

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	2MHz
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)	4.5V ~ 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 (19.1x19.1)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68p11e1cfne2r

PGM — EPROM Programming Voltage Enable Bit

PGM can be read any time and can be written only when ELAT = 1.

0 = Programming voltage to EPROM array disconnected

1 = Programming voltage to EPROM array connected

2.5 EEPROM

Some E-series devices contain 512 bytes of on-chip EEPROM. The MC68HC811E2 contains 2048 bytes of EEPROM with selectable base address. All E-series devices contain the EEPROM-based CONFIG register.

2.5.1 EEPROM and CONFIG Programming and Erasure

The erased state of an EEPROM bit is 1. During a read operation, bit lines are precharged to 1. The floating gate devices of programmed bits conduct and pull the bit lines to 0. Unprogrammed bits remain at the precharged level and are read as ones. Programming a bit to 1 causes no change. Programming a bit to 0 changes the bit so that subsequent reads return 0.

When appropriate bits in the BPROT register are cleared, the PPROG register controls programming and erasing the EEPROM. The PPROG register can be read or written at any time, but logic enforces defined programming and erasing sequences to prevent unintentional changes to EEPROM data. When the EELAT bit in the PPROG register is cleared, the EEPROM can be read as if it were a ROM.

The on-chip charge pump that generates the EEPROM programming voltage from V_{DD} uses MOS capacitors, which are relatively small in value. The efficiency of this charge pump and its drive capability are affected by the level of V_{DD} and the frequency of the driving clock. The load depends on the number of bits being programmed or erased and capacitances in the EEPROM array.

The clock source driving the charge pump is software selectable. When the clock select (CSEL) bit in the OPTION register is 0, the E clock is used; when CSEL is 1, an on-chip resistor-capacitor (RC) oscillator is used.

The EEPROM programming voltage power supply voltage to the EEPROM array is not enabled until there has been a write to PPROG with EELAT set and PGM cleared. This must be followed by a write to a valid EEPROM location or to the CONFIG address, and then a write to PPROG with both the EELAT and EPGM bits set. Any attempt to set both EELAT and EPGM during the same write operation results in neither bit being set.

2.5.1.1 Block Protect Register

This register prevents inadvertent writes to both the CONFIG register and EEPROM. The active bits in this register are initialized to 1 out of reset and can be cleared only during the first 64 E-clock cycles after reset in the normal modes. When these bits are cleared, the associated EEPROM section and the CONFIG register can be programmed or erased. EEPROM is only visible if the EEON bit in the CONFIG register is set. The bits in the BPROT register can be written to 1 at any time to protect EEPROM and the CONFIG register. In test or bootstrap modes, write protection is inhibited and BPROT can be written repeatedly. Address ranges for protected areas of EEPROM differ significantly for the MC68HC811E2. Refer to [Figure 2-16](#).

3.3 A/D Converter Power-Up and Clock Select

Bit 7 of the OPTION register controls A/D converter power-up. Clearing ADPU removes power from and disables the A/D converter system. Setting ADPU enables the A/D converter system. Stabilization of the analog bias voltages requires a delay of as much as 100 μ s after turning on the A/D converter. When the A/D converter system is operating with the MCU E clock, all switching and comparator operations are inherently synchronized to the main MCU clocks. This allows the comparator output to be sampled at relatively quiet times during MCU clock cycles. Since the internal RC oscillator is asynchronous to the MCU clock, there is more error attributable to internal system clock noise. A/D converter accuracy is reduced slightly while the internal RC oscillator is being used (CSEL = 1).

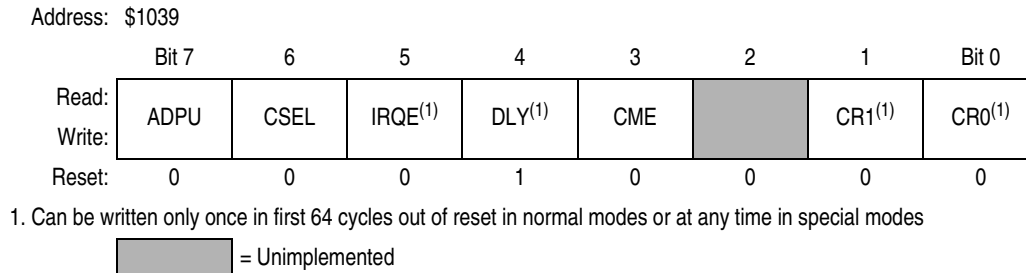


Figure 3-4. System Configuration Options Register (OPTION)

ADPU — A/D Power-Up Bit

- 0 = A/D powered down
- 1 = A/D powered up

CSEL — Clock Select Bit

- 0 = A/D and EEPROM use system E clock.
- 1 = A/D and EEPROM use internal RC clock.

IRQE — Configure $\overline{\text{IRQ}}$ for Edge-Sensitive Only Operation

Refer to [Chapter 5 Resets and Interrupts](#).

DLY — Enable Oscillator Startup Delay Bit

- 0 = The oscillator startup delay coming out of stop is bypassed and the MCU resumes processing within about four bus cycles.
- 1 = A delay of approximately 4000 E-clock cycles is imposed as the MCU is started up from the stop power-saving mode. This delay allows the crystal oscillator to stabilize.

CME — Clock Monitor Enable Bit

Refer to [Chapter 5 Resets and Interrupts](#).

Bit 2 — Not implemented

Always reads 0

CR[1:0] — COP Timer Rate Select Bits

Refer to [Chapter 5 Resets and Interrupts](#) and [Chapter 9 Timing Systems](#).

When this control bit is clear, the four requested conversions are performed once to fill the four result registers. When this control bit is set, conversions are performed continuously with the result registers updated as data becomes available.

MULT — Multiple Channel/Single Channel Control Bit

When this bit is clear, the A/D converter system is configured to perform four consecutive conversions on the single channel specified by the four channel select bits CD:CA (bits [3:0] of the ADCTL register). When this bit is set, the A/D system is configured to perform a conversion on each of four channels where each result register corresponds to one channel.

NOTE

When the multiple-channel continuous scan mode is used, extra care is needed in the design of circuitry driving the A/D inputs. The charge on the capacitive DAC array before the sample time is related to the voltage on the previously converted channel. A charge share situation exists between the internal DAC capacitance and the external circuit capacitance. Although the amount of charge involved is small, the rate at which it is repeated is every 64 μ s for an E clock of 2 MHz. The RC charging rate of the external circuit must be balanced against this charge sharing effect to avoid errors in accuracy. Refer to M68HC11 Reference Manual, Freescale document order number M68HC11RM/AD, for further information.

CD:CA — Channel Selects D:A Bits

Refer to Table 3-2. When a multiple channel mode is selected (MULT = 1), the two least significant channel select bits (CB and CA) have no meaning and the CD and CC bits specify which group of four channels is to be converted.

Table 3-2. A/D Converter Channel Selection

Channel Select Control Bits	Channel Signal	Result in ADRx if MULT = 1
CD:CC:CB:CA		
0000	AN0	ADR1
0001	AN1	ADR2
0010	AN2	ADR3
0011	AN3	ADR4
0100	AN4	ADR1
0101	AN5	ADR2
0110	AN6	ADR3
0111	AN7	ADR4
10XX	Reserved	—
1100	$V_{RH}^{(1)}$	ADR1
1101	$V_{RL}^{(1)}$	ADR2
1110	$(V_{RH})/2^{(1)}$	ADR3
1111	Reserved ⁽¹⁾	ADR4

1. Used for factory testing

4.4 Opcodes and Operands

The M68HC11 Family of microcontrollers uses 8-bit opcodes. Each opcode identifies a particular instruction and associated addressing mode to the CPU. Several opcodes are required to provide each instruction with a range of addressing capabilities. Only 256 opcodes would be available if the range of values were restricted to the number able to be expressed in 8-bit binary numbers.

A 4-page opcode map has been implemented to expand the number of instructions. An additional byte, called a prebyte, directs the processor from page 0 of the opcode map to one of the other three pages. As its name implies, the additional byte precedes the opcode.

A complete instruction consists of a prebyte, if any, an opcode, and zero, one, two, or three operands. The operands contain information the CPU needs for executing the instruction. Complete instructions can be from one to five bytes long.

4.5 Addressing Modes

Six addressing modes can be used to access memory:

- Immediate
- Direct
- Extended
- Indexed
- Inherent
- Relative

These modes are detailed in the following paragraphs. All modes except inherent mode use an effective address. The effective address is the memory address from which the argument is fetched or stored or the address from which execution is to proceed. The effective address can be specified within an instruction, or it can be calculated.

4.5.1 Immediate

In the immediate addressing mode, an argument is contained in the byte(s) immediately following the opcode. The number of bytes following the opcode matches the size of the register or memory location being operated on. There are 2-, 3-, and 4- (if prebyte is required) byte immediate instructions. The effective address is the address of the byte following the instruction.

4.5.2 Direct

In the direct addressing mode, the low-order byte of the operand address is contained in a single byte following the opcode, and the high-order byte of the address is assumed to be \$00. Addresses \$00–\$FF are thus accessed directly, using 2-byte instructions. Execution time is reduced by eliminating the additional memory access required for the high-order address byte. In most applications, this 256-byte area is reserved for frequently referenced data. In M68HC11 MCUs, the memory map can be configured for combinations of internal registers, RAM, or external memory to occupy these addresses.

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 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.

Parallel Input/Output (I/O) Ports

PORTCL is used in the handshake clearing mechanism. When an active edge occurs on the STRA pin, port C data is latched into the PORTCL register. Reads of this register return the last value latched into PORTCL and clear STAF flag (following a read of PIOC with STAF set).

Address:	\$1007							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 6-6. Port C Data Direction Register (DDRC)

DDRC[7:0] — Port C Data Direction Bits

In the 3-state variation of output handshake mode, clear the corresponding DDRC bits. Refer to [Figure 10-13. 3-State Variation of Output Handshake Timing Diagram \(STRA Enables Output Buffer\)](#).

0 = Input

1 = Output

6.5 Port D

In all modes, port D bits [5:0] can be used either for general-purpose I/O or with the serial communications interface (SCI) and serial peripheral interface (SPI) subsystems. During reset, port D pins PD[5:0] are configured as high-impedance inputs (DDRD bits cleared).

Address:	\$1008							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	PD5	PD4	PD3	PD2	PD1	PD0
Write:								
Reset:	—	—	I	I	I	I	I	I
Alternate Function:	—	—	PD5 \overline{SS}	PD4 SCK	PD3 MOSI	PD2 MISO	PD1 Tx	PD0 RxD

I = Indeterminate after reset

Figure 6-7. Port D Data Register (PORTD)

Address:	\$1009							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:			DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Write:								
Reset:	0	0	0	0	0	0	0	0


 = Unimplemented

Figure 6-8. Port D Data Direction Register (DDRD)

Bits [7:6] — Unimplemented

Always read 0

DDRD[5:0] — Port D Data Direction Bits

When DDRD bit 5 is 1 and MSTR = 1 in SPCR, PD5/ \overline{SS} is a general-purpose output and mode fault logic is disabled.

0 = Input

1 = Output

FE — Framing Error Flag

FE is set when a 0 is detected where a stop bit was expected. Clear the FE flag by reading SCSR with FE set and then reading SCDR.

0 = Stop bit detected

1 = Zero detected

Bit 0 — Unimplemented

Always reads 0

7.7.5 Baud Rate Register

Use this register to select different baud rates for the SCI system. The SCP[1:0] (SCP[2:0] in MC68HC(7)11E20) bits function as a prescaler 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](#) for normal baud rate selections.

Address:	\$102B							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TCLR	SCP2	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0
Write:								
Reset:	0	0	0	0	0	U	U	U
	U = Unaffected							

Figure 7-7. Baud Rate Register (BAUD)

TCLR — Clear Baud Rate Counter Bit (Test)

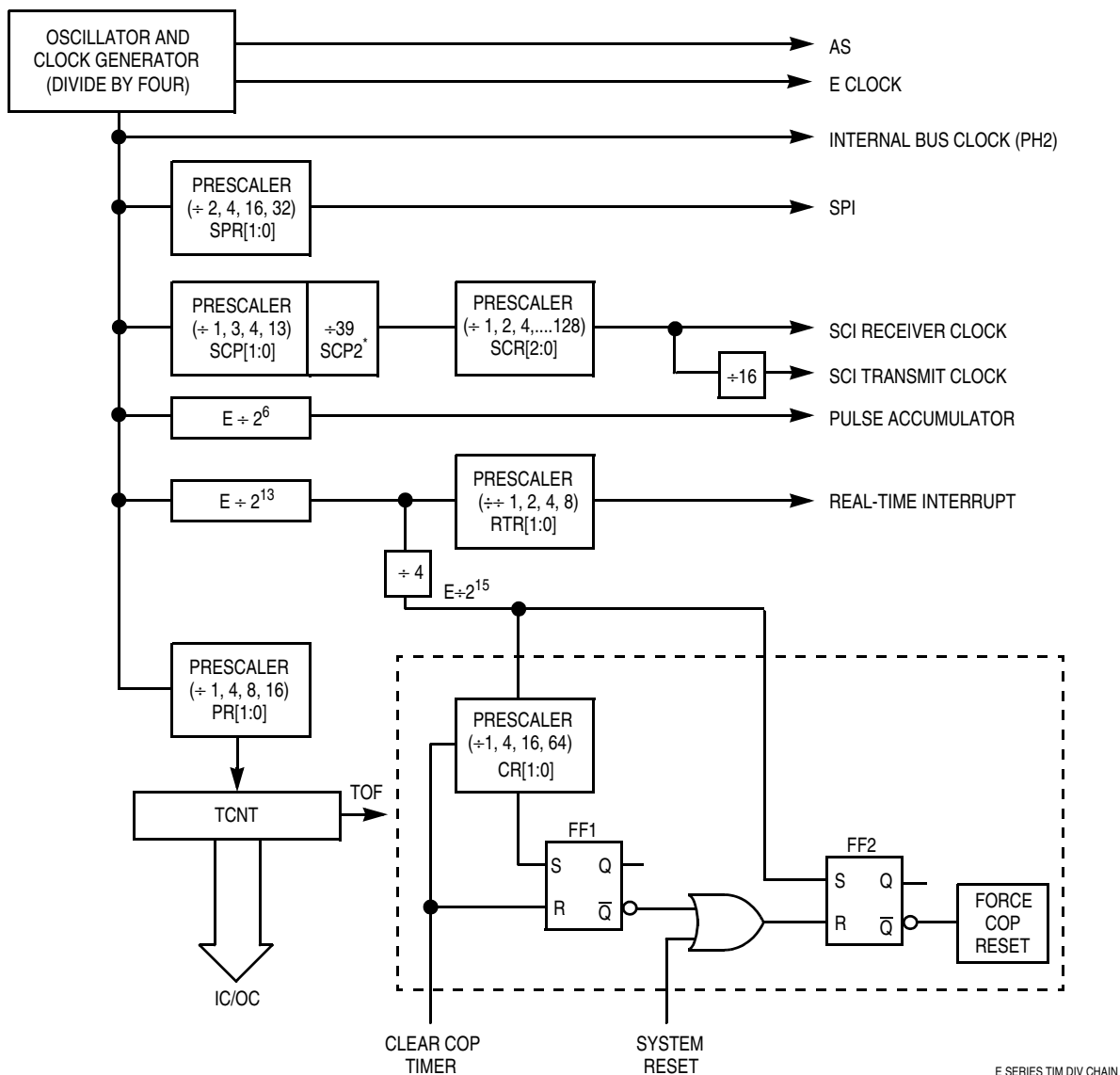
SCP[2:0] — SCI Baud Rate Prescaler Select Bits

NOTE

SCP2 applies to MC68HC(7)11E20 only. When SCP2 = 1, SCP[1:0] must equal 0s. Any other values for SCP[1:0] are not decoded in the prescaler and the results are unpredictable. Refer to [Figure 7-8](#) and [Figure 7-9](#).

RCKB — SCI Baud Rate Clock Check Bit (Test)

See [Table 7-1](#).



* SCP2 present on MC68HC(7)11E20 only

Figure 9-1. Timer Clock Divider Chains

Register name: Timer Output Compare 3 Register (High) Address: \$101A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	1	1	1	1	1	1	1	1

Register name: Timer Output Compare 3 Register (Low) Address: \$101B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 9-10. Timer Output Compare 3 Register Pair (TOC3)

Register name: Timer Output Compare 4 Register (High) Address: \$101C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	1	1	1	1	1	1	1	1

Register name: Timer Output Compare 4 Register (Low) Address: \$101D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 9-11. Timer Output Compare 4 Register Pair (TOC4)

9.4.2 Timer Compare Force Register

The CFORC register allows forced early compares. FOC[1:5] correspond to the five output compares. These bits are set for each output compare that is to be forced. The action taken as a result of a forced compare is the same as if there were a match between the OCx register and the free-running counter, except that the corresponding interrupt status flag bits are not set. The forced channels trigger their programmed pin actions to occur at the next timer count transition after the write to CFORC.

The CFORC bits should not be used on an output compare function that is programmed to toggle its output on a successful compare because a normal compare that occurs immediately before or after the force can result in an undesirable operation.

Address: \$100B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	FOC1	FOC2	FOC3	FOC4	FOC5			
Write:	FOC1	FOC2	FOC3	FOC4	FOC5			
Reset:	0	0	0	0	0	0	0	0


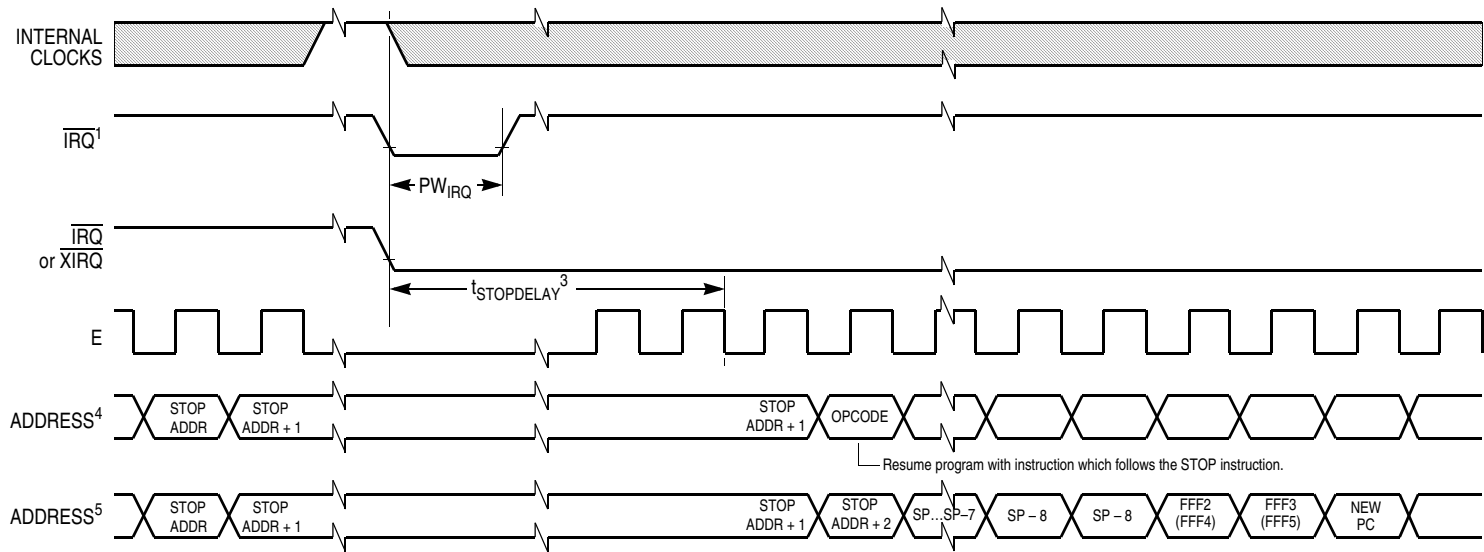
 = Unimplemented

Figure 9-12. Timer Compare Force Register (CFORC)

PAIF and PAIF — Pulse Accumulator Input Edge Interrupt Enable Bit and Flag

The PAIF status bit is automatically set each time a selected edge is detected at the PA7/PAI/OC1 pin. To clear this status bit, write to the TFLG2 register with a 1 in the corresponding data bit position (bit 4). The PAIF control bit allows configuring the pulse accumulator input edge detect for polled or interrupt-driven operation but does not affect setting or clearing the PAIF bit. When PAIF is 0, pulse accumulator input interrupts are inhibited, and the system operates in a polled mode. In this mode, the PAIF bit must be polled by user software to determine when an edge has occurred. When the PAIF control bit is set, a hardware interrupt request is generated each time PAIF is set. Before leaving the interrupt service routine, software must clear PAIF by writing to the TFLG2 register.





Notes:

1. Edge Sensitive \overline{IRQ} pin (IRQE bit = 1)
2. Level sensitive \overline{IRQ} pin (IRQE bit = 0)
3. $t_{STOPDELAY} = 4064 t_{CYC}$ if DLY bit = 1 or $4 t_{CYC}$ if DLY = 0.
4. \overline{XIRQ} with X bit in CCR = 1.
5. IRQ or (\overline{XIRQ} with X bit in CCR = 0).

Figure 10-4. STOP Recovery Timing Diagram

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 $t_{AV} = PW_{EL} - (t_{ASD} + 80 \text{ ns})^{(2) (3)a}$	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, $t_{DDW} = 1/8 t_{CYC} + 65.5 \text{ ns}^{(2) (3)a}$	t_{DDW}	—	190.5	—	128	—	71	ns
21	Write data hold time, $t_{DHW} = 1/8 t_{CYC} - 29.5 \text{ ns}^{(2) (3)a}$	t_{DHW}	95.5	—	33	—	26	—	ns
22	Multiplexed address valid time to E rise $t_{AVM} = PW_{EL} - (t_{ASD} + 90 \text{ ns})^{(2) (3)a}$	t_{AVM}	271.5	—	84	—	54	—	ns
24	Multiplexed address valid time to AS fall $t_{ASL} = PW_{ASH} - 70 \text{ ns}^{(2)}$	t_{ASL}	151	—	26	—	13	—	ns
25	Multiplexed address hold time $t_{AHL} = 1/8 t_{CYC} - 29.5 \text{ ns}^{(2) (3)b}$	t_{AHL}	95.5	—	33	—	31	—	ns
26	Delay time, E to AS rise, $t_{ASD} = 1/8 t_{CYC} - 9.5 \text{ ns}^{(2) (3)a}$	t_{ASD}	115.5	—	53	—	31	—	ns
27	Pulse width, AS high, $PW_{ASH} = 1/4 t_{CYC} - 29 \text{ ns}^{(2)}$	PW_{ASH}	221	—	96	—	63	—	ns
28	Delay time, AS to E rise, $t_{ASED} = 1/8 t_{CYC} - 9.5 \text{ ns}^{(2) (3)b}$	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) $t_{MAD} = t_{ASD} + 30 \text{ ns}^{(2) (3)a}$	t_{MAD}	145.5	—	83	—	51	—	ns

1. $V_{DD} = 5.0 \text{ Vdc} \pm 10\%$, $V_{SS} = 0 \text{ Vdc}$, $T_A = T_L$ 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:

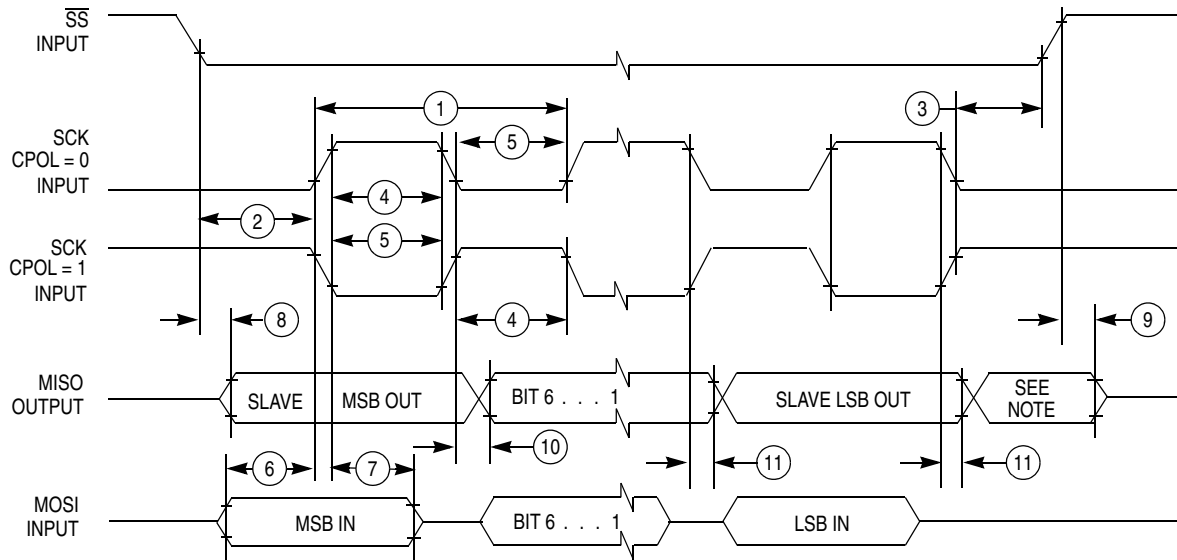
(a) $(1 - dc) \times 1/4 t_{CYC}$

(b) $dc \times 1/4 t_{CYC}$

Where:

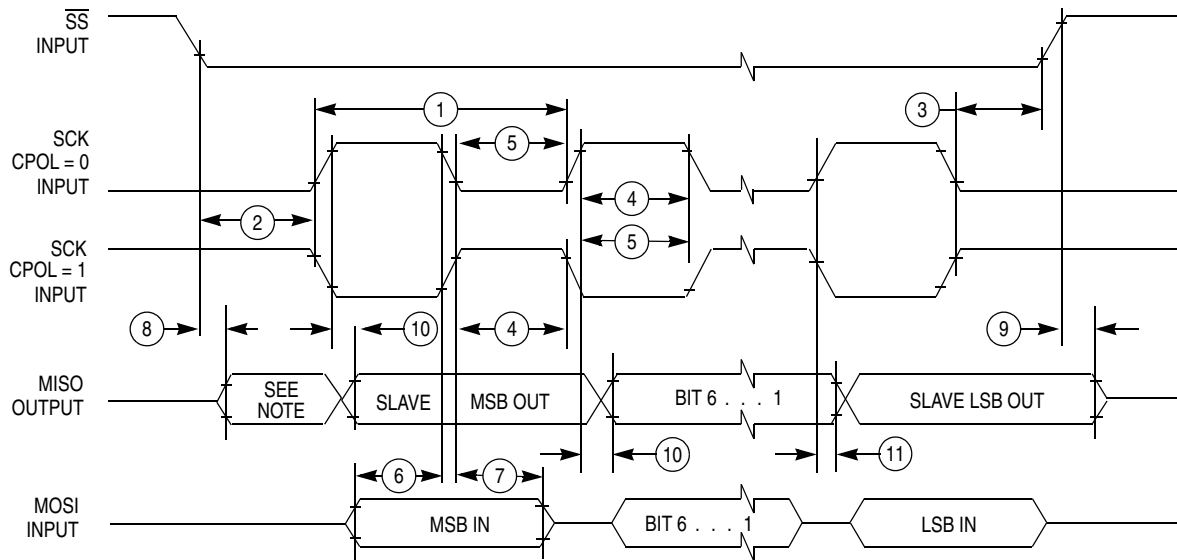
dc is the decimal value of duty cycle percentage (high time)

Electrical Characteristics



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)

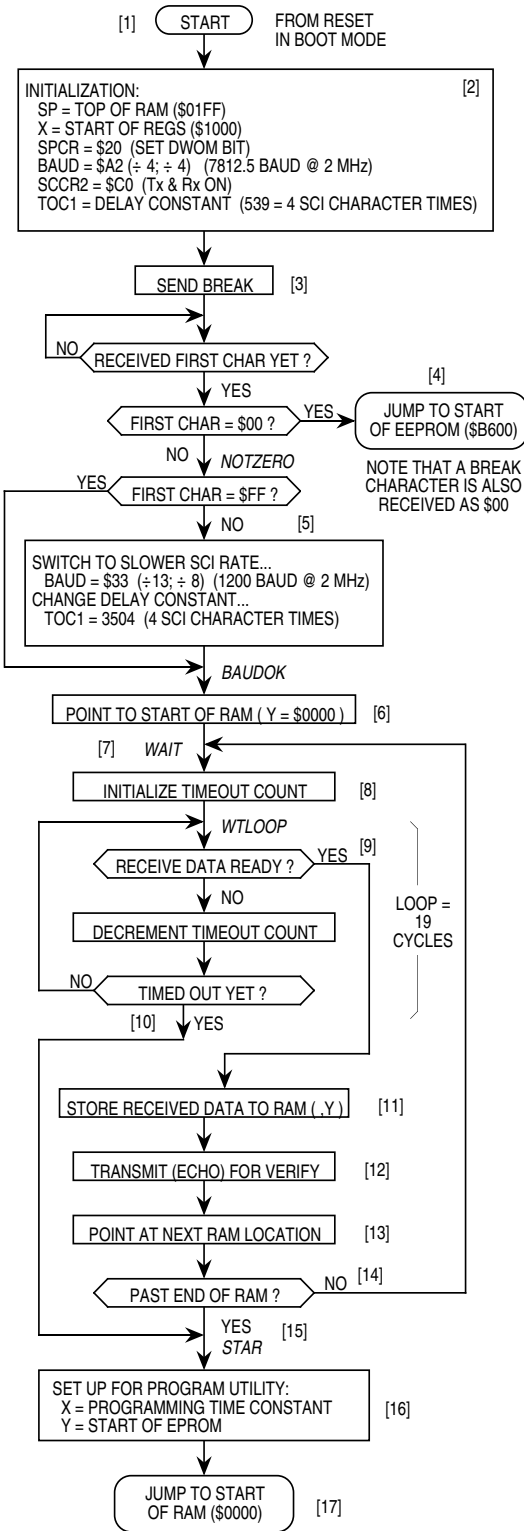


Figure 3. MC68HC711E9 Bootloader Flowchart

Allowing for Bootstrap Mode

In systems where the PD0/RxD pin is normally used as a general-purpose output, a serial signal from the host can be connected to the pin without resulting in output driver conflicts. It may be important to consider what the existing logic will do with the SCI serial data instead of the signals that would have been produced by the PD0 pin. In systems where the PD0 pin is used normally as a general-purpose input, the driver circuit that drives the PD0 pin must be designed so that the serial data can override this driver, or the driver must be disconnected during the bootstrap download. A simple series resistor between the driver and the PD0 pin solves this problem as shown in [Figure 5](#). The serial data from the host computer can then be connected to the PD0/RxD pin, and the series resistor will prevent direct conflict between the host driver and the normal PD0 driver.

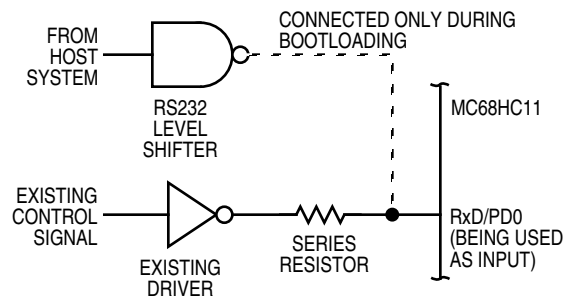


Figure 5. Preventing Driver Conflict

TxD Pin

The bootloader program uses the PD1/TxD pin to send verification data back to the host computer. To minimize the possibility of conflicts with circuitry connected to this pin, port D is configured for wire-OR mode by the bootloader program during initialization. Since the wire-OR configuration prevents the pin from driving active high levels, a pullup resistor to V_{DD} is needed if the TxD signal is used.

In systems where the PD1/TxD pin is normally used as a general-purpose output, there are no output driver conflicts. It may be important to consider what the existing logic will do with the SCI serial data instead of the signals that would have been produced by the PD1 pin.

In systems where the PD1 pin is normally used as a general-purpose input, the driver circuit that drives the PD1 pin must be designed so that the PD1/TxD pin driver in the MCU can override this driver. A simple series resistor between the driver and the PD1 pin can solve this problem. The TxD pin can then be configured as an output, and the series resistor will prevent direct conflict between the internal TxD driver and the external driver connected to PD1 through the series resistor.

Other

The bootloader firmware sets the DWOM control bit, which configures all port D pins for wire-OR operation. During the bootloading process, all port D pins except the PD1/TxD pin are configured as high-impedance inputs. Any port D pin that normally is used as an output should have a pullup resistor so it does not float during the bootloading process.

Listing 2. BASIC Program for Personal Computer

```

1090 BYTECOUNT = BYTECOUNT - 3      'ADJUST FOR ADDRESS + CHECKSUM
1099 REM ***** NEXT 4 HEX DIGITS BECOME THE STARTING ADDRESS FOR THE DATA *****
1100 GOSUB 6000                        'GET FIRST NIBBLE OF ADDRESS
1102 GOSUB 7000                        'CONVERT TO DECIMAL
1104 ADDRESS= 4096 * X
1106 GOSUB 6000                        'GET NEXT NIBBLE
1108 GOSUB 7000
1110 ADDRESS= ADDRESS+ 256 * X
1112 GOSUB 6000
1114 GOSUB 7000
1116 ADDRESS= ADDRESS+ 16 * X
1118 GOSUB 6000
1120 GOSUB 7000
1122 ADDRESS= ADDRESS+ X
1124 ARRAYCNT = ADDRESS-ADRSTART        'INDEX INTO ARRAY
1129 REM ***** CONVERT THE DATA DIGITS TO BINARY AND SAVE IN THE ARRAY *****
1130 FOR I = 1 TO BYTECOUNT
1140 GOSUB 6000
1150 GOSUB 7000
1160 Y = 16 * X                        'SAVE UPPER NIBBLE OF BYTE
1170 GOSUB 6000
1180 GOSUB 7000
1190 Y = Y + X                          'ADD LOWER NIBBLE
1200 CODE%(ARRAYCNT) = Y               'SAVE BYTE IN ARRAY
1210 ARRAYCNT = ARRAYCNT + 1           'INCREMENT ARRAY INDEX
1220 NEXT I
1230 GOTO 1000
1250 CLOSE 1
1499 REM ***** DUMP BOOTLOAD CODE TO PART *****
1500 'OPEN "R",#2,"COM1:1200,N,8,1" 'Macintosh COM statement
1505 OPEN "COM1:1200,N,8,1,CD0,CS0,DS0,RS" FOR RANDOM AS #2 'DOS COM statement
1510 INPUT "Comm port open"; Q$
1512 WHILE LOC(2) >0                    'FLUSH INPUT BUFFER
1513 GOSUB 8020
1514 WEND
1515 PRINT : PRINT "Sending bootload code to target part..."
1520 A$ = CHR$(255) + BOOTCODE$ 'ADD HEX FF TO SET BAUD RATE ON TARGET HC11
1530 GOSUB 6500
1540 PRINT
1550 FOR I = 1 TO BOOTCOUNT            '# OF BYTES IN BOOT CODE BEING ECHOED
1560 GOSUB 8000
1564 K=ASC(B$):GOSUB 8500
1565 PRINT "Character #"; I; " received = "; HX$
1570 NEXT I
1590 PRINT "Programming is ready to begin.": INPUT "Are you ready"; Q$
1595 CLS
1597 WHILE LOC(2) > 0                    'FLUSH INPUT BUFFER
1598 GOSUB 8020
1599 WEND
1600 XMT = 0: RCV = 0                   'POINTERS TO XMIT AND RECEIVE BYTES
1610 A$ = CHR$(CODE%(XMT))
1620 GOSUB 6500                          'SEND FIRST BYTE
1625 FOR I = 1 TO CODESIZE% - 1          'ZERO BASED ARRAY 0 -> CODESIZE-1
1630 A$ = CHR$(CODE%(I))                 'SEND SECOND BYTE TO GET ONE IN QUEUE
1635 GOSUB 6500                          'SEND IT

```

```

1640 GOSUB 8000          'GET BYTE FOR VERIFICATION
1650 RCV = I - 1
1660 LOCATE 10,1:PRINT "Verifying byte #"; I; "      "
1664 IF CHR$(CODE%(RCV)) = B$ THEN 1670
1665 K=CODE%(RCV):GOSUB 8500
1666 LOCATE 1,1:PRINT "Byte #"; I; "      ", " - Sent "; HX$;
1668 K=ASC(B$):GOSUB 8500
1669 PRINT "  Received "; HX$;
1670 NEXT I
1680 GOSUB 8000          'GET BYTE FOR VERIFICATION
1690 RCV = CODESIZE% - 1
1700 LOCATE 10,1:PRINT "Verifying byte #"; CODESIZE%; "      "
1710 IF CHR$(CODE%(RCV)) = B$ THEN 1720
1713 K=CODE(RCV):GOSUB 8500
1714 LOCATE 1,1:PRINT "Byte #"; CODESIZE%; "      ", " - Sent "; HX$;
1715 K=ASC(B$):GOSUB 8500
1716 PRINT "  Received "; HX$;
1720 LOCATE 8, 1: PRINT : PRINT "Done!!!!"
4900 CLOSE
4910 INPUT "Press [RETURN] to quit...", Q$
5000 END
5900 '*****
5910 '*          SUBROUTINE TO READ IN ONE BYTE FROM A DISK FILE
5930 '*          RETURNS BYTE IN A$
5940 '*****
6000 FLAG = 0
6010 IF EOF(1) THEN FLAG = 1: RETURN
6020 A$ = INPUT$(1, #1)
6030 RETURN
6490 '*****
6492 '*          SUBROUTINE TO SEND THE STRING IN A$ OUT TO THE DEVICE
6494 '*          OPENED AS FILE #2.
6496 '*****
6500 PRINT #2, A$;
6510 RETURN
6590 '*****
6594 '*          SUBROUTINE THAT CONVERTS THE HEX DIGIT IN A$ TO AN INTEGER
6596 '*****
7000 X = INSTR(H$, A$)
7010 IF X = 0 THEN FLAG = 1
7020 X = X - 1
7030 RETURN
7990 '*****
7992 '*          SUBROUTINE TO READ IN ONE BYTE THROUGH THE COMM PORT OPENED
7994 '*          AS FILE #2.  WAITS INDEFINITELY FOR THE BYTE TO BE
7996 '*          RECEIVED.  SUBROUTINE WILL BE ABORTED BY ANY
7998 '*          KEYBOARD INPUT.  RETURNS BYTE IN B$.  USES Q$.
7999 '*****
8000 WHILE LOC(2) = 0          'WAIT FOR COMM PORT INPUT
8005 Q$ = INKEY$: IF Q$ <> "" THEN 4900 'IF ANY KEY PRESSED, THEN ABORT
8010 WEND
8020 B$ = INPUT$(1, #2)
8030 RETURN
8490 '*****

```

Listing 3. MC68HC711E9 Bootloader ROM

```

1          *****
2          * BOOTLOADER FIRMWARE FOR 68HC711E9 - 21 Aug 89
3          *****
4          * Features of this bootloader are...
5          *
6          * Auto baud select between 7812.5 and 1200 (8 MHz)
7          * 0 - 512 byte variable length download
8          * Jump to EEPROM at $B600 if 1st download byte = $00
9          * PROGRAM - Utility subroutine to program EPROM
10         * UPLOAD - Utility subroutine to dump memory to host
11         * Mask I.D. at $BFD4 = $71E9
12         *****
13         * Revision A -
14         *
15         * Fixed bug in PROGRAM routine where the first byte
16         * programmed into the EPROM was not transmitted for
17         * verify.
18         * Also added to PROGRAM routine a skip of bytes
19         * which were already programmed to the value desired.
20         *
21         * This new version allows variable length download
22         * by quitting reception of characters when an idle
23         * of at least four character times occurs
24         *
25         *****
26
27         * EQUATES FOR USE WITH INDEX OFFSET = $1000
28         *
29 0008     PORTD      EQU      $08
30 000E     TCNT       EQU      $0E
31 0016     TOC1       EQU      $16
32 0023     TFLG1      EQU      $23
33         * BIT EQUATES FOR TFLG1
34 0080     OC1F       EQU      $80
35         *
36 0028     SPCR       EQU      $28                      (FOR DWOM BIT)
37 002B     BAUD       EQU      $2B
38 002D     SCCR2      EQU      $2D
39 002E     SCSR       EQU      $2E
40 002F     SCDAT      EQU      $2F
41 003B     PPROG      EQU      $3B
42         * BIT EQUATES FOR PPROG
43 0020     ELAT       EQU      $20
44 0001     EPGM       EQU      $01
45         *
46
47         * MEMORY CONFIGURATION EQUATES
48         *
49 B600     EEPMSTR     EQU      $B600                    Start of EEPROM
50 B7FF     EEPMEND     EQU      $B7FF                    End of EEPROM
51         *

```

Listing 3. MC68HC711E9 Bootloader ROM

```

213
214 *****
215 * Boot ROM revision level in ASCII
216 *      (ORG      $BFD1)
217 BFD1 41      FCC      "A"
218 *****
219 * Mask set I.D. ($0000 FOR EPROM PARTS)
220 *      (ORG      $BFD2)
221 BFD2 0000      FDB      $0000
222 *****
223 * '711E9 I.D. - Can be used to determine MCU type
224 *      (ORG      $BFD4)
225 BFD4 71E9      FDB      $71E9
226
227 *****
228 * VECTORS - point to RAM for pseudo-vector JUMPs
229
230 BFD6 00C4      FDB      $100-60      SCI
231 BFD8 00C7      FDB      $100-57      SPI
232 BFDA 00CA      FDB      $100-54      PULSE ACCUM INPUT EDGE
233 BFDC 00CD      FDB      $100-51      PULSE ACCUM OVERFLOW
234 BFDE 00D0      FDB      $100-48      TIMER OVERFLOW
235 BFE0 00D3      FDB      $100-45      TIMER OUTPUT COMPARE 5
236 BFE2 00D6      FDB      $100-42      TIMER OUTPUT COMPARE 4
237 BFE4 00D9      FDB      $100-39      TIMER OUTPUT COMPARE 3
238 BFE6 00DC      FDB      $100-36      TIMER OUTPUT COMPARE 2
239 BFE8 00DF      FDB      $100-33      TIMER OUTPUT COMPARE 1
240 BFEA 00E2      FDB      $100-30      TIMER INPUT CAPTURE 3
241 BFEC 00E5      FDB      $100-27      TIMER INPUT CAPTURE 2
242 BFEE 00E8      FDB      $100-24      TIMER INPUT CAPTURE 1
243 BFF0 00EB      FDB      $100-21      REAL TIME INT
244 BFF2 00EE      FDB      $100-18      IRQ
245 BFF4 00F1      FDB      $100-15      XIRQ
246 BFF6 00F4      FDB      $100-12      SWI
247 BFF8 00F7      FDB      $100-9       ILLEGAL OP-CODE
248 BFFA 00FA      FDB      $100-6       COP FAIL
249 BFFC 00FD      FDB      $100-3       CLOCK MONITOR
250 BFFE BF54      FDB      BEGIN      RESET
251 C000      END

```

Symbol Table:

Symbol Name	Value	Def.#	Line Number	Cross Reference
BAUD	002B	*00037	00160	00180
BAUDOK	BF8A	*00183	00178	
BEGIN	BF54	*00155	00250	
DELAYF	021B	*00061	00163	
DELAYS	0DB0	*00060	00181	
DONEIT	BF47	*00142	00124	
EEPMEND	B7FF	*00050		
EEPMSTR	B600	*00049	00175	
ELAT	0020	*00043	00125	00128
EPGM	0001	*00044	00128	
EPRMEND	FFFF	*00053		
EPRMSTR	D000	*00052	00206	