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	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°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=mc68hc11e1vfne3r

Chapter 5
Resets and Interrupts

Chapter 9
Timing Systems

Chapter 10
Electrical Characteristics

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$1025	Timer Interrupt Flag 2 (TFLG2) See page 142.	Read:	TOF	RTIF	PAOVF	PAIF				
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$1026	Pulse Accumulator Control Register (PACTL) See page 142.	Read:	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$1027	Pulse Accumulator Count Register (PACNT) See page 146.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1028	Serial Peripheral Control Register (SPCR) See page 123.	Read:	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	0	1	U	U
$1029	Serial Peripheral Status Register (SPSR) See page 124.	Read:	SPIF	WCOL		MODF				
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$102A	Serial Peripheral Data I/O Register (SPDR) See page 125.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$102B	Baud Rate Register (BAUD) See page 113.	Read:	TCLR	SCP2(1)	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	U	U	U
$102C	Serial Communications Control Register 1 (SCCR1) See page 110.	Read:	R8	T8		M	WAKE			
		Write:								
		Reset:	I	I	0	0	0	0	0	0
$102D	Serial Communications Control Register 2 (SCCR2) See page 111.	Read:	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$102E	Serial Communications Status Register (SCSR) See page 112.	Read:	TDRE	TC	RDRF	IDLE	OR	NF	FE	
		Write:								
		Reset:	1	1	0	0	0	0	0	0

1. SCP2 adds ÷39 to SCI prescaler and is present only in MC68HC(7)11E20.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$102F	Serial Communications Data Register (SCDR) See page 110.	Read:	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
		Write:								
		Reset:	Indeterminate after reset							
$1030	Analog-to-Digital Control Status Register (ADCTL) See page 62.	Read:	CCF		SCAN	MULT	CD	CC	CB	CA
		Write:								
		Reset:	0	0	Indeterminate after reset					

= Unimplemented R = Reserved U = Unaffected

I = Indeterminate after reset

Figure 2-7. Register and Control Bit Assignments (Sheet 4 of 6)

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

NOSEC — Security Disable Bit

NOSEC is invalid unless the security mask option is specified before the MCU is manufactured. If the security mask option is omitted NOSEC always reads 1. The enhanced security feature is available in the MC68S711E9 MCU. The enhancement to the standard security feature protects the EPROM as well as RAM and EEPROM.

0 = Security enabled
1 = Security disabled

NOCOP — COP System Disable Bit

Refer to Chapter 5 Resets and Interrupts.

1 = COP disabled
0 = COP enabled

ROMON — ROM/EPROM/OTPROM Enable Bit

When this bit is 0, the ROM or EPROM is disabled and that memory space becomes externally addressed. In single-chip mode, ROMON is forced to 1 to enable ROM/EPROM regardless of the state of the ROMON bit.

0 = ROM disabled from the memory map
1 = ROM present in the memory map

EEON — EEPROM Enable Bit

When this bit is 0, the EEPROM is disabled and that memory space becomes externally addressed.

0 = EEPROM removed from the memory map
1 = EEPROM present in the memory map

2.3.3.2 RAM and I/O Mapping Register

The internal registers used to control the operation of the MCU can be relocated on 4-Kbyte boundaries within the memory space with the use of the RAM and I/O mapping register (INIT). This 8-bit special-purpose register can change the default locations of the RAM and control registers within the MCU memory map. It can be written only once within the first 64 E-clock cycles after a reset in normal modes, and then it becomes a read-only register.

Address: $103D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0
Reset:	0	0	0	0	0	0	0	1

Figure 2-12. RAM and I/O Mapping Register (INIT)

RAM[3:0] — RAM Map Position Bits

These four bits, which specify the upper hexadecimal digit of the RAM address, control position of RAM in the memory map. RAM can be positioned at the beginning of any 4-Kbyte page in the memory map. It is initialized to address $0000 out of reset. Refer to Table 2-4.

REG[3:0] — 64-Byte Register Block Position

These four bits specify the upper hexadecimal digit of the address for the 64-byte block of internal registers. The register block, positioned at the beginning of any 4-Kbyte page in the memory map, is initialized to address $1000 out of reset. Refer to Table 2-5.

2.5.1.5 EEPROM Byte Erase

This is an example of how to erase a single byte of EEPROM.

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

2.5.1.6 CONFIG Register Programming

Because the CONFIG register is implemented with EEPROM cells, use EEPROM procedures to erase and program this register. The procedure for programming is the same as for programming a byte in the EEPROM array, except that the CONFIG register address is used. CONFIG can be programmed or erased (including byte erase) while the MCU is operating in any mode, provided that PTCON in BPROT is clear.

To change the value in the CONFIG register, complete this procedure.

1. Erase the CONFIG register.
2. Program the new value to the CONFIG address.
3. Initiate reset.

NOTE

Do not initiate a reset until the procedure is complete.

2.5.2 EEPROM Security

The optional security feature, available only on ROM-based MCUs, protects the EEPROM and RAM contents from unauthorized access. A program, or a key portion of a program, can be protected against unauthorized duplication. To accomplish this, the protection mechanism restricts operation of protected devices to the single-chip modes. This prevents the memory locations from being monitored externally because single-chip modes do not allow visibility of the internal address and data buses. Resident programs, however, have unlimited access to the internal EEPROM and RAM and can read, write, or transfer the contents of these memories.

An enhanced security feature which protects EPROM contents, RAM, and EEPROM from unauthorized accesses is available in MC68S711E9. Refer to Chapter 11 Ordering Information and Mechanical Specifications for the exact part number.

For further information, these engineering bulletins have been included at the back of this data book:

- EB183 — Enabling the Security Feature on the MC68HC711E9 Devices with PCbug11 on the M68HC711E9PGMR
- EB188 — Enabling the Security Feature on M68HC811E2 Devices with PCbug11 on the M68HC711E9PGMR

3.2.3 Digital Control

All A/D converter operations are controlled by bits in register ADCTL. In addition to selecting the analog input to be converted, ADCTL bits indicate conversion status and control whether single or continuous conversions are performed. Finally, the ADCTL bits determine whether conversions are performed on single or multiple channels.

3.2.4 Result Registers

Four 8-bit registers ADR[4:1] store conversion results. Each of these registers can be accessed by the processor in the CPU. The conversion complete flag (CCF) indicates when valid data is present in the result registers. The result registers are written during a portion of the system clock cycle when reads do not occur, so there is no conflict.

3.2.5 A/D Converter Clocks

The CSEL bit in the OPTION register selects whether the A/D converter uses the system E clock or an internal RC oscillator for synchronization. When E-clock frequency is below 750 kHz, charge leakage in the capacitor array can cause errors, and the internal oscillator should be used. When the RC clock is used, additional errors can occur because the comparator is sensitive to the additional system clock noise.

3.2.6 Conversion Sequence

A/D converter operations are performed in sequences of four conversions each. A conversion sequence can repeat continuously or stop after one iteration. The conversion complete flag (CCF) is set after the fourth conversion in a sequence to show the availability of data in the result registers. Figure 3-3 shows the timing of a typical sequence. Synchronization is referenced to the system E clock.

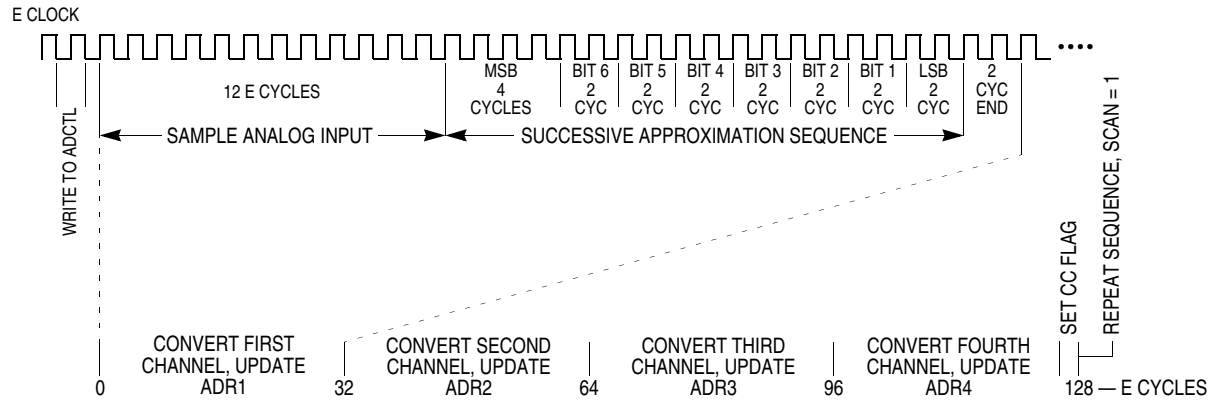


Figure 3-3. A/D Conversion Sequence

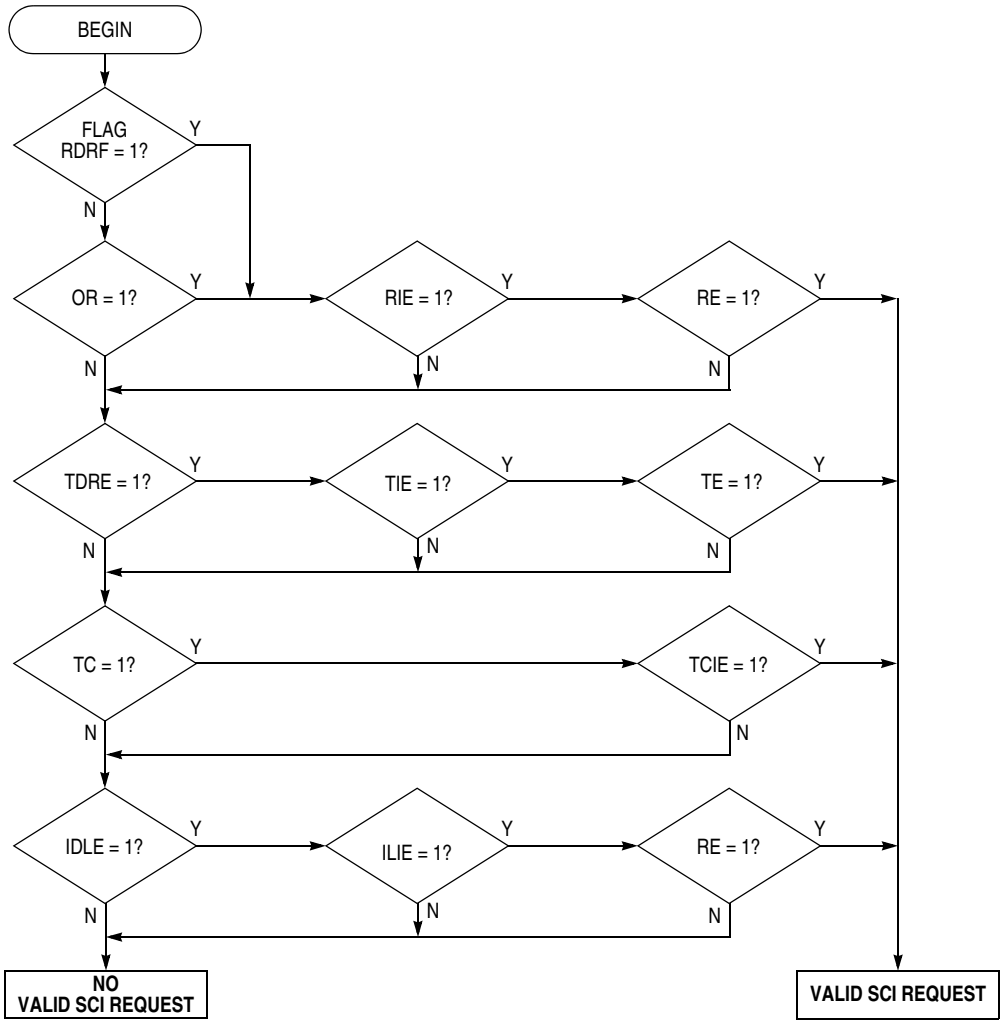


Figure 5-7. Interrupt Source Resolution Within SCI

5.6.2 Stop Mode

Executing the STOP instruction while the S bit in the CCR is equal to 0 places the MCU in stop mode. If the S bit is not 0, the stop opcode is treated as a no-op (NOP). Stop mode offers minimum power consumption because all clocks, including the crystal oscillator, are stopped while in this mode. To exit stop and resume normal processing, a logic low level must be applied to one of the external interrupts ($\overline{\text{IRQ}}$ or $\overline{\text{XIRQ}}$) or to the $\overline{\text{RESET}}$ pin. A pending edge-triggered $\overline{\text{IRQ}}$ can also bring the CPU out of stop.

Because all clocks are stopped in this mode, all internal peripheral functions also stop. The data in the internal RAM is retained as long as V_{DD} power is maintained. The CPU state and I/O pin levels are static and are unchanged by stop. Therefore, when an interrupt comes to restart the system, the MCU resumes processing as if there were no interruption. If reset is used to restart the system, a normal reset sequence results in which all I/O pins and functions are also restored to their initial states.

To use the $\overline{\text{IRQ}}$ pin as a means of recovering from stop, the I bit in the CCR must be clear ($\overline{\text{IRQ}}$ not masked). The $\overline{\text{XIRQ}}$ pin can be used to wake up the MCU from stop regardless of the state of the X bit in the CCR, although the recovery sequence depends on the state of the X bit. If X is set to 0 ($\overline{\text{XIRQ}}$ not

6.3 Port B

In single-chip or bootstrap modes, port B pins are general-purpose outputs. In expanded or special test modes, port B pins are high-order address outputs.

Address: $1004

	Bit 7	6	5	4	3	2	1	Bit 0
Single-chip or bootstrap modes:								
Read: Write:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Reset:	0	0	0	0	0	0	0	0
Expanded or special test modes:								
Read: Write:	ADDR15	ADDR14	ADDR13	ADDR12	ADDR11	ADDR10	ADDR9	ADDR8
Reset:	0	0	0	0	0	0	0	0

Figure 6-3. Port B Data Register (PORTB)

6.4 Port C

In single-chip and bootstrap modes, port C pins reset to high-impedance inputs. (DDRC bits are set to 0.) In expanded and special test modes, port C pins are multiplexed address/data bus and the port C register address is treated as an external memory location.

Address: $1003

	Bit 7	6	5	4	3	2	1	Bit 0
Single-chip or bootstrap modes:								
Read: Write:	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
Reset:	Indeterminate after reset							
Expanded or special test modes:								
Read: Write:	ADDR7 DATA7	ADDR6 DATA6	ADDR5 DATA5	ADDR4 DATA4	ADDR3 DATA3	ADDR2 DATA2	ADDR1 DATA1	ADDR0 DATA0
Reset:	Indeterminate after reset							

Figure 6-4. Port C Data Register (PORTC)

Address: $1005

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PCL7	PCL6	PCL5	PCL4	PCL3	PCL2	PCL1	PCL0
Reset:	Indeterminate after reset							

Figure 6-5. Port C Latched Register (PORTCL)

Bit 5 — Unimplemented

Always reads 0

MODF — Mode Fault Bit

To clear the MODF bit, read the SPSR (with MODF set), then write to the SPCR. Refer to 8.5.4 Slave Select and 8.6 SPI System Errors.

0 = No mode fault
1 = Mode fault

Bits [3:0] — Unimplemented

Always read 0

8.7.3 Serial Peripheral Data I/O 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.

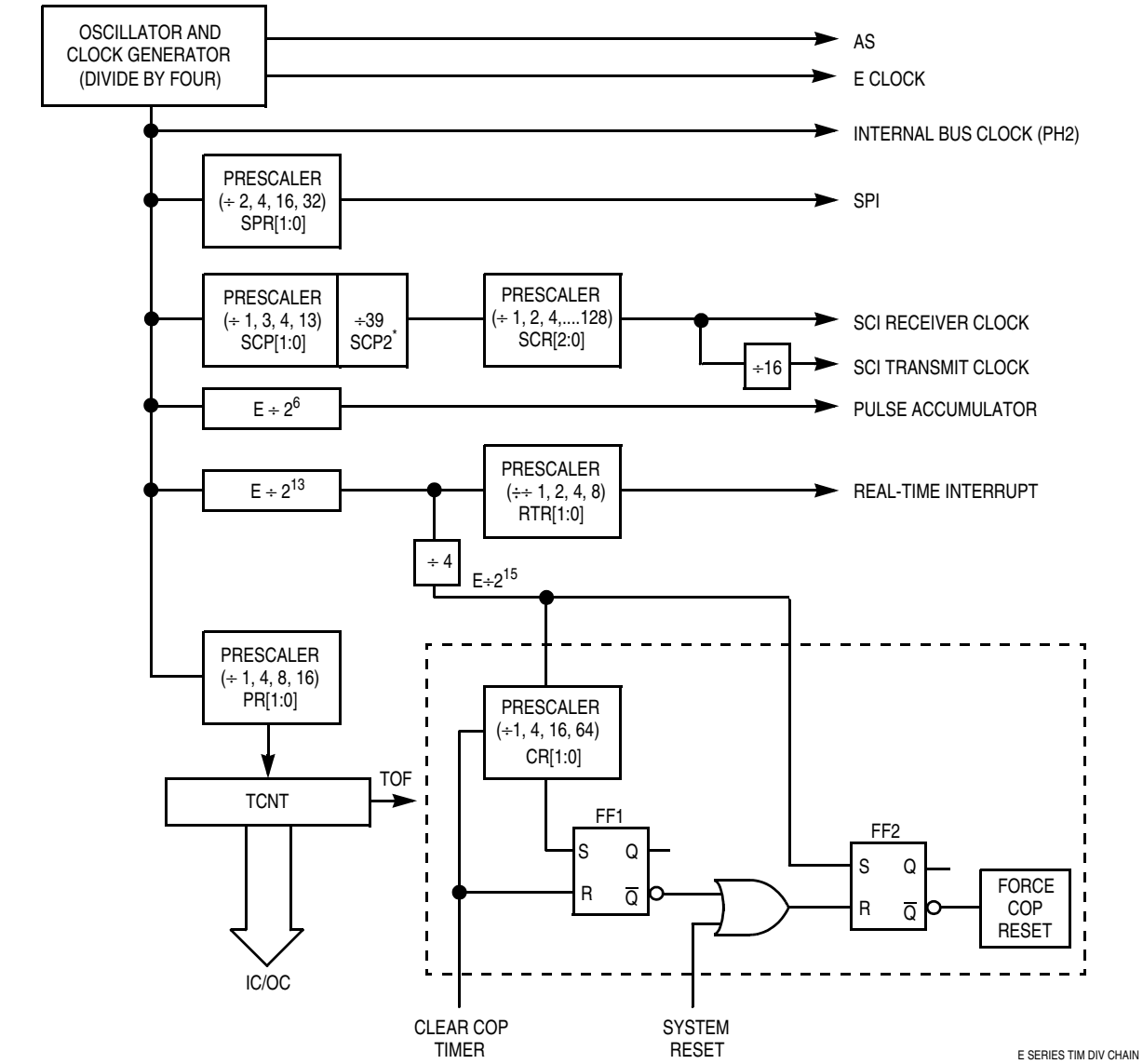
A read of the SPDR is actually a read of a buffer. To prevent an overrun and the loss of the byte that caused the overrun, the first SPIF must be cleared by the time a second transfer of data from the shift register to the read buffer is initiated.

Address: $102A

	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:	Indeterminate after reset							

Figure 8-5. Serial Peripheral Data I/O Register (SPDR)

SPI is double buffered in and single buffered out.



* SCP2 present on MC68HC(7)11E20 only

Figure 9-1. Timer Clock Divider Chains

10.9 Control Timing

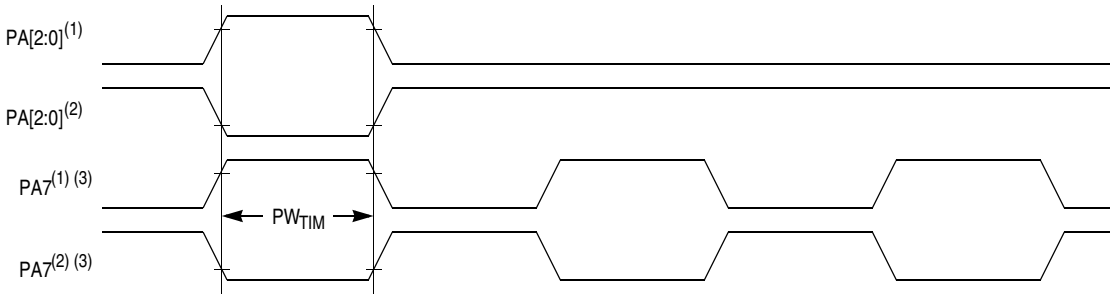
Characteristic[1] [2]	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of operation	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	333	—	ns
Crystal frequency	f_{XTAL}	—	4.0	—	8.0	—	12.0	MHz
External oscillator frequency	4 f_o	dc	4.0	dc	8.0	dc	12.0	MHz
Processor control setup time t_{PCSU} = 1/4 t_{CYC}+ 50 ns	t_{PCSU}	300	—	175	—	133	—	ns
Reset input pulse width To guarantee external reset vector Minimum input time (can be pre-empted by internal reset)	PW_{RSTL}	8 1	— —	8 1	— —	8 1	— —	t_{CYC}
Mode programming setup time	t_{MPS}	2	—	2	—	2	—	t_{CYC}
Mode programming hold time	t_{MPH}	10	—	10	—	10	—	ns
Interrupt pulse width, $\overline{IRQ}$ edge-sensitive mode PW_{IRQ} = t_{CYC} + 20 ns	PW_{IRQ}	1020	—	520	—	353	—	ns
Wait recovery startup time	t_{WRS}	—	4	—	4	—	4	t_{CYC}
Timer pulse width input capture pulse accumulator input PW_{TIM} = t_{CYC} + 20 ns	PW_{TIM}	1020	—	520	—	353	—	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. $\overline{RESET}$ is recognized during the first clock cycle it is held low. Internal circuitry then drives the pin low for four clock cycles, releases the pin, and samples the pin level two cycles later to determine the source of the interrupt. Refer to Chapter 5 Resets and Interrupts for further detail.

10.10 MC68L11E9/E20 Control Timing

Characteristic(1) (2)	Symbol	1.0 MHz		2.0 MHz		Unit
		Min	Max	Min	Max	
Frequency of operation	f_o	dc	1.0	dc	2.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	ns
Crystal frequency	f_{XTAL}	—	4.0	—	8.0	MHz
External oscillator frequency	4 f_o	dc	4.0	dc	8.0	MHz
Processor control setup time t_{PCSU} = 1/4 t_{CYC}+ 75 ns	t_{PCSU}	325	—	200	—	ns
Reset input pulse width To guarantee external reset vector Minimum input time (can be pre-empted by internal reset)	PW_{RSTL}	8 1	— —	8 1	— —	t_{CYC}
Mode programming setup time	t_{MPS}	2	—	2	—	t_{CYC}
Mode programming hold time	t_{MPH}	10	—	10	—	ns
Interrupt pulse width, $\overline{IRQ}$ edge-sensitive mode PW_{IRQ} = t_{CYC} + 20 ns	PW_{IRQ}	1020	—	520	—	ns
Wait recovery startup time	t_{WRS}	—	4	—	4	t_{CYC}
Timer pulse width input capture pulse accumulator input PW_{TIM} = t_{CYC} + 20 ns	PW_{TIM}	1020	—	520	—	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. $\overline{RESET}$ is recognized during the first clock cycle it is held low. Internal circuitry then drives the pin low for four clock cycles, releases the pin, and samples the pin level two cycles later to determine the source of the interrupt. Refer to Chapter 5 Resets and Interrupts for further detail.



Notes:
1. Rising edge sensitive input
2. Falling edge sensitive input
3. Maximum pulse accumulator clocking rate is E-clock frequency divided by 2.

Figure 10-2. Timer Inputs

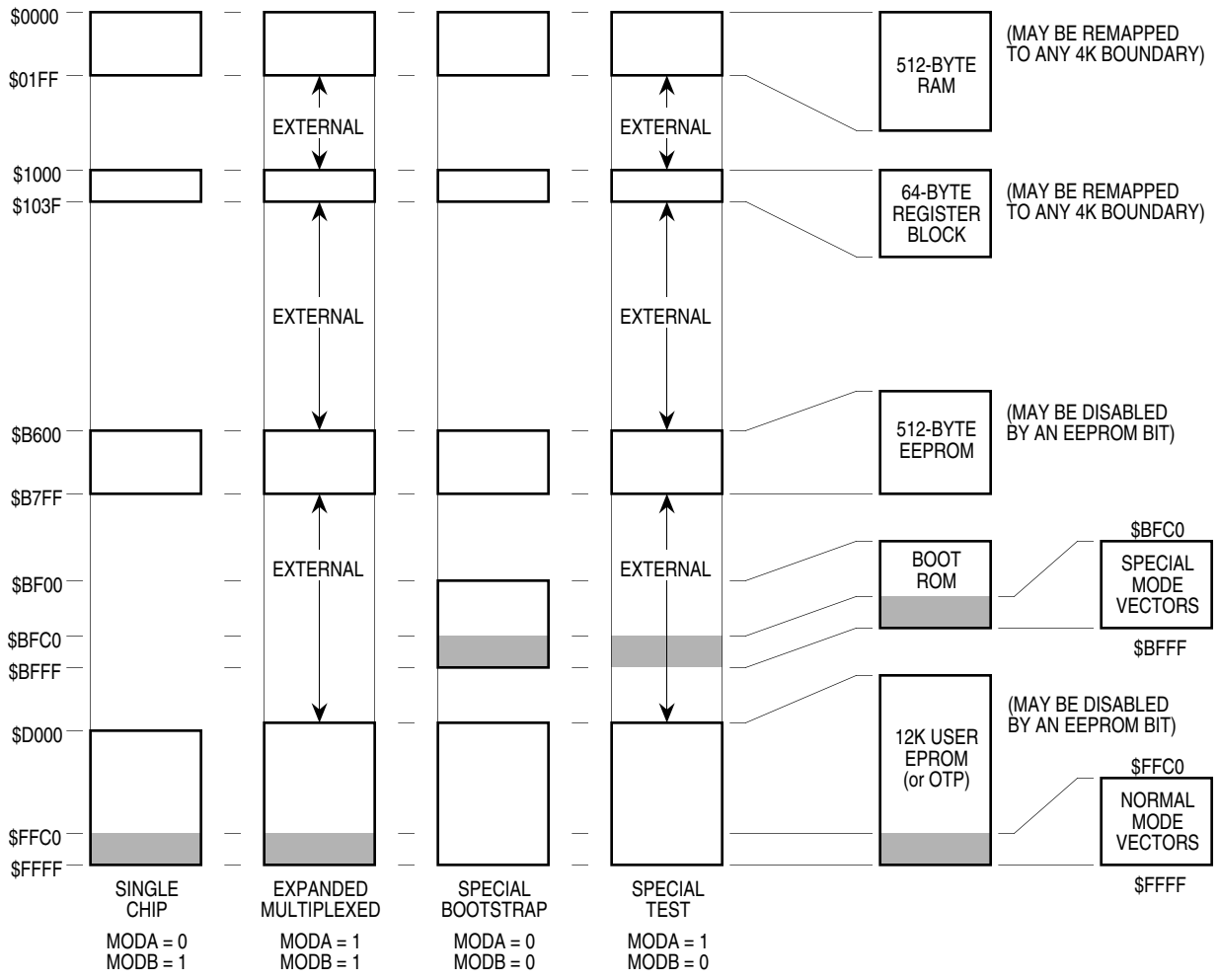
Description	CONFIG	Temperature	Frequency	MC Order Number
52-pin windowed ceramic leaded chip carrier (CLCC)				
EPROM	$0F	–40°C to +85°C	2 MHz	MC68HC711E9CFS2
			3 MHz	MC68HC711E9CFS3
		–40°C to +105°C	2 MHz	MC68HC711E9VFS2
		–40°C to +125°C	2 MHz	MC68HC711E9VFS2
20 Kbytes EPROM	$0F	0°C o +70°°C	3 MHz	MC68HC711E20FS3
		–40°C to +85°C	2 MHz	MC68HC711E20CFS2
			3 MHz	MC68HC711E20CFS3
		–40°C to +105°C	2 MHz	MC68HC711E20VFS2
		–40°C to +125°C	2 MHz	MC68HC711E20MFS2
48-pin dual in-line package (DIP) — MC68HC811E2 only				
No ROM, 2 Kbytes EEPROM	$FF	0°C to +70°°C	2 MHz	MC68HC811E2P2
		–40°C to +85°C	2 MHz	MC68HC811E2CP2
		–40°C to +105°C	2 MHz	MC68HC811E2VP2
		–40°C to +125°C	2 MHz	MC68HC811E2MP2
56-pin dual in-line package with 0.70-inch lead spacing (SDIP)				
BUFFALO ROM	$0F	–40°C to +85°C	2 MHz	MC68HC11E9BCB2
			3 MHz	MC68HC11E9BCB3
No ROM	$0D	–40°C to +85°C	2 MHz	MC68HC11E1CB2
			3 MHz	MC68HC11E1CB3
		–40°C to +105°C	2 MHz	MC68HC11E1VB2
		–40°C to +125°C	2 MHz	MC68HC11E1MB2
No ROM, no EEPROM	$0C	–40°C to +85°C	2 MHz	MC68HC11E0CB2
			3 MHz	MC68HC11E0CB3
		–40°C to +105°C	2 MHz	MC68HC11E0VB2
		–40°C to +125°C	2 MHz	MC68HC11E0MB2

11.3 Custom ROM Device Ordering Information

Description	Temperature	Frequency	MC Order Number
52-pin plastic leaded chip carrier (PLCC)			
Custom ROM	0°C to +70°°C	3 MHz	MC68HC11E9FN3
	–40°C to +85°C	2 MHz	MC68HC11E9CFN2
		3 MHz	MC68HC11E9CFN3
	–40°C to +105°C	2 MHz	MC68HC11E9VFN2
	–40°C to +125°C	2 MHz	MC68HC11E9MFN2

Figure 2 shows how the bootloader program differentiates between the default baud rate (7812 baud at a 2-MHz E-clock rate) and the alternate baud rate (1200 baud at a 2-MHz E-clock rate). The host computer sends an initial $FF character, which is used by the bootloader to determine the baud rate that will be used for the downloading operation. The top half of Figure 2 shows normal reception of $FF. Receive data samples at [1] detect the falling edge of the start bit and then verify the start bit by taking a sample at the center of the start bit time. Samples are then taken at the middle of each bit time [2] to reconstruct the value of the received character (all 1s in this case). A sample is then taken at the middle of the stop bit time as a framing check (a 1 is expected) [3]. Unless another character immediately follows this $FF character, the receive data line will idle in the high state as shown at [4].

The bottom half of Figure 2 shows how the receiver will incorrectly receive the $FF character that is sent from the host at 1200 baud. Because the receiver is set to 7812 baud, the receive data samples are taken at the same times as in the upper half of Figure 2. The start bit at 1200 baud [5] is 6.5 times as long as the start bit at 7812 baud [6].



NOTE: Software can change some aspects of the memory map after reset.

Figure 1. MC68HC711E9 Composite Memory Map

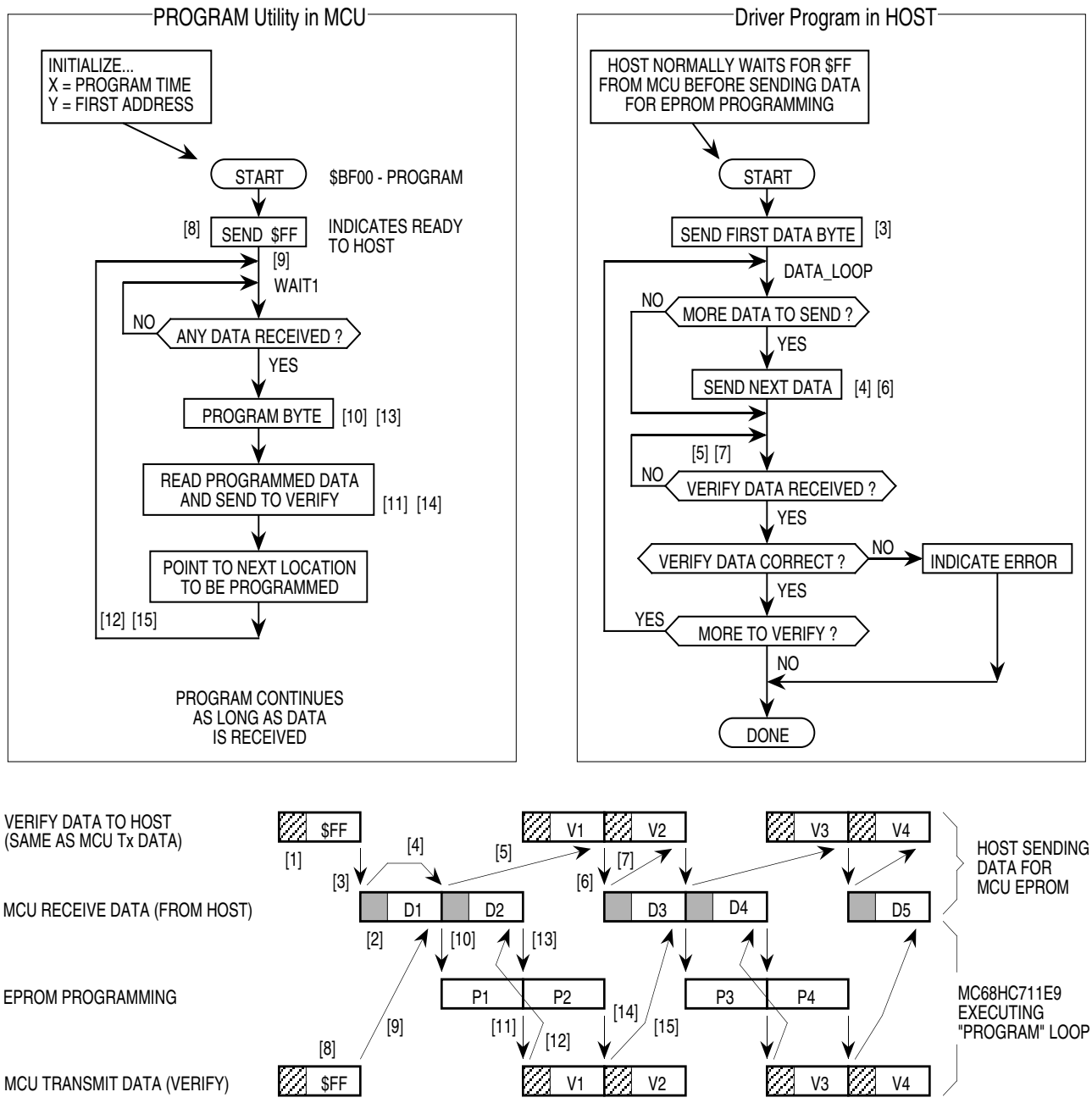


Figure 4. Host and MCU Activity during EPROM PROGRAM Utility

```
162 BF62 E72D              STAB   SCCR2,X        Rx and Tx Enabled
163 BF64 CC021B            LDD    #DELAYF        Delay for fast baud rate
164 BF67 ED16              STD    TOC1,X         Set as default delay
165
166                * Send BREAK to signal ready for download
167 BF69 1C2D01            BSET   SCCR2,X $01    Set send break bit
168 BF6C 1E0801FC          BRSET  PORTD,X $01 *  Wait for RxD pin to go low
169 BF70 1D2D01            BCLR   SCCR2,X $01    Clear send break bit
170 BF73
171 BF73 1F2E20FC          BRCLR  SCSR,X $20 *   Wait for RDRF
172 BF77 A62F              LDAA   SCDAT,X        Read data
173                * Data will be $00 if BREAK OR $00 received
174 BF79 2603              BNE    NOTZERO        Bypass JMP if not 0
175 BF7B 7EB600            JMP    EEPMSTR        Jump to EEPROM if it was 0
176 BF7E           NOTZERO EQU    *
177 BF7E 81FF              CMPA   #$FF           $FF will be seen as $FF
178 BF80 2708              BEQ    BAUDOK         If baud was correct
179                * Or else change to ÷104 (÷13 & ÷8) 1200 @ 2MHZ
180 BF82 1C2B33            BSET   BAUD,X $33     Works because $22 -> $33
181 BF85 CC0DB0            LDD    #DELAYS        And switch to slower...
182 BF88 ED16              STD    TOC1,X         delay constant
183 BF8A           BAUDOK  EQU    *
184 BF8A 18CE0000          LDY    #RAMSTR        Point at start of RAM
185
186 BF8E           WAIT    EQU    *
187 BF8E EC16              LDD    TOC1,X         Move delay constant to D
188 BF90           WTLOOP  EQU    *
189 BF90 1E2E2007          BRSET  SCSR,X $20 NEWONE  Exit loop if RDRF set
190 BF94 8F                XGDX                  Swap delay count to X
191 BF95 09                DEX                   Decrement count
192 BF96 8F                XGDX                  Swap back to D
193 BF97 26F7              BNE    WTLOOP         Loop if not timed out
194 BF99 200F              BRA    STAR           Quit download on timeout
195
196 BF9B           NEWONE  EQU    *
197 BF9B A62F              LDAA   SCDAT,X        Get received data
198 BF9D 18A700            STAA   $00,Y          Store to next RAM location
199 BFA0 A72F              STAA   SCDAT,X        Transmit it for handshake
200 BFA2 1808              INY                   Point at next RAM location
201 BFA4 188C0200          CPY    #RAMEND+1      See if past end
202 BFA8 26E4              BNE    WAIT           If not, Get another
203
204 BFAA           STAR    EQU    *
205 BFAA CE1068            LDX    #PROGDEL       Init X with programming delay
206 BFAD 18CED000          LDY    #EPRMSTR       Init Y with EPROM start addr
207 BFB1 7E0000            JMP    RAMSTR         ** EXIT to start of RAM **
208 BFB4
209                ************************************************
210                * Block fill unused bytes with zeros
211
212 BFB4 000000000000        BSZ    $BFD1-*
       000000000000
       000000000000
       000000000000
       0000000000
```

To Execute the Program

Once you have obtained PCbug11, use this step-by-step procedure.

Step 1

- Before applying power to the programming board, connect the M68HC711E9PGMR serial port P2 to one of your PC COM ports with a standard 25 pin RS-232 cable. Do not use a null modem cable or adapter which swaps the transmit and receive signals between the connectors at each end of the cable.
- Place your MC68HC811E2 part in the PLCC socket on your board.
- Insert the part upside down with the notched corner pointing toward the red power LED.
- Make sure both S1 and S2 switches are turned off.
- Apply +5 volts to +5 volts and ground to GND on the programmer board's power connector, P1. Applying voltage to the V_{PP} pin is not necessary.

Step 2

Apply power to the programmer board by moving the +5-volt switch to the ON position.

From a DOS command line prompt, start PCbug11 this way:

- C:\PCBUG11\> PCBUG11 –A PORT = 1 when the E9PGMR connected to COM1 or
- C:\PCBUG11\> PCBUG11 –A PORT = 2 when the E9PGMR connected to COM2

PCbug11only supports COM ports 1 and 2.

Step 3

PCbug11 defaults to base ten for its input parameters.

Change this to hexadecimal by typing: CONTROL BASE HEX

Step 4

Clear the block protect register (BPROT) to allow programming of the MC68HC811E2 EEPROM.

At the PCbug11 command prompt, type: MS 1035 00

Step 5

PCbug11 defaults to a 512-byte EEPROM array located at $B600. This must be changed since the EEPROM is, by default, located at $F800 on the MC68HC811E2.

At the PCbug11 command prompt, type: EEPROM 0

Then type: EEPROM F800 FFFF
EEPROM 103F 103F

This assumes you have not relocated the EEPROM by previously reprogramming the upper 4 bits of the CONFIG register. But if you have done this and your S records reside in an address range other than $F800 to $FFFF, you will need to first relocate the EEPROM.

Programming Procedure

Once you have obtained PCbug11, use this step-by-step procedure to program your MC68HC711E9 part.

Step 1

- Before applying power to the EVBU, remove the jumper from J7 and place it across J3 to ground the MODB pin.
- Place a jumper across J4 to ground the MODA pin. This will force the EVBU into special bootstrap mode on power up.
- Remove the resident MC68HC11E9 MCU from the EVBU.
- Place your MC68HC711E9 in the open socket with the notched corner of the part aligned with the notch on the PLCC socket.
- Connect the EVBU to one of your PC COM ports. Apply +5 volts to V_{DD} and ground to GND on the power connector of your EVBU.

Also take note of P4 connector pin 18. In step 5, you will connect a +12-volt (at most +12.5 volts) programming voltage through a 100-Ω current limiting resistor to the XIRQ pin. Do not connect this programming voltage until you are instructed to do so in step 5.

Step 2

- From a DOS command line prompt, start PCbug11 with
  - C:\PCBUG11\> PCBUG11 –E PORT = 1 with the EVBU connected to COM1
  - C:\PCBUG11\> PCBUG11 –E PORT = 2 with the EVBU connected to COM2

PCbug11 only supports COM ports 1 and 2. If you have made the proper connections and have a high quality cable, you should quickly get a PCbug11 command prompt. If you do receive a Comms fault error, check your cable and board connections. Most PCbug11 communications problems can be traced to poorly made cables or bad board connections.

Step 3

- PCbug11 defaults to base 10 for its input parameters; change this to hexadecimal by typing

CONTROL BASE HEX

Step 4

- You must declare the addresses of the EPROM array to PCbug11. To do this, type:
  EPROM D000 FFFF

Step 5

You are now ready to download your program into the EPROM.

- Connect +12 volts (at most +12.5 volts) through a 100-Ω current limiting resistor to P4 connector pin 18, the XIRQ* pin.
- At the PCbug11 command prompt type: LOADS C:\MYPROG\ISHERE.S19

Substitute the name of your program into the command above. Use a full path name if your program is not located in the same directory as PCbug11.

Step 6

After the programming operation is complete, PCbug11 will display this message

Total bytes loaded: $xxxx

Total bytes programmed: $yyyy

- You should now remove the programming voltage from P4 connector pin 18, the XIRQ* pin.
- Each ORG directive in your assembly language source will cause a pair of these lines to be generated. For this operation, $yyyy will be incremented by the size of each block of code programmed into the EPROM of the MC68HC711E9.
- PCbug11 will display the above message whether or not the programming operation was successful. As a precaution, you should have PCbug11 verify your code.
- At the PCbug11 command prompt type: VERF C:\MYPROG\ISHERE.S19

Substitute the name of your program into the command above. Use a full path name if your program is not located in the same directory as PCbug11.

If the verify operation fails, a list of addresses which did not program correctly is displayed. Should this occur, you probably need to erase your part more completely. To do so, allow the MC68HC711E9 to sit for at least 45 minutes under an ultraviolet light source. Attempt the programming operation again. If you have purchased devices in plastic packages (one-time programmable parts), you will need to try again with a new, unprogrammed device.