

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                      | 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) | 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               | <a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc11e1cfn3">https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc11e1cfn3</a> |

# Freescale Semiconductor, Inc.

Operating Modes and On-Chip Memory  
EPROM/OTPROM

## ELAT — EPROM/OTPROM Latch Control Bit

When ELAT = 1, writes to EPROM cause address and data to be latched and the EPROM/OTPROM cannot be read. ELAT can be read any time. ELAT can be written any time except when EPGM = 1; then the write to ELAT is disabled.

0 = EPROM address and data bus configured for normal reads

1 = EPROM address and data bus configured for programming

For the MC68HC711E9:

- EPGM enables the high voltage necessary for both EEPROM and EPROM/OTPROM programming.
- ELAT and EELAT are mutually exclusive and cannot both equal 1.

## BYTE — Byte/Other EEPROM Erase Mode Bit

Refer to [2.5 EEPROM](#).

## ROW — Row/All EEPROM Erase Mode Bit

Refer to [2.5 EEPROM](#).

## ERASE — Erase Mode Select Bit

Refer to [2.5 EEPROM](#).

## EELAT — EEPROM Latch Control Bit

Refer to [2.5 EEPROM](#).

## EPGM — EPROM/OTPROM/EEPROM Programming Voltage Enable Bit


EPGM can be read any time and can be written only when ELAT = 1 (for EPROM/OTPROM programming) or when EELAT = 1 (for EEPROM programming).

0 = Programming voltage to EPROM/OTPROM/EEPROM array disconnected

1 = Programming voltage to EPROM/OTPROM/EEPROM array connected

Address: \$1036

|        | Bit 7 | 6 | 5    | 4     | 3     | 2  | 1  | Bit 0 |
|--------|-------|---|------|-------|-------|----|----|-------|
| Read:  | MBE   |   | ELAT | EXCOL | EXROW | T1 | T0 | PGM   |
| Write: |       |   |      |       |       |    |    |       |
| Reset: | 0     | 0 | 0    | 0     | 0     | 0  | 0  | 0     |

 = Unimplemented

**Figure 2-15. MC68HC711E20 EPROM Programming Control Register (EPROG)**

## MBE — Multiple-Byte Programming Enable Bit

When multiple-byte programming is enabled, address bit 5 is considered a don't care so that bytes with address bit 5 = 0 and address bit 5 = 1 both get programmed. MBE can be read in any mode and always reads 0 in normal modes. MBE can be written only in special modes.

0 = EPROM array configured for normal programming

1 = Program two bytes with the same data

## Resets and Interrupts

5.2.2 External Reset ( $\overline{\text{RESET}}$ )

The CPU distinguishes between internal and external reset conditions by sensing whether the reset pin rises to a logic 1 in less than two E-clock cycles after an internal device releases reset. When a reset condition is sensed, the  $\overline{\text{RESET}}$  pin is driven low by an internal device for four E-clock cycles, then released. Two E-clock cycles later it is sampled. If the pin is still held low, the CPU assumes that an external reset has occurred. If the pin is high, it indicates that the reset was initiated internally by either the COP system or the clock monitor.

**CAUTION:** Do not connect an external resistor capacitor (RC) power-up delay circuit to the reset pin of M68HC11 devices because the circuit charge time constant can cause the device to misinterpret the type of reset that occurred.

## 5.2.3 Computer Operating Properly (COP) Reset

The MCU includes a COP system to help protect against software failures. When the COP is enabled, the software is responsible for keeping a free-running watchdog timer from timing out. When the software is no longer being executed in the intended sequence, a system reset is initiated.

The state of the NOCOP bit in the CONFIG register determines whether the COP system is enabled or disabled. To change the enable status of the COP system, change the contents of the CONFIG register and then perform a system reset. In the special test and bootstrap operating modes, the COP system is initially inhibited by the disable resets (DISR) control bit in the TEST1 register. The DISR bit can subsequently be written to 0 to enable COP resets.

The COP timer rate control bits CR[1:0] in the OPTION register determine the COP timeout period. The system E clock is divided by  $2^{15}$  and then further scaled by a factor shown in Table 5-1. After reset, these bits are 0, which selects the fastest timeout period. In normal operating modes, these bits can be written only once within 64 bus cycles after reset.

Table 5-1. COP Timer Rate Select

| CR[1:0] | Divide<br>E/ $2^{15}$ By | XTAL = 4.0 MHz<br>Timeout<br>– 0 ms, + 32.8 ms | XTAL = 8.0 MHz<br>Timeout<br>– 0 ms, + 16.4 ms | XTAL = 12.0 MHz<br>Timeout<br>– 0 ms, + 10.9 ms | XTAL = 16.0 MHz<br>Timeout<br>– 0 ms, + 8.2 ms |
|---------|--------------------------|--|--|---|--|
| 0 0     | 1                        | 32.768 ms                                      | 16.384 ms                                      | 10.923 ms                                       | 8.19 ms  |
| 0 1     | 4                        | 131.072 ms                                     | 65.536 ms                                      | 43.691 ms                                       | 32.8 ms  |
| 1 0     | 16                       | 524.28 ms                                      | 262.14 ms                                      | 174.76 ms                                       | 131 ms   |
| 1 1     | 64                       | 2.098 s  | 1.049 s  | 699.05 ms                                       | 524 ms   |
| E =     |                          | 1.0 MHz  | 2.0 MHz  | 3.0 MHz   | 4.0 MHz  |

## Resets and Interrupts

## 5.4 Reset and Interrupt Priority

Resets and interrupts have a hardware priority that determines which reset or interrupt is serviced first when simultaneous requests occur. Any maskable interrupt can be given priority over other maskable interrupts.

The first six interrupt sources are not maskable. The priority arrangement for these sources is:

1. POR or  $\overline{\text{RESET}}$  pin
2. Clock monitor reset
3. COP watchdog reset
4.  $\overline{\text{XIRQ}}$  interrupt
5. Illegal opcode interrupt
6. Software interrupt (SWI)

The maskable interrupt sources have this priority arrangement:

1.  $\overline{\text{IRQ}}$
2. Real-time interrupt
3. Timer input capture 1
4. Timer input capture 2
5. Timer input capture 3
6. Timer output compare 1
7. Timer output compare 2
8. Timer output compare 3
9. Timer output compare 4
10. Timer input capture 4/output compare 5
11. Timer overflow
12. Pulse accumulator overflow
13. Pulse accumulator input edge
14. SPI transfer complete
15. SCI system (refer to [Figure 5-7](#))

Any one of these interrupts can be assigned the highest maskable interrupt priority by writing the appropriate value to the PSEL bits in the HPRIO register. Otherwise, the priority arrangement remains the same. An interrupt that is assigned highest priority is still subject to global masking by the I bit in the CCR, or by any associated local bits. Interrupt vectors are not affected by priority assignment. To avoid race conditions, HPRIO can be written only while I-bit interrupts are inhibited.

## Resets and Interrupts

## 5.5.1 Interrupt Recognition and Register Stacking

An interrupt can be recognized at any time after it is enabled by its local mask, if any, and by the global mask bit in the CCR. Once an interrupt source is recognized, the CPU responds at the completion of the instruction being executed. Interrupt latency varies according to the number of cycles required to complete the current instruction. When the CPU begins to service an interrupt, the contents of the CPU registers are pushed onto the stack in the order shown in [Table 5-5](#). After the CCR value is stacked, the I bit and the X bit, if  $\overline{\text{XIRQ}}$  is pending, are set to inhibit further interrupts. The interrupt vector for the highest priority pending source is fetched and execution continues at the address specified by the vector. At the 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 [Section 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 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

## 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:                           | PB7    | PB6    | PB5    | PB4    | PB3    | PB2    | PB1   | PB0   |
| Write:                          |        |        |        |        |        |        |       |       |
| Reset:                          | 0      | 0      | 0      | 0      | 0      | 0      | 0     | 0     |
| Expanded or special test modes: |        |        |        |        |        |        |       |       |
| Read:                           | ADDR15 | ADDR14 | ADDR13 | ADDR12 | ADDR11 | ADDR10 | ADDR9 | ADDR8 |
| Write:                          |        |        |        |        |        |        |       |       |
| 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 | ADDR6 | ADDR5 | ADDR4 | ADDR3 | ADDR2 | ADDR1 | ADDR0 |
| DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 |

Reset: Indeterminate after reset

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

Register name: Timer Input Capture 1 Register (High) Address: \$1010

|        | 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: Indeterminate after reset

Register name: Timer Input Capture 1 Register (Low) Address: \$1011

|        | 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: Indeterminate after reset

**Figure 9-4. Timer Input Capture 1 Register Pair (TIC1)**

Register name: Timer Input Capture 2 Register (High) Address: \$1012

|        | 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: Indeterminate after reset

Register name: Timer Input Capture 2 Register (Low) Address: \$1013

|        | 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: Indeterminate after reset

**Figure 9-5. Timer Input Capture 2 Register Pair (TIC2)**

Register name: Timer Input Capture 3 Register (High) Address: \$1014

|        | 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: Indeterminate after reset

Register name: Timer Input Capture 3 Register (Low) Address: \$1015

|        | 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: Indeterminate after reset

**Figure 9-6. Timer Input Capture 3 Register Pair (TIC3)**

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.

|  |                 |        |        |                 |        |        |       |       |
|--|-----------------|--------|--------|-----------------|--------|--------|-------|-------|
| Register name: Timer Counter Register (High) |                 |        |        | Address: \$100E |        |        |       |       |
|  | 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:                                       |                 |        |        |                 |        |        |       |       |
| Reset:                                       | 0               | 0      | 0      | 0               | 0      | 0      | 0     | 0     |
| Register name: Timer Counter Register (Low)  |                 |        |        | Address: \$100F |        |        |       |       |
|  | 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:                                       |                 |        |        |                 |        |        |       |       |
| Reset:                                       | 0               | 0      | 0      | 0               | 0      | 0      | 0     | 0     |
|  | = Unimplemented |        |        |                 |        |        |       |       |

**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.

Address: \$1020

|        |       |     |     |     |     |     |     |       |
|--------|-------|-----|-----|-----|-----|-----|-----|-------|
|        | Bit 7 | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
| Read:  | OM2   | OL2 | OM3 | OL3 | OM4 | OL4 | OM5 | OL5   |
| Write: |       |     |     |     |     |     |     |       |
| Reset: | 0     | 0   | 0   | 0   | 0   | 0   | 0   | 0     |

**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                 |




The clock source for the RTI function is a free-running clock that cannot be stopped or interrupted except by reset. This clock causes the time between successive RTI timeouts to be a constant that is independent of the software latencies associated with flag clearing and service. For this reason, an RTI period starts from the previous timeout, not from when RTIF is cleared.

Every timeout causes the RTIF bit in TFLG2 to be set, and if RTII is set, an interrupt request is generated. After reset, one entire RTI period elapses before the RTIF is set for the first time. Refer to the [9.4.9 Timer Interrupt Mask 2 Register](#), [9.5.2 Timer Interrupt Flag Register 2](#), and [9.5.3 Pulse Accumulator Control Register](#).

## 9.5.1 Timer Interrupt Mask Register 2

This register contains the real-time interrupt enable bits.

|          |        |     |       |      |   |   |     |       |
|----------|--------|-----|-------|------|---|---|-----|-------|
| Address: | \$1024 |     |       |      |   |   |     |       |
|          | Bit 7  | 6   | 5     | 4    | 3 | 2 | 1   | Bit 0 |
| Read:    | TOI    | RTI | PAOVI | PAII |   |   | PR1 | PR0   |
| Write:   |        |     |       |      |   |   |     |       |
| Reset:   | 0      | 0   | 0     | 0    | 0 | 0 | 0   | 0     |

 = Unimplemented

**Figure 9-21. Timer Interrupt Mask 2 Register (TMSK2)**

TOI — Timer Overflow Interrupt Enable Bit

0 = TOF interrupts disabled

1 = Interrupt requested when TOF is set to 1

RTII — Real-Time Interrupt Enable Bit

0 = RTIF interrupts disabled

1 = Interrupt requested when RTIF set to 1

PAOVI — Pulse Accumulator Overflow Interrupt Enable Bit

Refer to [9.7 Pulse Accumulator](#).

PAII — Pulse Accumulator Input Edge Bit

Refer to [9.7 Pulse Accumulator](#).

Bits [3:2] — Unimplemented

Always read 0

PR[1:0] — Timer Prescaler Select Bits

Refer to [Table 9-4](#).

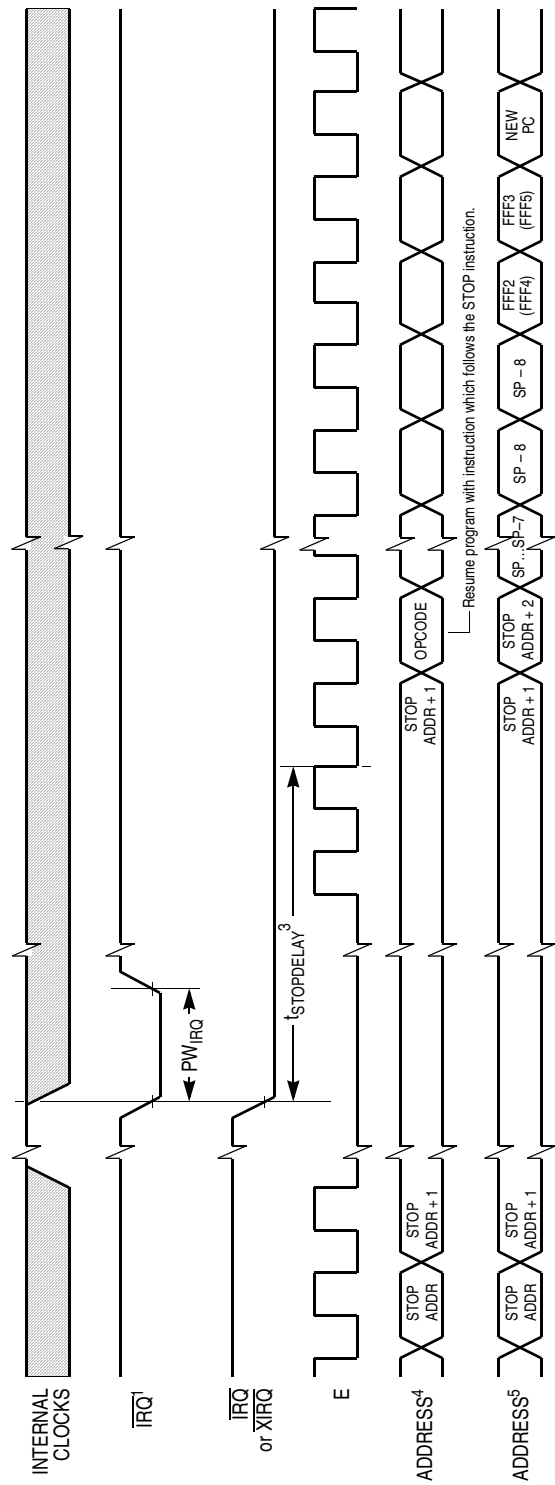
**NOTE:** Bits in TMSK2 correspond bit for bit with flag bits in TFLG2. Bits in TMSK2 enable the corresponding interrupt sources.

## PAOVI and PAOVF — Pulse Accumulator Interrupt Enable and Overflow Flag

The PAOVF status bit is set each time the pulse accumulator count rolls over from \$FF to \$00. To clear this status bit, write a 1 in the corresponding data bit position (bit 5) of the TFLG2 register. The PAOVI control bit allows configuring the pulse accumulator overflow for polled or interrupt-driven operation and does not affect the state of PAOVF. When PAOVI is 0, pulse accumulator overflow interrupts are inhibited, and the system operates in a polled mode, which requires that PAOVF be polled by user software to determine when an overflow has occurred. When the PAOVI control bit is set, a hardware interrupt request is generated each time PAOVF is set. Before leaving the interrupt service routine, software must clear PAOVF by writing to the TFLG2 register.

## PAII 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 PAII 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 PAII 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 PAII 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.  $\overline{IRQ}$  or  $\overline{XIRQ}$  with X bit in CCR = 0).

Figure 10-4. STOP Recovery Timing Diagram

# Electrical Characteristics

## 10.13 Analog-to-Digital Converter Characteristics

| Characteristic <sup>(1)</sup> | Parameter <sup>(2)</sup>   | Min                  | Absolute   | 2.0 MHz                                      | 3.0 MHz                                      | Unit                |
|-------------------------------|--|----------------------|------------|--|--|---------------------|
|                               |  |                      |            | Max  | Max  |                     |
| Resolution                    | Number of bits resolved by A/D converter   | —                    | 8          | —  | —  | Bits                |
| Non-linearity                 | Maximum deviation from the ideal A/D transfer characteristics  | —                    | —          | ±1/2   | ±1   | LSB                 |
| Zero error                    | Difference between the output of an ideal and an actual for 0 input voltage  | —                    | —          | ±1/2   | ±1   | LSB                 |
| Full scale error              | Difference between the output of an ideal and an actual A/D for full-scale input voltage   | —                    | —          | ±1/2   | ±1   | LSB                 |
| Total unadjusted error        | Maximum sum of non-linearity, zero error, and full-scale error   | —                    | —          | ±1/2   | ±1/2   | LSB                 |
| Quantization error            | Uncertainty because of converter resolution  | —                    | —          | ±1/2   | ±1/2   | LSB                 |
| Absolute accuracy             | Difference between the actual input voltage and the full-scale weighted equivalent of the binary output code, all error sources included | —                    | —          | ±1   | ±2   | LSB                 |
| Conversion range              | Analog input voltage range   | V <sub>RL</sub>      | —          | V <sub>RH</sub>                              | V <sub>RH</sub>                              | V                   |
| V <sub>RH</sub>               | Maximum analog reference voltage <sup>(3)</sup>  | V <sub>RL</sub>      | —          | V <sub>DD</sub> +0.1                         | V <sub>DD</sub> +0.1                         | V                   |
| V <sub>RL</sub>               | Minimum analog reference voltage <sup>(2)</sup>  | V <sub>SS</sub> -0.1 | —          | V <sub>RH</sub>                              | V <sub>RH</sub>                              | V                   |
| ΔV <sub>R</sub>               | Minimum difference between V <sub>RH</sub> and V <sub>RL</sub> <sup>(2)</sup>  | 3                    | —          | —  | —  | V                   |
| Conversion time               | Total time to perform a single A/D conversion:   | —                    | 32         | —  | —  | t <sub>CYC</sub> μs |
|                               | E clock<br>Internal RC oscillator  | —<br>—               | —<br>—     | t <sub>CYC</sub> +32<br>t <sub>CYC</sub> +32 | t <sub>CYC</sub> +32<br>t <sub>CYC</sub> +32 |                     |
| Monotonicity                  | Conversion result never decreases with an increase in input voltage; has no missing codes  | —                    | Guaranteed | —  | —  | —                   |
| Zero input reading            | Conversion result when V <sub>In</sub> = V <sub>RL</sub>   | 00                   | —          | —  | —  | Hex                 |
| Full scale reading            | Conversion result when V <sub>In</sub> = V <sub>RH</sub>   | —                    | —          | FF   | FF   | Hex                 |
| Sample acquisition time       | Analog input acquisition sampling time:<br>E clock<br>Internal RC oscillator   | —                    | 12         | —  | —  | t <sub>CYC</sub> μs |
|                               |  | —                    | —          | 12   | 12   |                     |
| Sample/hold capacitance       | Input capacitance during sample PE[7:0]  | —                    | 20 typical | —  | —  | pF                  |
| Input leakage                 | Input leakage on A/D pins<br>PE[7:0]<br>V <sub>RL</sub> , V <sub>RH</sub>  | —                    | —          | 400  | 400  | nA                  |
|                               |  | —                    | —          | 1.0  | 1.0  | μA                  |

1. V<sub>DD</sub> = 5.0 Vdc ±10%, V<sub>SS</sub> = 0 Vdc, T<sub>A</sub> = T<sub>L</sub> to T<sub>H</sub>, 750 kHz ≤ E ≤ 3.0 MHz, unless otherwise noted

2. Source impedances greater than 10 kΩ affect accuracy adversely because of input leakage.

3. Performance verified down to 2.5 V ΔV<sub>R</sub>, but accuracy is tested and guaranteed at ΔV<sub>R</sub> = 5 V ±10%.

**Ordering Information and Mechanical Specifications**

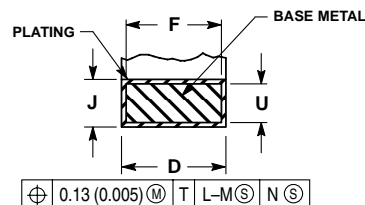
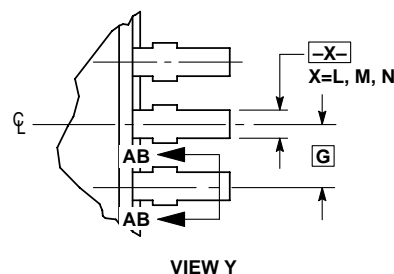
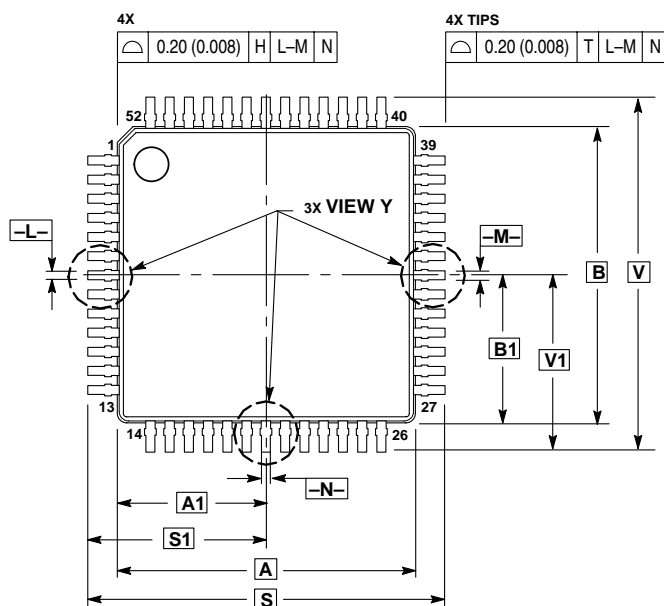
**11.3 Custom ROM Device Ordering Information**

| Description   | Temperature     | Frequency | MC Order Number |
|---|-----------------|-----------|-----------------|
| <b>52-pin plastic leaded chip carrier (PLCC)</b>                      |                 |           |                 |
| 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  |
| 20 Kbytes custom ROM  | 0°C to +70°C    | 3 MHz     | MC68HC11E20FN3  |
|   | -40°C to +85°C  | 2 MHz     | MC68HC11E20CFN2 |
|   |                 | 3 MHz     | MC68HC11E20CFN3 |
|   | -40°C to +105°C | 2 MHz     | MC68HC11E20VFN2 |
|   | -40°C to +125°C | 2 MHz     | MC68HC11E20MFN2 |
| <b>64-pin quad flat pack (QFP)</b>                                    |                 |           |                 |
| Custom ROM  | 0°C to +70°C    | 3 MHz     | MC68HC11E9FU3   |
|   | -40°C to +85°C  | 2 MHz     | MC68HC11E9CFU2  |
|   |                 | 3 MHz     | MC68HC11E9CFU3  |
|   | -40°C to +105°C | 2 MHz     | MC68HC11E9VFU2  |
|   | -40°C to +125°C | 2 MHz     | MC68HC11E9MFU2  |
| <b>64-pin quad flat pack (continued)</b>                              |                 |           |                 |
| 20 Kbytes Custom ROM  | 0°C to +70°C    | 3 MHz     | MC68HC11E20FU3  |
|   | -40°C to +85°C  | 2 MHz     | MC68HC11E20CFU2 |
|   |                 | 3 MHz     | MC68HC11E20CFU3 |
|   | -40°C to +105°C | 2 MHz     | MC68HC11E20VFU2 |
|   | -40°C to +125°C | 2 MHz     | MC68HC11E20MFU2 |
| <b>52-pin thin quad flat pack (10 mm x 10 mm)</b>                     |                 |           |                 |
| Custom ROM  | 0°C to +70°C    | 3 MHz     | MC68HC11E9PB3   |
|   | -40°C to +85°C  | 2 MHz     | MC68HC11E9CPB2  |
|   |                 | 3 MHz     | MC68HC11E9CPB3  |
|   | -40°C to +105°C | 2 MHz     | MC68HC11E9VPB2  |
|   | -40°C to +125°C | 2 MHz     | MC68HC11E9MPB2  |
| <b>56-pin dual in-line package with 0.70-inch lead spacing (SDIP)</b> |                 |           |                 |
| Custom ROM  | 0°C to +70°C    | 3 MHz     | MC68HC11E9B3    |
|   | -40°C to +85°C  | 2 MHz     | MC68HC11E9CB2   |
|   |                 | 3 MHz     | MC68HC11E9CB3   |
|   | -40°C to +105°C | 2 MHz     | MC68HC11E9VB2   |
|   | -40°C to +125°C | 2 MHz     | MC68HC11E9MB2   |

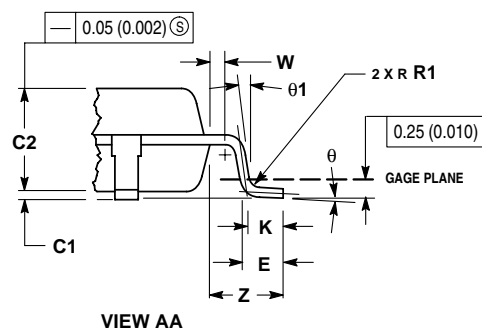
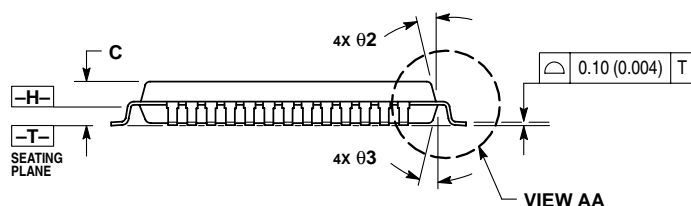
# Freescall Semiconductor, Inc.

Ordering Information and Mechanical Specifications  
52-Pin Thin Quad Flat Pack (Case 848D)

## 11.8 52-Pin Thin Quad Flat Pack (Case 848D)



SECTION AB-AB  
ROTATED 90° CLOCKWISE



- NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
  2. CONTROLLING DIMENSION: MILLIMETER.
  3. DATUM PLANE -H- IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
  4. DATUMS -L-, -M- AND -N- TO BE DETERMINED AT DATUM PLANE -H-.
  5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -T-.
  6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE -H-.
  7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED 0.46 (0.018). MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD OR PROTRUSION 0.07 (0.003).

| DIM | MILLIMETERS |      | INCHES    |       |
|-----|-------------|------|-----------|-------|
|     | MIN         | MAX  | MIN       | MAX   |
| A   | 10.00 BSC   |      | 0.394 BSC |       |
| A1  | 5.00 BSC    |      | 0.197 BSC |       |
| B   | 10.00 BSC   |      | 0.394 BSC |       |
| B1  | 5.00 BSC    |      | 0.197 BSC |       |
| C   | —           | 1.70 | —         | 0.067 |
| C1  | 0.05        | 0.20 | 0.002     | 0.008 |
| C2  | 1.30        | 1.50 | 0.051     | 0.059 |
| D   | 0.20        | 0.40 | 0.008     | 0.016 |
| E   | 0.45        | 0.75 | 0.018     | 0.030 |
| F   | 0.22        | 0.35 | 0.009     | 0.014 |
| G   | 0.65 BSC    |      | 0.026 BSC |       |
| J   | 0.07        | 0.20 | 0.003     | 0.008 |
| K   | 0.50 REF    |      | 0.020 REF |       |
| R1  | 0.08        | 0.20 | 0.003     | 0.008 |
| S   | 12.00 BSC   |      | 0.472 BSC |       |
| S1  | 6.00 BSC    |      | 0.236 BSC |       |
| U   | 0.09        | 0.16 | 0.004     | 0.006 |
| V   | 12.00 BSC   |      | 0.472 BSC |       |
| V1  | 6.00 BSC    |      | 0.236 BSC |       |
| W   | 0.20 REF    |      | 0.008 REF |       |
| Z   | 1.00 REF    |      | 0.039 REF |       |
| θ   | 0°          | 7°   | 0°        | 7°    |
| θ1  | 0°          | —    | 0°        | —     |
| θ2  | 12° REF     |      | 12° REF   |       |
| θ3  | 5°          | 13°  | 5°        | 13°   |

## UPLOAD Utility

---

The UPLOAD utility subroutine transfers data from the MCU to a host computer system over the SCI serial data link.

**NOTE:** *Only EPROM versions of the M68HC11 include this utility.*

Verification of EPROM contents is one example of how the UPLOAD utility could be used. Before calling this program, the Y index register is loaded (by user firmware) with the address of the first data byte to be uploaded. If a baud rate other than the current SCI baud rate is to be used for the upload process, the user's firmware must also write to the baud register. The UPLOAD program sends successive bytes of data out the SCI transmitter until a reset is issued (the upload loop is infinite).

For a complete commented listing example of the UPLOAD utility, refer to [Listing 3. MC68HC711E9 Bootloader ROM](#).

## EPROM Programming Utility

---

The EPROM programming utility is one way of programming data into the internal EPROM of the MC68HC711E9 MCU. An external 12-V programming power supply is required to program on-chip EPROM. The simplest way to use this utility program is to bootstrap a 3-byte program consisting of a single jump instruction to the start of the PROGRAM utility program (\$BF00). The bootloader program sets the X and Y index registers to default values before jumping to the downloaded program (see [16] at the bottom of [Figure 3](#)). When the host computer sees the \$FF character, data to be programmed into the EPROM is sent, starting with the character for location \$D000. After the last byte to be programmed is sent to the MC68HC711E9 and the corresponding verification data is returned to the host, the programming operation is terminated by resetting the MCU.

The number of bytes to be programmed, the first address to be programmed, and the programming time can be controlled by the user if values other than the default values are desired.

# Application Note

R14–R15. The PE7 input was chosen because the internal circuitry for port E pins can tolerate voltages slightly higher than  $V_{DD}$ ; therefore, resistors R14 and R15 are less critical. No data to be programmed is passed to the target MCU until the master MCU senses that  $V_{PP}$  has been stable for about 200 ms.

When  $V_{PP}$  is ready, the master MCU turns on the red LED (light-emitting diode) and begins passing data to the target MCU. **EPROM Programming Utility** explains the activity as data is sent from the master MCU to the target MCU and programmed into the EPROM of the target. The master MCU in the EVBU corresponds to the HOST in the programming utility description and the "PROGRAM utility in MCU" is running in the bootstrap ROM of the target MCU.

Each byte of data sent to the target is programmed and then the programmed location is read and sent back to the master for verification. If any byte fails, the red and green LEDs are turned off, and the programming operation is aborted. If the entire 12 Kbytes are programmed and verified successfully, the red LED is turned off, and the green LED is turned on to indicate success. The programming of all 12 Kbytes takes about 30 seconds.

After a programming operation, the  $V_{PP}$  switch (S2) should be turned off before the EVBU power is turned off.

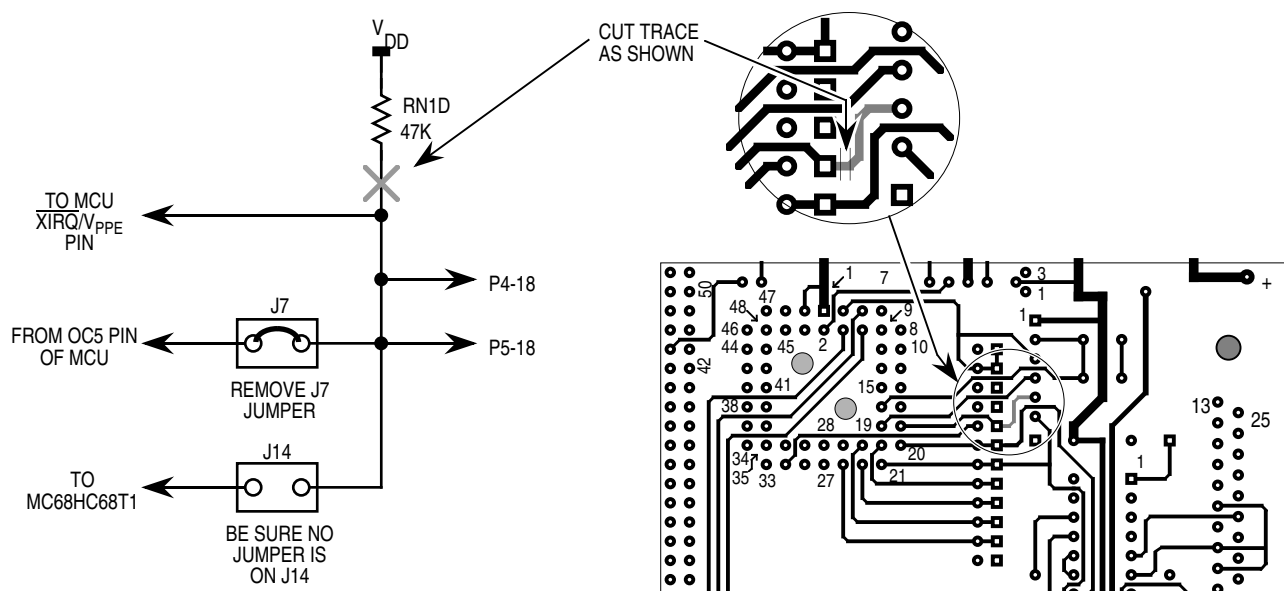


Figure 7. Isolating EVBU XIRQ Pin



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.

## Application Note

```

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

```

## To Execute the Program

---

Use this step-by-step procedure to program the MC68HC711E9 device.

### 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 the MC68HC711E9 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-V, +12 volts (at most +12.5 volts) to  $V_{PP}$ , and ground to GND on your programmer board's power connector, P1. The remaining TXD/PD1 and RXD/PD0 connections are not used in this procedure. They are for gang programming MC68HC711E9 devices, which is discussed in the M68HC711E9PGMR Manual. You cannot gang program with PCbug11.
- Ensure that the "remove for multi-programming" jumper, J1, below the +5-V power switch has a fabricated jumper installed.

### Step 2

Apply power to the programmer board by moving the +5-V switch to the ON position. From a DOS command line prompt, start PCbug11 this way:

```
C:\PCBUG11\ > PCBUG11 -E PORT = 1
with the E9PGMR connected to COM1
```

or

```
C:\PCBUG11\ > PCBUG11 -E PORT = 2
with the E9PGMR connected to COM2
```

PCbug11 only supports COM ports 1 and 2. If the proper connections are made and you have a high-quality cable, you should quickly get a

PCbug11 command prompt. If you do receive a Comms fault error, check the 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**

Clear the block protect register (BPROT) to allow programming of the MC68HC711E9 EEPROM.

At the PCbug11 command prompt, type: MS 1035 00.

**Step 5**

The CONFIG register defaults to hexadecimal 103F on the MC68HC711E9. PCBUG11 needs addressing parameters to allow programming of a specific block of memory so the following parameter must be given.

At the PCbug11 command prompt, type: EEPROM 0.

Then type: EEPROM 103F 103F.

**Step 6**

Erase the CONFIG to allow byte programming.

At the PCbug11 command prompt, type: EEPROM ERASE BULK 103F.

**Step 7**

You are now ready to download the program into the EEPROM and EPROM.

At the PCbug11 command prompt, type:  
LOADSC:\MYPROG\MYPROG.S19.

For more details on programming the EPROM, read the engineering bulletin *Programming MC68HC711E9 Devices with PCbug11 and the M68HC11EVB*, Motorola document number EB187.

## **Motorola Semiconductor Engineering Bulletin**

---

**EB188**

### **Enabling the Security Feature on M68HC811E2 Devices with PCbug11 on the M68HC711E9PGMR**

By Edgar Saenz  
Austin, Texas

#### **Introduction**

---

The PCbug11 software, needed along with the M68HC711E9PGMR to program MC68HC811E2 devices, is available from the download section of the Microcontroller Worldwide Web site <http://www.motorola.com/semiconductors/>.

Retrieve the file pcbug342.exe (a self-extracting archive) from the MCU11 directory.

Some Motorola evaluation board products also are shipped with PCbug11.

**NOTE:** *For specific information about any of the PCbug11 commands, see the appropriate sections in the PCbug11 User's Manual (part number M68PCBUG11/D2), which is available from the Motorola Literature Distribution Center, as well as the Worldwide Web at <http://www.motorola.com/semiconductors/>. The file is also on the software download system and is called pcbug11.pdf.*

