed address valid time to AS fall $t_{ASL} = PW_{ASH} – 70\ ns$[2]	t_{ASL}	151	—	26	—	13	—	ns
25	Multiplexed address hold time t_{AHL} = 1/8 t_{CYC}–29.5 ns[2] [3]b	t_{AHL}	95.5	—	33	—	31	—	ns
26	Delay time, E to AS rise, t_{ASD} = 1/8 t_{CYC}–9.5 ns[2] [3]a	t_{ASD}	115.5	—	53	—	31	—	ns
27	Pulse width, AS high, PW_{ASH} = 1/4 t_{CYC}–29 ns[2]	PW_{ASH}	221	—	96	—	63	—	ns
28	Delay time, AS to E rise, t_{ASED} = 1/8 t_{CYC}–9.5 ns[2] [3]b	t_{ASED}	115.5	—	53	—	31	—	ns
29	MPU address access time[3]a $t_{ACCA} = t_{CYC} – (PW_{EL} – t_{AVM}) – t_{DSR} – t_f$	t_{ACCA}	744.5	—	307	—	196	—	ns
35	MPU access time, $t_{ACCE} = PW_{EH} – t_{DSR}$	t_{ACCE}	—	442	—	192		111	ns
36	Multiplexed address delay (Previous cycle MPU read) $t_{MAD} = t_{ASD} + 30\ ns$[2] [3]a	t_{MAD}	145.5	—	83	—	51	—	ns

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Formula only for dc to 2 MHz
3. Input clocks with duty cycles other than 50% affect bus performance. Timing parameters affected by input clock duty cycle are identified by (a) and (b). To recalculate the approximate bus timing values, substitute the following expressions in place of 1/8 t_{CYC}in the above formulas, where applicable:
   (a) (1–dc) × 1/4 t_{CYC}
   (b) dc × 1/4 t_{CYC}

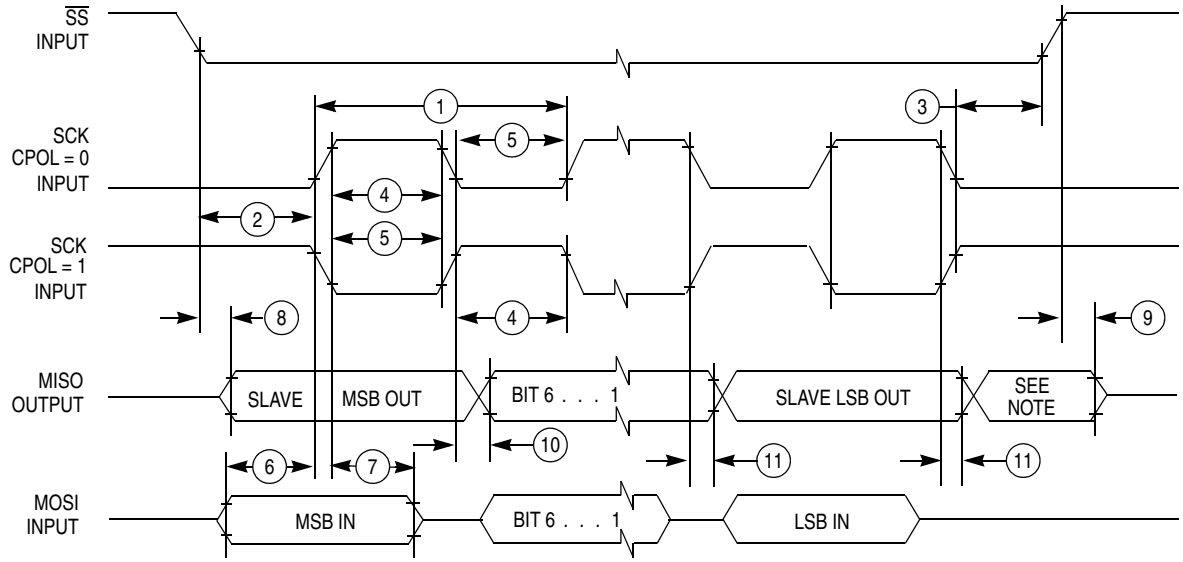
Where:
dc is the decimal value of duty cycle percentage (high time)

10.16 MC68L11E9/E20 Expansion Bus Timing Characteristics

Num	Characteristic[1]	Symbol	1.0 MHz		2.0 MHz		Unit
			Min	Max	Min	Max	
	Frequency of operation (E-clock frequency)	f_o	dc	1.0	dc	2.0	MHz
1	Cycle time	t_{CYC}	1000	—	500	—	ns
2	Pulse width, E low, PW_{EL} = 1/2 t_{CYC}–25 ns	PW_{EL}	475	—	225	—	ns
3	Pulse width, E high, PW_{EH} = 1/2 t_{CYC}–30 ns	PW_{EH}	470	—	220	—	ns
4a	E and AS rise time	t_r	—	25	—	25	ns
4b	E and AS fall time	t_f	—	25	—	25	ns
9	Address hold time[2] [2]a, t_{AH} = 1/8 t_{CYC}–30 ns	t_{AH}	95	—	33	—	ns
12	Non-multiplexed address valid time to E rise $t_{AV} = PW_{EL} – (t_{ASD} + 80\ ns)$[2]a	t_{AV}	275	—	88	—	ns
17	Read data setup time	t_{DSR}	30	—	30	—	ns
18	Read data hold time , max = t_{MAD}	t_{DHR}	0	150	0	88	ns
19	Write data delay time, t_{DDW} = 1/8 t_{CYC}+ 70 ns[2]a	t_{DDW}	—	195	—	133	ns
21	Write data hold time, t_{DHW} = 1/8 t_{CYC}–30 ns[2]a	t_{DHW}	95	—	33	—	ns
22	Multiplexed address valid time to E rise $t_{AVM} = PW_{EL} – (t_{ASD} + 90\ ns)$[2]a	t_{AVM}	268	—	78	—	ns
24	Multiplexed address valid time to AS fall $t_{ASL} = PW_{ASH} – 70\ ns$	t_{ASL}	150	—	25	—	ns
25	Multiplexed address hold time, t_{AHL} = 1/8 t_{CYC}–30 ns[2]b	t_{AHL}	95	—	33	—	ns
26	Delay time, E to AS rise, t_{ASD} = 1/8 t_{CYC}–5 ns[2]a	t_{ASD}	120	—	58	—	ns
27	Pulse width, AS high, PW_{ASH} = 1/4 t_{CYC}–30 ns	PW_{ASH}	220	—	95	—	ns
28	Delay time, AS to E rise, t_{ASED} = 1/8 t_{CYC}–5 ns[2]b	t_{ASED}	120	—	58	—	ns
29	MPU address access time[3]a $t_{ACCA} = t_{CYC} – (PW_{EL} – t_{AVM}) – t_{DSR} – t_f$	t_{ACCA}	735	—	298	—	ns
35	MPU access time, $t_{ACCE} = PW_{EH} – t_{DSR}$	t_{ACCE}	—	440	—	190	ns
36	Multiplexed address delay (Previous cycle MPU read) $t_{MAD} = t_{ASD} + 30\ ns$[2]a	t_{MAD}	150	—	88	—	ns

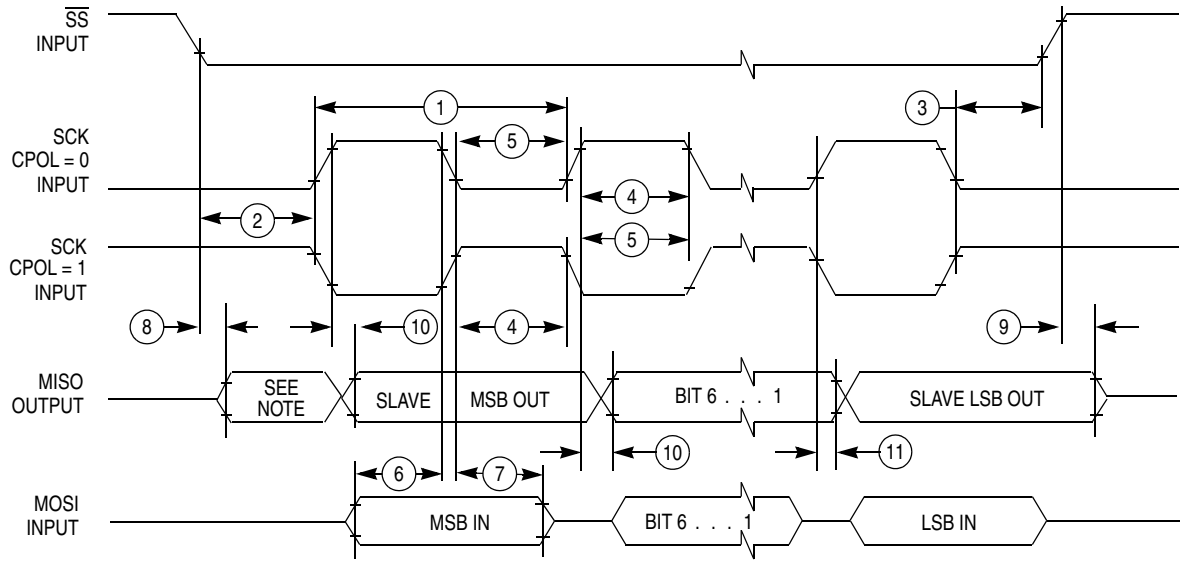
1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Input clocks with duty cycles other than 50% affect bus performance. Timing parameters affected by input clock duty cycle are identified by (a) and (b). To recalculate the approximate bus timing values, substitute the following expressions in place of 1/8 t_{CYC}in the above formulas, where applicable:
   (a) (1–dc) × 1/4 t_{CYC}
   (b) dc × 1/4 t_{CYC}

Where:
dc is the decimal value of duty cycle percentage (high time).



Note: Not defined but normally MSB of character just received

A) SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character previously transmitted

B) SPI Slave Timing (CPHA = 1)

Figure 11-15. SPI Timing Diagram (Sheet 2 of 2)

Bootstrap mode can also be used to interactively calibrate critical analog sensors. Since this calibration is done in the final assembled system, it can compensate for any errors in discrete interface circuitry and cabling between the sensor and the analog inputs to the MCU. Note that this calibration routine is a downloaded program that does not take up space in the normal application program.

Bootstrap Mode Logic

In the M68HC11 MCUs, very little logic is dedicated to the bootstrap mode. Consequently, this mode adds almost no extra cost to the MCU system. The biggest piece of circuitry for bootstrap mode is the small boot ROM. This ROM is 192 bytes in the original MC68HC11A8, but some of the newest members of the M68HC11 Family, such as the MC68HC711K4, have as much as 448 bytes to accommodate added features. Normally, this boot ROM is present in the memory map only when the MCU is reset in bootstrap mode to prevent interference with the user's normal memory space. The enable for this ROM is controlled by the read boot ROM (RBOOT) control bit in the highest priority interrupt (HPRIO) register. The RBOOT bit can be written by software whenever the MCU is in special test or special bootstrap modes; when the MCU is in normal modes, RBOOT reverts to 0 and becomes a read-only bit. All other logic in the MCU would be present whether or not there was a bootstrap mode.

Figure 1 shows the composite memory map of the MC68HC711E9 in its four basic modes of operation, including bootstrap mode. The active mode is determined by the mode A (MDA) and special mode (SMOD) control bits in the HPRIO control register. These control bits are in turn controlled by the state of the mode A (MODA) and mode B (MODB) pins during reset. Table 1 shows the relationship between the state of these pins during reset, the selected mode, and the state of the MDA, SMOD, and RBOOT control bits. Refer to the composite memory map and information in Table 1 for the following discussion.

The MDA control bit is determined by the state of the MODA pin as the MCU leaves reset. MDA selects between single-chip and expanded operating modes. When MDA is 0, a single-chip mode is selected, either normal single-chip mode or special bootstrap mode. When MDA is 1, an expanded mode is selected, either normal expanded mode or special test mode.

The SMOD control bit is determined by the inverted state of the MODB pin as the MCU leaves reset. SMOD controls whether a normal mode or a special mode is selected. When SMOD is 0, one of the two normal modes is selected, either normal single-chip mode or normal expanded mode. When SMOD is 1, one of the two special modes is selected, either special bootstrap mode or special test mode. When either special mode is in effect (SMOD = 1), certain privileges are in effect, for instance, the ability to write to the mode control bits and fetching the reset and interrupt vectors from $BFxx rather than $FFxx.

Table 1. Mode Selection Summary

Input Pins		Mode Selected	Control Bits in HPRIO		
MODB	MODA		RBOOT	SMOD	MDA
1	0	Normal single chip	0	0	0
0	0	Normal expanded	0	0	1
0	0	Special bootstrap	1	1	0
0	1	Special test	0	1	1

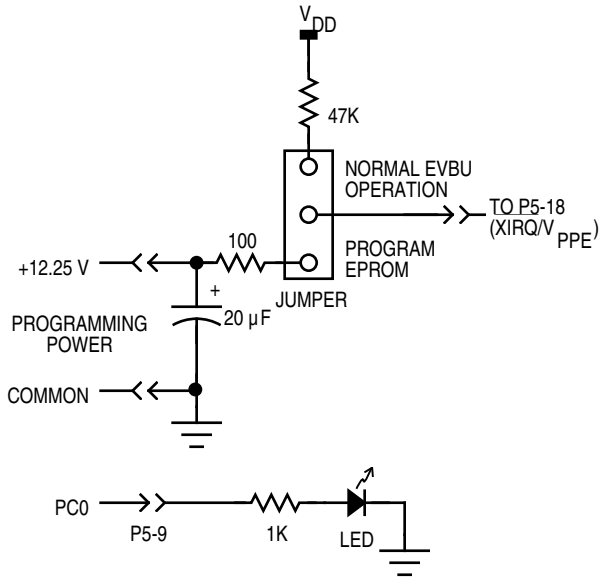


Figure 8. PC-to-MCU Programming Circuit

Lines 50–95 read in the small bootloader from DATA statements at the end of the listing. The source code for this bootloader is presented in the DATA statements. The bootloaded code makes port C bit 0 low, initializes the X and Y registers for use by the EPROM programming utility routine contained in the boot ROM, and then jumps to that routine. The hexadecimal values read in from the DATA statements are converted to binary values by a subroutine. The binary values are then saved as one string (BOOTCODE$).

The next long section of code (lines 97–1250) reads in the S records from an external disk file (in this case, BUF34.S19), converts them to integer, and saves them in an array. The techniques used in this section show how to convert ASCII S records to binary form that can be sent (bootloaded) to an M68HC11.

This S-record translator only looks for the S1 records that contain the actual object code. All other S-record types are ignored.

When an S1 record is found (lines 1000–1024), the next two characters form the hex byte giving the number of hex bytes to follow. This byte is converted to integer by the same subroutine that converted the bootloaded code from the DATA statements. This BYTECOUNT is adjusted by subtracting 3, which accounts for the address and checksum bytes and leaves just the number of object-code bytes in the record.

Starting at line 1100, the 2-byte (4-character) starting address is converted to decimal. This address is the starting address for the object code bytes to follow. An index into the CODE% array is formed by subtracting the base address initialized at the start of the program from the starting address for this S record.

A FOR-NEXT loop starting at line 1130 converts the object code bytes to decimal and saves them in the CODE% array. When all the object code bytes have been converted from the current S record, the program loops back to find the next S1 record.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

M68HC11E
Rev. 5.1, 07/2005

freescale™
semiconductor