

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

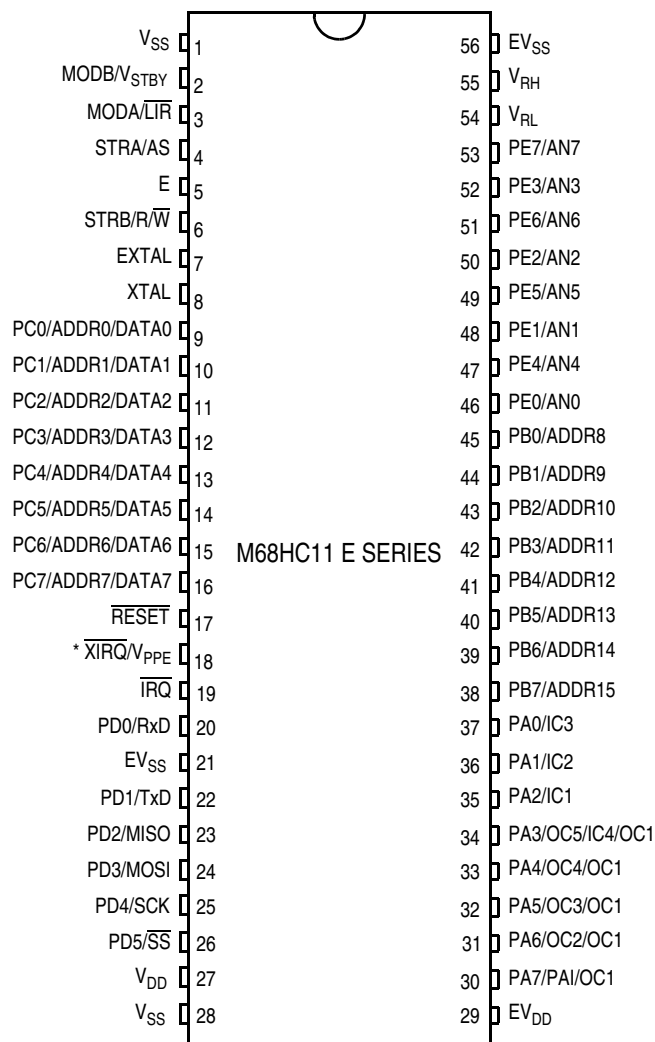
Product Status	Obsolete
Core Processor	HC11
Core Size	8-Bit
Speed	2MHz
Connectivity	SCI, SPI
Peripherals	POR, WDT
Number of I/O	38
Program Memory Size	12KB (12K x 8)
Program Memory Type	OTP
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°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/mc68hc711e9mfne2

4.3	Data Types	69
4.4	Opcodes and Operands	70
4.5	Addressing Modes	70
4.5.1	Immediate	70
4.5.2	Direct	70
4.5.3	Extended	71
4.5.4	Indexed	71
4.5.5	Inherent	71
4.5.6	Relative	71
4.6	Instruction Set	71

Chapter 5

Resets and Interrupts

5.1	Introduction	79
5.2	Resets	79
5.2.1	Power-On Reset (POR)	79
5.2.2	External Reset (RESET)	80
5.2.3	Computer Operating Properly (COP) Reset	80
5.2.4	Clock Monitor Reset	81
5.2.5	System Configuration Options Register	82
5.2.6	Configuration Control Register	83
5.3	Effects of Reset	83
5.3.1	Central Processor Unit (CPU)	83
5.3.2	Memory Map	84
5.3.3	Timer	84
5.3.4	Real-Time Interrupt (RTI)	84
5.3.5	Pulse Accumulator	84
5.3.6	Computer Operating Properly (COP)	84
5.3.7	Serial Communications Interface (SCI)	84
5.3.8	Serial Peripheral Interface (SPI)	84
5.3.9	Analog-to-Digital (A/D) Converter	85
5.3.10	System	85
5.4	Reset and Interrupt Priority	85
5.4.1	Highest Priority Interrupt and Miscellaneous Register	86
5.5	Interrupts	87
5.5.1	Interrupt Recognition and Register Stacking	88
5.5.2	Non-Maskable Interrupt Request (XIRQ)	89
5.5.3	Illegal Opcode Trap	89
5.5.4	Software Interrupt (SWI)	90
5.5.5	Maskable Interrupts	90
5.5.6	Reset and Interrupt Processing	90
5.6	Low-Power Operation	90
5.6.1	Wait Mode	90
5.6.2	Stop Mode	95



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

Figure 1-5. Pin Assignments for 56-Pin SDIP

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.										
\$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					
<div><div></div> = Unimplemented</div> <div><div>R</div> = Reserved</div> <div>U = Unaffected</div> <div>I = Indeterminate after reset</div>										

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$103E	Reserved		R	R	R	R	R	R	R	R
\$103F	System Configuration Register (CONFIG) See page 43.	Read:					NOSEC	NOCOP	ROMON	EEON
		Write:								
		Reset:	0	0	0	0	U	U	1	U
\$103F	System Configuration Register (CONFIG) ⁽³⁾ See page 43.	Read:	EE3	EE2	EE1	EE0	NOSEC	NOCOP		EEON
		Write:								
		Reset:	1	1	1	1	U	U	1	1

1. Can be written only once in first 64 cycles out of reset in normal modes or at any time during special modes.

2. MC68HC711E9 only

3. MC68HC811E2 only


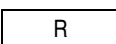
 = Unimplemented
 = Reserved
 U = Unaffected
 I = Indeterminate after reset

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

2.3.1 RAM and Input/Output Mapping

Hardware priority is built into RAM and I/O mapping. Registers have priority over RAM and RAM has priority over ROM. When a lower priority resource is mapped at the same location as a higher priority resource, a read/write of a location results in a read/write of the higher priority resource only. For example, if both the register block and the RAM are mapped to the same location, only the register block will be accessed. If RAM and ROM are located at the same position, RAM has priority.

The fully static RAM can be used to store instructions, variables, and temporary data. The direct addressing mode can access RAM locations using a 1-byte address operand, saving program memory space and execution time, depending on the application.

RAM contents can be preserved during periods of processor inactivity by two methods, both of which reduce power consumption. They are:

1. In the software-based stop mode, the clocks are stopped while V_{DD} powers the MCU. Because power supply current is directly related to operating frequency in CMOS integrated circuits, only a very small amount of leakage exists when the clocks are stopped.
2. In the second method, the MODB/ V_{STBY} pin can supply RAM power from a battery backup or from a second power supply. [Figure 2-8](#) shows a typical standby voltage circuit for a standard 5-volt device. Adjustments to the circuit must be made for devices that operate at lower voltages. Using the MODB/ V_{STBY} pin may require external hardware, but can be justified when a significant amount of external circuitry is operating from V_{DD} . If V_{STBY} is used to maintain RAM contents, reset must be held low whenever V_{DD} is below normal operating level. Refer to [Chapter 5 Resets and Interrupts](#).

2.3.3.1 System Configuration Register

The system configuration register (CONFIG) consists of an EEPROM byte and static latches that control the startup configuration of the MCU. The contents of the EEPROM byte are transferred into static working latches during reset sequences. The operation of the MCU is controlled directly by these latches and not by CONFIG itself. In normal modes, changes to CONFIG do not affect operation of the MCU until after the next reset sequence. When programming, the CONFIG register itself is accessed. When the CONFIG register is read, the static latches are accessed. See [2.5.1 EEPROM and CONFIG Programming and Erasure](#) for information on modifying CONFIG.

To take full advantage of the MCU's functionality, customers can program the CONFIG register in bootstrap mode. This can be accomplished by setting the mode pins to logic 0 and downloading a small program to internal RAM. For more information, Freescale application note AN1060 entitled [M68HC11 Bootstrap Mode](#) has been included at the back of this document. The downloadable talker will consist of:

- Bulk erase
- Byte programming
- Communication server

All of this functionality is provided by PCbug11 which can be found on the Freescale Web site at <http://www.freescale.com>. For more information on using PCbug11 to program an E-series device, Freescale engineering bulletin EB296 entitled [Programming MC68HC711E9 Devices with PCbug11 and the M68HC11EVBU](#) has been included at the back of this document.

NOTE

The CONFIG register on the 68HC11 is an EEPROM cell and must be programmed accordingly.

Operation of the CONFIG register in the MC68HC811E2 differs from other devices in the M68HC11 E series. See [Figure 2-10](#) and [Figure 2-11](#).

Address: \$103F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:					NOSEC	NOCOP	ROMON	EEON
Write:								

Resets:

Single chip:	0	0	0	0	U	U	1	U
Bootstrap:	0	0	0	0	U	U(L)	U	U
Expanded:	0	0	0	0	1	U	U	U
Test:	0	0	0	0	1	U(L)	U	U

= Unimplemented

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

Figure 2-10. System Configuration Register (CONFIG)

end of the interrupt service routine, the return-from-interrupt instruction is executed and the saved registers are pulled from the stack in reverse order so that normal program execution can resume. Refer to [Chapter 4 Central Processor Unit \(CPU\)](#).

Table 5-5. Stacking Order on Entry to Interrupts

Memory Location	CPU Registers
SP	PCL
SP-1	PCH
SP-2	IYL
SP-3	IYH
SP-4	IXL
SP-5	IXH
SP-6	ACCA
SP-7	ACCB
SP-8	CCR

5.5.2 Non-Maskable Interrupt Request ($\overline{\text{XIRQ}}$)

Non-maskable interrupts are useful because they can always interrupt CPU operations. The most common use for such an interrupt is for serious system problems, such as program runaway or power failure. The $\overline{\text{XIRQ}}$ input is an updated version of the $\overline{\text{NMI}}$ (non-maskable interrupt) input of earlier MCUs.

Upon reset, both the X bit and I bit of the CCR are set to inhibit all maskable interrupts and $\overline{\text{XIRQ}}$. After minimum system initialization, software can clear the X bit by a TAP instruction, enabling $\overline{\text{XIRQ}}$ interrupts. Thereafter, software cannot set the X bit. Thus, an $\overline{\text{XIRQ}}$ interrupt is a non-maskable interrupt. Because the operation of the I-bit-related interrupt structure has no effect on the X bit, the internal $\overline{\text{XIRQ}}$ pin remains unmasked. In the interrupt priority logic, the $\overline{\text{XIRQ}}$ interrupt has a higher priority than any source that is maskable by the I bit. All I-bit-related interrupts operate normally with their own priority relationship.

When an I-bit-related interrupt occurs, the I bit is automatically set by hardware after stacking the CCR byte. The X bit is not affected. When an X-bit-related interrupt occurs, both the X and I bits are automatically set by hardware after stacking the CCR. A return-from-interrupt instruction restores the X and I bits to their pre-interrupt request state.

5.5.3 Illegal Opcode Trap

Because not all possible opcodes or opcode sequences are defined, the MCU includes an illegal opcode detection circuit, which generates an interrupt request. When an illegal opcode is detected and the interrupt is recognized, the current value of the program counter is stacked. After interrupt service is complete, reinitialize the stack pointer so repeated execution of illegal opcodes does not cause stack underflow. Left uninitialized, the illegal opcode vector can point to a memory location that contains an illegal opcode. This condition causes an infinite loop that causes stack underflow. The stack grows until the system crashes.

The illegal opcode trap mechanism works for all unimplemented opcodes on all four opcode map pages. The address stacked as the return address for the illegal opcode interrupt is the address of the first byte of the illegal opcode. Otherwise, it would be almost impossible to determine whether the illegal opcode had been one or two bytes. The stacked return address can be used as a pointer to the illegal opcode so the illegal opcode service routine can evaluate the offending opcode.

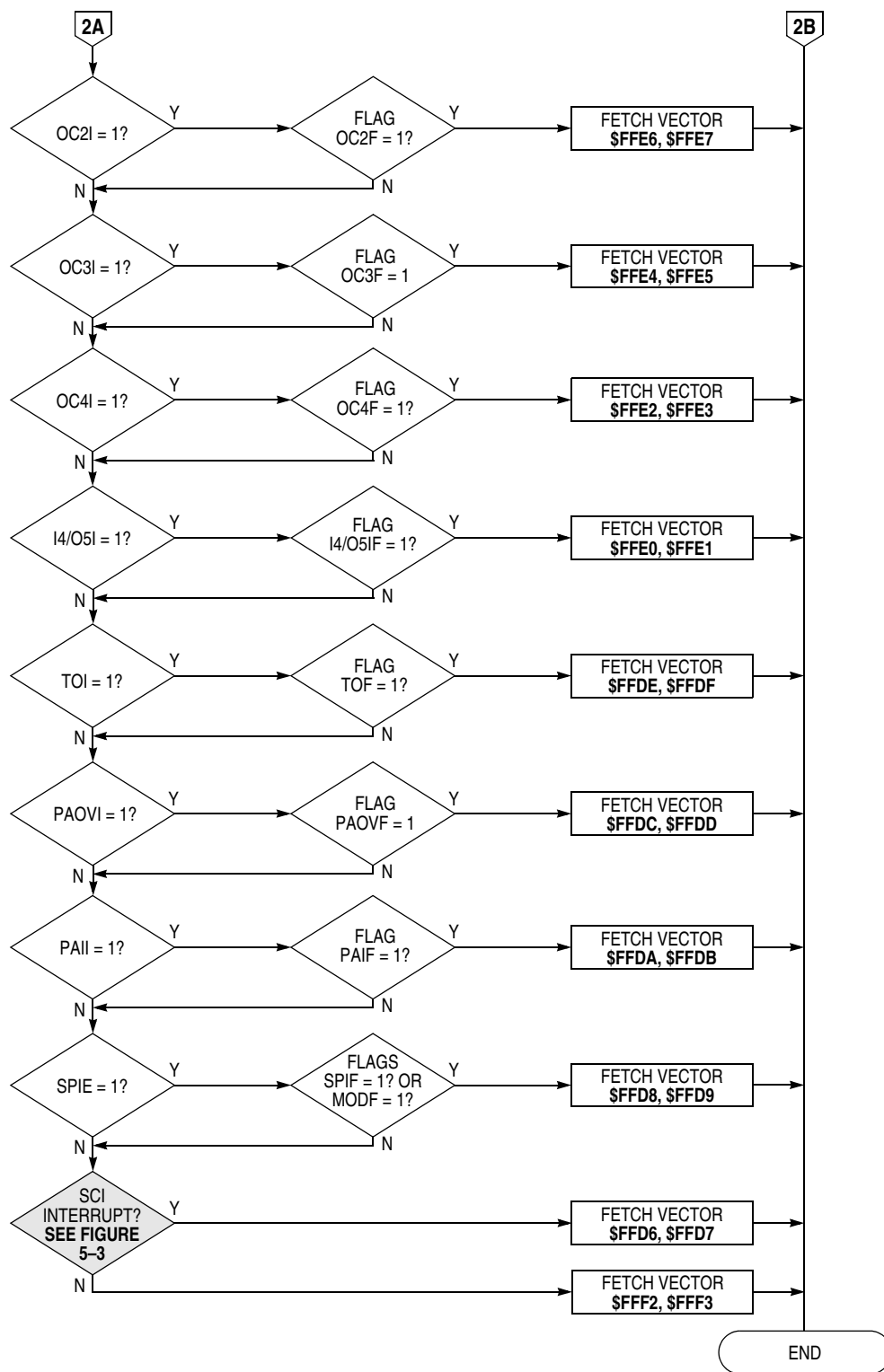


Figure 5-6. Interrupt Priority Resolution (Sheet 2 of 2)

TDRE and TC flags are normally set when the transmitter is first enabled (TE set to 1). The TDRE flag indicates there is room in the transmit queue to store another data character in the TDR. The TIE bit is the local interrupt mask for TDRE. When TIE is 0, TDRE must be polled. When TIE and TDRE are 1, an interrupt is requested.

The TC flag indicates the transmitter has completed the queue. The TCIE bit is the local interrupt mask for TC. When TCIE is 0, TC must be polled. When TCIE is 1 and TC is 1, an interrupt is requested.

Writing a 0 to TE requests that the transmitter stop when it can. The transmitter completes any transmission in progress before actually shutting down. Only an MCU reset can cause the transmitter to stop and shut down immediately. If TE is written to 0 when the transmitter is already idle, the pin reverts to its general-purpose I/O function (synchronized to the bit-rate clock). If anything is being transmitted when TE is written to 0, that character is completed before the pin reverts to general-purpose I/O, but any other characters waiting in the transmit queue are lost. The TC and TDRE flags are set at the completion of this last character, even though TE has been disabled.

7.9 Receiver Flags

The SCI receiver has five status flags, three of which can generate interrupt requests. The status flags are set by the SCI logic in response to specific conditions in the receiver. These flags can be read (polled) at any time by software. Refer to [Figure 7-10](#), which shows SCI interrupt arbitration.

When an overrun takes place, the new character is lost, and the character that was in its way in the parallel RDR is undisturbed. RDRF is set when a character has been received and transferred into the parallel RDR. The OR flag is set instead of RDRF if overrun occurs. A new character is ready to be transferred into RDR before a previous character is read from RDR.

The NF and FE flags provide additional information about the character in the RDR, but do not generate interrupt requests.

The last receiver status flag and interrupt source come from the IDLE flag. The RxD line is idle if it has constantly been at logic 1 for a full character time. The IDLE flag is set only after the RxD line has been busy and becomes idle, which prevents repeated interrupts for the whole time RxD remains idle.

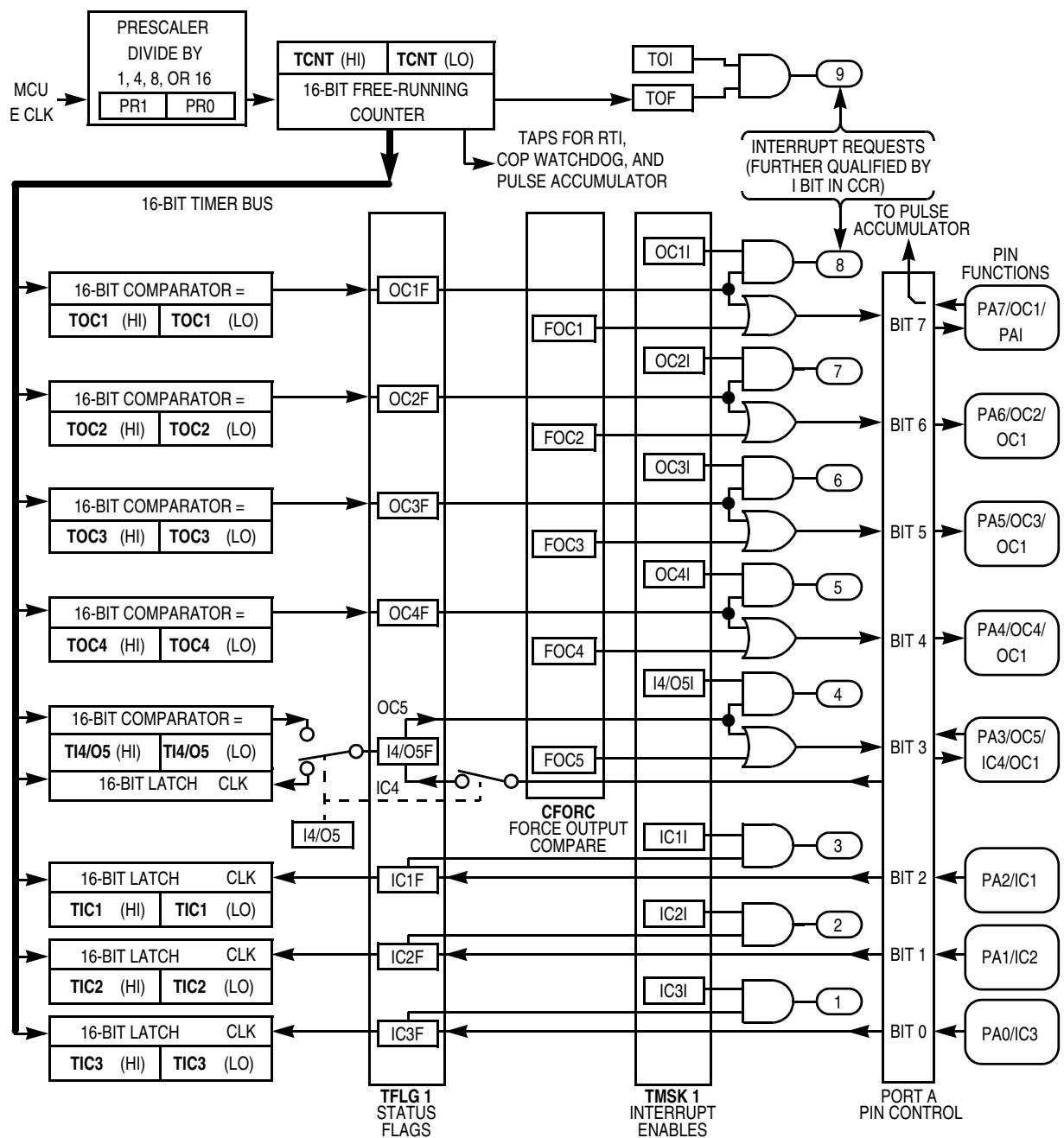


Figure 9-2. Capture/Compare Block Diagram

9.4.5 Timer Counter Register

The 16-bit read-only TCNT register contains the prescaled value of the 16-bit timer. A full counter read addresses the most significant byte (MSB) first. A read of this address causes the least significant byte (LSB) to be latched into a buffer for the next CPU cycle so that a double-byte read returns the full 16-bit state of the counter at the time of the MSB read cycle.



Figure 9-15. Timer Counter Register (TCNT)

9.4.6 Timer Control Register 1

The bits of this register specify the action taken as a result of a successful OCx compare.

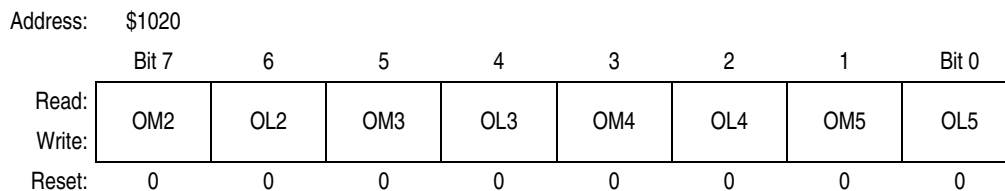


Figure 9-16. Timer Control Register 1 (TCTL1)

OM[2:5] — Output Mode Bits

OL[2:5] — Output Level Bits

These control bit pairs are encoded to specify the action taken after a successful OCx compare. OC5 functions only if the I4/O5 bit in the PACTL register is clear. Refer to [Table 9-3](#) for the coding.

Table 9-3. Timer Output Compare Actions

OMx	OLx	Action Taken on Successful Compare
0	0	Timer disconnected from output pin logic
0	1	Toggle OCx output line
1	0	Clear OCx output line to 0
1	1	Set OCx output line to 1

10.3 Functional Operating Range

Rating	Symbol	Value	Unit
Operating temperature range MC68HC(7)11Ex MC68HC(7)11ExC MC68HC(7)11ExV MC68HC(7)11ExM MC68HC811E2 MC68HC811E2C MC68HC811E2V MC68HC811E2M MC68L11Ex	T_A	T_L to T_H 0 to +70 –40 to +85 –40 to +105 –40 to +125 0 to +70 –40 to +85 –40 to +105 –40 to +125 –20 to +70	°C
Operating voltage range	V_{DD}	$5.0 \pm 10\%$	V

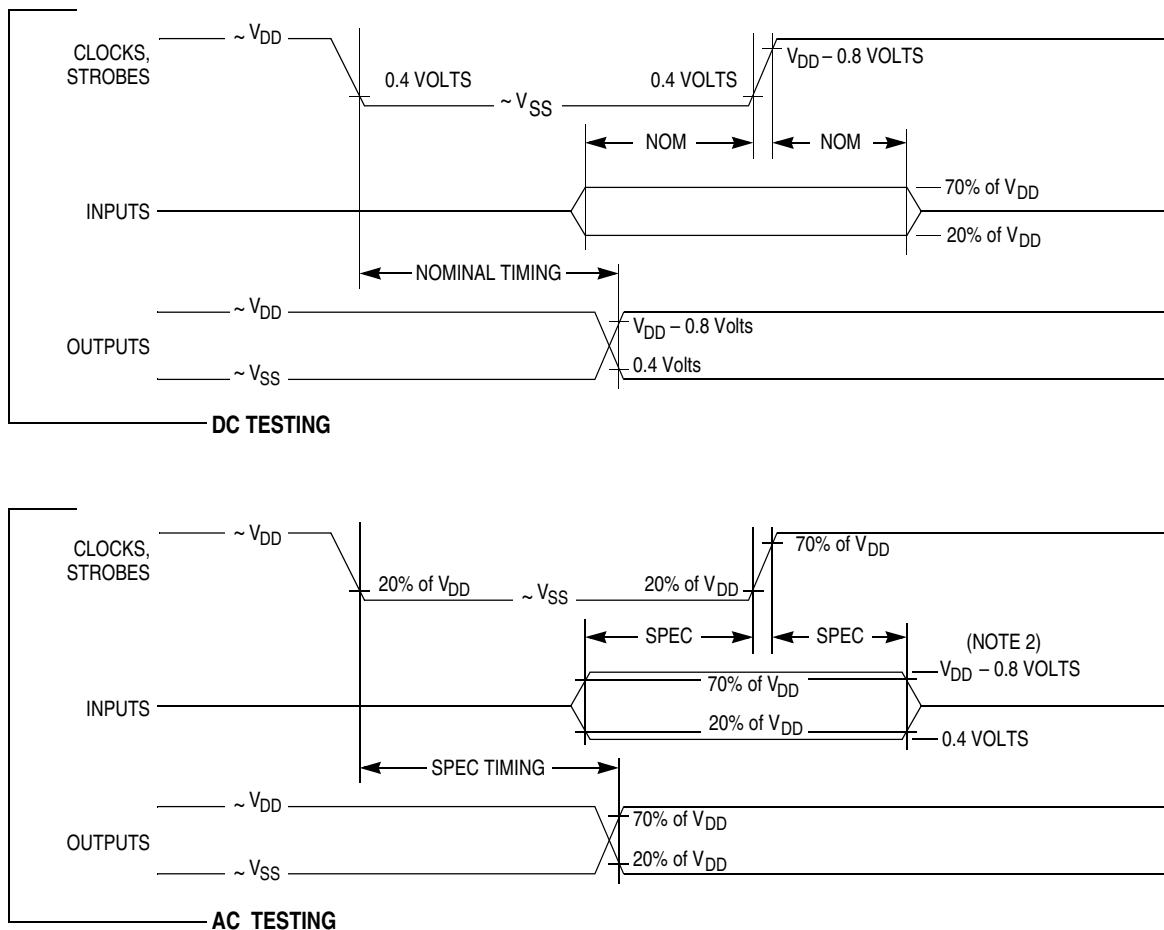
10.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Average junction temperature	T_J	$T_A + (P_D \times \Theta_{JA})$	°C
Ambient temperature	T_A	User-determined	°C
Package thermal resistance (junction-to-ambient) 48-pin plastic DIP (MC68HC811E2 only) 56-pin plastic SDIP 52-pin plastic leaded chip carrier 52-pin plastic thin quad flat pack (TQFP) 64-pin quad flat pack	Θ_{JA}	50 50 50 85 85	°C/W
Total power dissipation ⁽¹⁾	P_D	$\frac{P_{INT} + P_{I/O}}{K / T_J + 273^\circ\text{C}}$	W
Device internal power dissipation	P_{INT}	$I_{DD} \times V_{DD}$	W
I/O pin power dissipation ⁽²⁾	$P_{I/O}$	User-determined	W
A constant ⁽³⁾	K	$P_D \times (T_A + 273^\circ\text{C}) + \Theta_{JA} \times P_D^2$	W/°C

1. This is an approximate value, neglecting $P_{I/O}$.

2. For most applications, $P_{I/O} \leq P_{INT}$ and can be neglected.

3. K is a constant pertaining to the device. Solve for K with a known T_A and a measured P_D (at equilibrium). Use this value of K to solve for P_D and T_J iteratively for any value of T_A .



Notes:

1. Full test loads are applied during all dc electrical tests and ac timing measurements.
2. During ac timing measurements, inputs are driven to 0.4 volts and $V_{DD} - 0.8$ volts while timing measurements are taken at 20% and 70% of V_{DD} points.

Figure 10-1. Test Methods

10.19 EEPROM Characteristics

Characteristic ⁽¹⁾	Temperature Range			Unit
	–40 to 85°C	–40 to 105°C	–40 to 125°C	
Programming time ⁽²⁾ < 1.0 MHz, RCO enabled 1.0 to 2.0 MHz, RCO disabled ≥ 2.0 MHz (or anytime RCO enabled)	10 20 10	15 Must use RCO 15	20 Must use RCO 20	ms
Erase time ⁽²⁾ Byte, row, and bulk	10	10	10	ms
Write/erase endurance	10,000	10,000	10,000	Cycles
Data retention	10	10	10	Years

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

2. The RC oscillator (RCO) must be enabled (by setting the CSEL bit in the OPTION register) for EEPROM programming and erasure when the E-clock frequency is below 1.0 MHz.

10.20 MC68L11E9/E20 EEPROM Characteristics

Characteristic ⁽¹⁾	Temperature Range –20 to 70°C	Unit
Programming time ⁽²⁾ 3 V, $E \leq 2.0 \text{ MHz}$, RCO enabled 5 V, $E \leq 2.0 \text{ MHz}$, RCO enabled	25 10	ms
Erase time ⁽²⁾ (byte, row, and bulk) 3 V, $E \leq 2.0 \text{ MHz}$, RCO enabled 5 V, $E \leq 2.0 \text{ MHz}$, RCO enabled	25 10	ms
Write/erase endurance	10,000	Cycles
Data retention	10	Years

1. $V_{DD} = 3.0 \text{ Vdc}$ to 5.5 Vdc , $V_{SS} = 0 \text{ Vdc}$, $T_A = T_L$ to T_H

2. The RC oscillator (RCO) must be enabled (by setting the CSEL bit in the OPTION register) for EEPROM programming and erasure.

10.21 EPROM Characteristics

Characteristics ⁽¹⁾	Symbol	Min	Typ	Max	Unit
Programming voltage ⁽²⁾	V_{PPE}	11.75	12.25	12.75	V
Programming current ⁽³⁾	I_{PPE}	—	3	10	mA
Programming time	t_{EPROG}	2	2	4	ms

1. $V_{DD} = 5.0 \text{ Vdc} \pm 10\%$

2. During EPROM programming of the MC68HC711E9 device, the V_{PPE} pin circuitry may latch-up and be damaged if the input current is not limited to 10 mA. For more information please refer to MC68HC711E9 8-Bit Microcontroller Unit Mask Set Errata 3 (Freescale document order number 68HC711E9MSE3).

3. Typically, a 1-k Ω series resistor is sufficient to limit the programming current for the MC68HC711E9. A 100- Ω series resistor is sufficient to limit the programming current for the MC68HC711E20.

11.4 Extended Voltage Device Ordering Information (3.0 Vdc to 5.5 Vdc)

Description	Temperature	Frequency	MC Order Number
52-pin plastic leaded chip carrier (PLCC)			
Custom ROM	−20°C to +70°C	2 MHz	MC68L11E9FN2 MC68L11E20FN2
No ROM		2 MHz	MC68L11E1FN2
No ROM, no EEPROM		2 MHz	MC68L11E0FN2
64-pin quad flat pack (QFP)			
Custom ROM	−20°C to +70°C	2 MHz	MC68L11E9FU2 MC68L11E20FU2
No ROM		2 MHz	MC68L11E1FU2
No ROM, no EEPROM		2 MHz	MC68L11E0FU2
52-pin thin quad flat pack (10 mm x 10 mm)			
Custom ROM	−20°C to +70°C	2 MHz	MC68L11E9PB2
No ROM		2 MHz	MC68L11E1PB2
No ROM, no EEPROM		2 MHz	MC68L11E0PB2
56-pin dual in-line package with 0.70-inch lead spacing (SDIP)			
Custom ROM	−20°C to +70°C	2 MHz	MC68L11E9B2
No ROM		2 MHz	MC68L11E1B2
No ROM, no EEPROM		2 MHz	MC68L11E0B2



Basic Bootstrap Mode

This section describes only basic functions of the bootstrap mode. Other functions of the bootstrap mode are described in detail in the remainder of this application note.

When an M68HC11 is reset in bootstrap mode, the reset vector is fetched from a small internal read-only memory (ROM) called the bootstrap ROM or boot ROM. The firmware program in this boot ROM then controls the bootloading process, in this manner:

- First, the on-chip SCI (serial communications interface) is initialized. The first character received (\$FF) determines which of two possible baud rates should be used for the remaining characters in the download operation.
- Next, a binary program is received by the SCI system and is stored in RAM.
- Finally, a jump instruction is executed to pass control from the bootloader firmware to the user's loaded program.

Bootstrap mode is useful both at the component level and after the MCU has been embedded into a finished user system.

At the component level, Freescale uses bootstrap mode to control a monitored burn-in program for the on-chip electrically erasable programmable read-only memory (EEPROM). Units to be tested are loaded into special circuit boards that each hold many MCUs. These boards are then placed in burn-in ovens. Driver boards outside the ovens download an EEPROM exercise and diagnostic program to all MCUs in parallel. The MCUs under test independently exercise their internal EEPROM and monitor programming and erase operations. This technique could be utilized by an end user to load program information into the EPROM or EEPROM of an M68HC11 before it is installed into an end product. As in the burn-in setup, many M68HC11s can be gang programmed in parallel. This technique can also be used to program the EPROM of finished products after final assembly.

Freescale also uses bootstrap mode for programming target devices on the M68HC11 evaluation modules (EVM). Because bootstrap mode is a privileged mode like special test, the EEPROM-based configuration register (CONFIG) can be programmed using bootstrap mode on the EVM.

The greatest benefits from bootstrap mode are realized by designing the finished system so that bootstrap mode can be used after final assembly. The finished system need not be a single-chip mode application for the bootstrap mode to be useful because the expansion bus can be enabled after resetting the MCU in bootstrap mode. Allowing this capability requires almost no hardware or design cost and the addition of this capability is invisible in the end product until it is needed.

The ability to control the embedded processor through downloaded programs is achieved without the disassembly and chip-swapping usually associated with such control. This mode provides an easy way to load non-volatile memories such as EEPROM with calibration tables or to program the application firmware into a one-time programmable (OTP) MCU after final assembly.

Another powerful use of bootstrap mode in a finished assembly is for final test. Short programs can be downloaded to check parts of the system, including components and circuitry external to the embedded MCU. If any problems appear during product development, diagnostic programs can be downloaded to find the problems, and corrected routines can be downloaded and checked before incorporating them into the main application program.

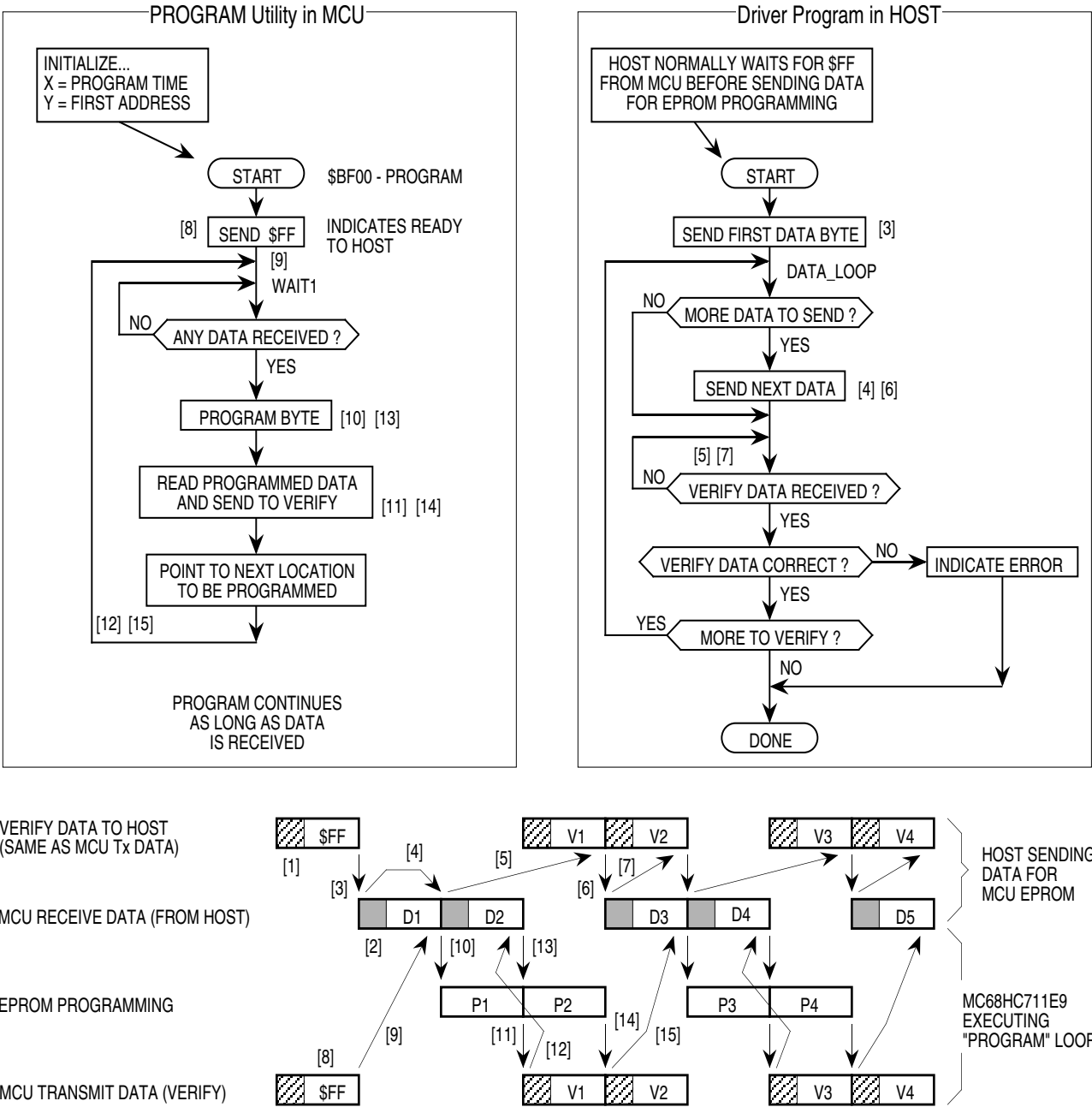


Figure 4. Host and MCU Activity during EPROM PROGRAM Utility

Operation

Configure the EVBU for boot mode operation by putting a jumper at J3. Ensure that the trace command jumper at J7 is not installed because this would connect the 12-V programming voltage to the OC5 output of the MCU.

Connect the EVBU to its dc power supply. When it is time to program the MCU EPROM, turn on the 12-volt programming power supply to the new circuitry in the wire-wrap area.

Connect the EVBU serial port to the appropriate serial port on the host system. For the Macintosh, this is the modem port with a modem cable. For the MS-DOS[®] computer, it is connected to COM1 with a straight through or modem cable. Power up the host system and start the BASIC program. If the program has not been compiled, this is accomplished from within the appropriate BASIC compiler or interpreter. Power up the EVBU.

Answer the prompt for filename with either a [RETURN] to accept the default shown or by typing in a new filename and pressing [RETURN].

The program will inform the user that it is working on converting the file from S records to binary. This process will take from 30 seconds to a few minutes, depending on the computer.

A prompt reading, "Comm port open?" will appear at the end of the file conversion. This is the last chance to ensure that everything is properly configured on the EVBU. Pressing [RETURN] will send the bootcode to the target MC68HC711E9. The program then informs the user that the bootload code is being sent to the target, and the results of the echoing of this code are displayed on the screen.

Another prompt reading "Programming is ready to begin. Are you?" will appear. Turn on the 12-volt programming power supply and press [RETURN] to start the actual programming of the target EPROM.

A count of the byte being verified will be updated continually on the screen as the programming progresses. Any failures will be flagged as they occur.

When programming is complete, a message will be displayed as well as a prompt requesting the user to press [RETURN] to quit.

Turn off the 12-volt programming power supply before turning off 5 volts to the EVBU.

[®] MS-DOS is a registered trademark of Microsoft Corporation in the United States and other countries.

```

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 '*****

```

