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	30
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	4.75V ~ 5.25V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.21x24.21)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mchc11f1cfne3

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

4.5 Chip Selects 4-18
4.5.1 Program Chip Select 4-18
4.5.2 I/O Chip Selects 4-18
4.5.3 General-Purpose Chip Select 4-19

SECTION 5 RESETS AND INTERRUPTS

5.1 Resets 5-1
5.1.1 Power-On Reset 5-1
5.1.2 External Reset ($\overline{\text{RESET}}$) 5-1
5.1.3 Computer Operating Properly (COP) Reset 5-2
5.1.4 Clock Monitor Reset 5-2
5.1.5 OPTION Register 5-3
5.1.6 CONFIG Register 5-4
5.2 Effects of Reset 5-4
5.2.1 Central Processing Unit 5-5
5.2.2 Memory Map 5-5
5.2.3 Parallel I/O 5-5
5.2.4 Timer 5-5
5.2.5 Real-Time Interrupt (RTI) 5-5
5.2.6 Pulse Accumulator 5-6
5.2.7 Computer Operating Properly (COP) 5-6
5.2.8 Serial Communications Interface (SCI) 5-6
5.2.9 Serial Peripheral Interface (SPI) 5-6
5.2.10 Analog-to-Digital Converter 5-6
5.2.11 System 5-6
5.3 Reset and Interrupt Priority 5-6
5.3.1 Highest Priority Interrupt and Miscellaneous Register 5-7
5.4 Interrupts 5-8
5.4.1 Interrupt Recognition and Register Stacking 5-9
5.4.2 Non-Maskable Interrupt Request ($\overline{\text{XIRQ}}$) 5-10
5.4.3 Illegal Opcode Trap 5-10
5.4.4 Software Interrupt 5-11
5.4.5 Maskable Interrupts 5-11
5.4.6 Reset and Interrupt Processing 5-11
5.5 Low Power Operation 5-16
5.5.1 WAIT 5-17
5.5.2 STOP 5-17

SECTION 6 PARALLEL INPUT/OUTPUT

6.1 Port A 6-1

TABLE OF CONTENTS
(Continued)

Paragraph	Title	Page
6.2	Port B	6-2
6.3	Port C	6-2
6.4	Port D	6-3
6.5	Port E	6-4
6.6	Port F	6-4
6.7	Port G	6-5
6.8	System Configuration Options 2	6-5

SECTION 7 SERIAL COMMUNICATIONS INTERFACE

7.1	Data Format	7-1
7.2	Transmit Operation	7-1
7.3	Receive Operation	7-2
7.4	Wakeup Feature	7-4
7.4.1	Idle-Line Wakeup	7-4
7.4.2	Address-Mark Wakeup	7-4
7.5	SCI Error Detection	7-5
7.6	SCI Registers	7-5
7.6.1	Serial Communications Data Register	7-5
7.6.2	Serial Communications Control Register 1	7-5
7.6.3	Serial Communications Control Register 2	7-6
7.6.4	Serial Communication Status Register	7-7
7.6.5	Baud Rate Register	7-8
7.7	Status Flags and Interrupts	7-10
7.7.1	Receiver Flags	7-11

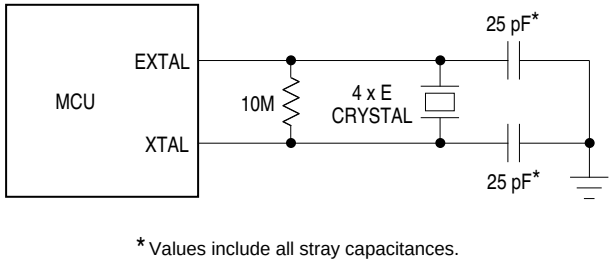
SECTION 8 SERIAL PERIPHERAL INTERFACE

8.1	Functional Description	8-1
8.2	SPI Transfer Formats	8-2
8.2.1	Clock Phase and Polarity Controls	8-3
8.3	SPI Signals	8-3
8.3.1	Master In Slave Out	8-4
8.3.2	Master Out Slave In	8-4
8.3.3	Serial Clock	8-4
8.3.4	Slave Select	8-4
8.4	SPI System Errors	8-4
8.5	SPI Registers	8-5
8.5.1	Serial Peripheral Control	8-5
8.5.2	Serial Peripheral Status	8-7
8.5.3	Serial Peripheral Data Register	8-7

The XTAL pin is normally left unterminated when an external CMOS compatible clock is connected to the EXTAL pin. However, a 10 kΩ to 100 kΩ load resistor connected from the XTAL output to ground can be used to reduce RFI noise emission.

The XTAL output is normally used to drive a crystal. The XTAL output can be buffered with a high-impedance buffer, or it can be used to drive the EXTAL input of another M68HC11 device. Refer to **Figure 2-6**.

In all cases, use caution when designing circuitry associated with the oscillator pins. Load capacitances shown in the oscillator circuits include all stray layout capacitances. Refer to **Figure 2-4**, **Figure 2-5**, and **Figure 2-6**.



*Values include all stray capacitances.

Figure 2-4 Common Crystal Connections

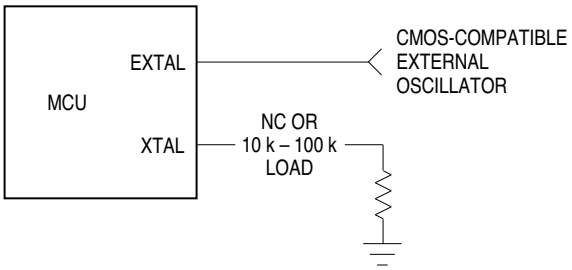
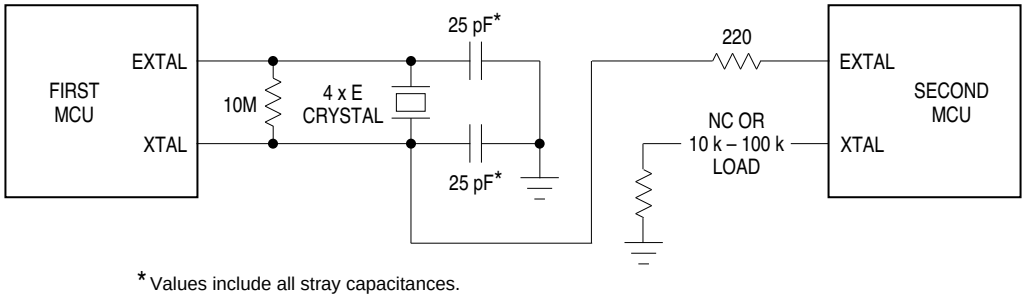


Figure 2-5 External Oscillator Connections



*Values include all stray capacitances.

Figure 2-6 One Crystal Driving Two MCUs

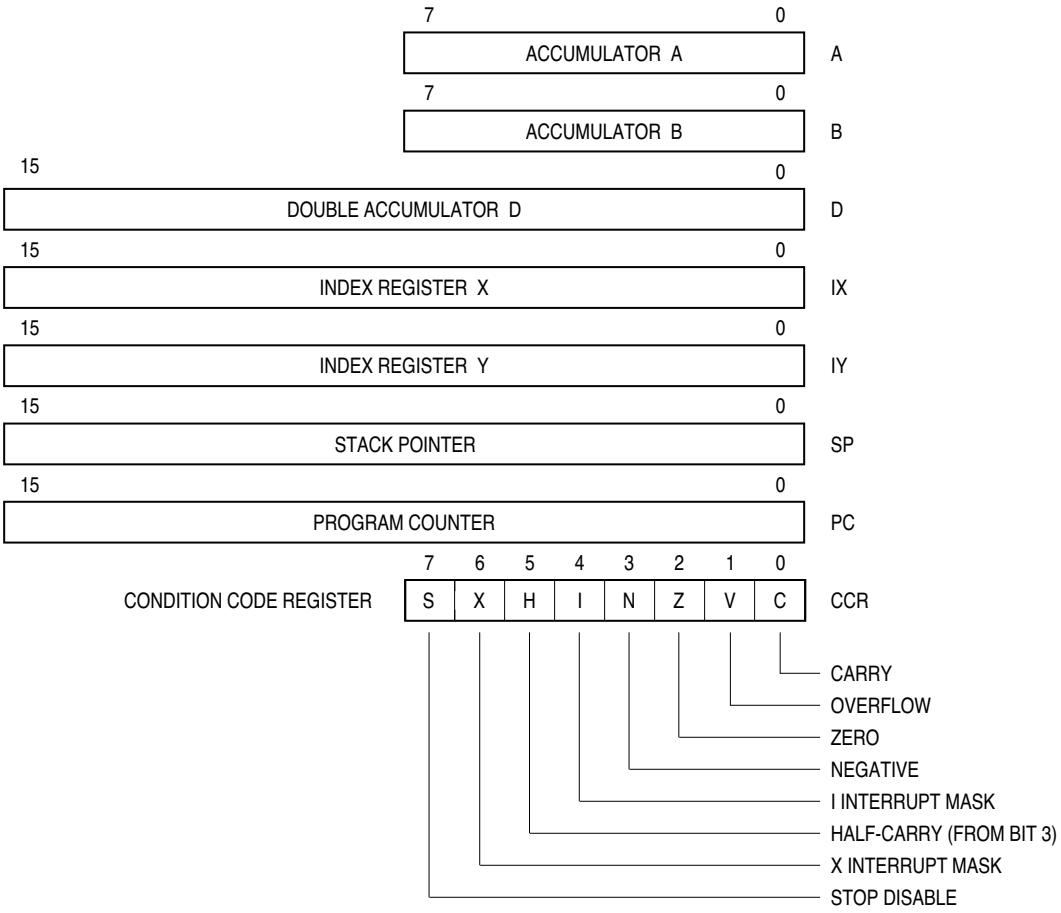


Figure 3-1 Programming Model

3.1.1 Accumulators A, B, and D

Accumulators A and B are general-purpose 8-bit registers that hold operands and results of arithmetic calculations or data manipulations. For some instructions, these two accumulators are treated as a single double-byte (16-bit) accumulator called accumulator D. Although most instructions can use accumulators A or B interchangeably, the following exceptions apply:

The ABX and ABY instructions add the contents of 8-bit accumulator B to the contents of 16-bit register X or Y, but there are no equivalent instructions that use A instead of B.

The TAP and TPA instructions transfer data from accumulator A to the condition code register, or from the condition code register to accumulator A, however, there are no equivalent instructions that use B rather than A.

The decimal adjust accumulator A (DAA) instruction is used after binary-coded decimal (BCD) arithmetic operations, but there is no equivalent BCD instruction to adjust accumulator B.

Table 3-2 Instruction Set (Sheet 2 of 6)

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
BGT (rel)	Branch if > Zero	? Z + (N ⊕ V) = 0	REL	2E	rr	3	—	—	—	—	—	—	—	—
BHI (rel)	Branch if Higher	? C + Z = 0	REL	22	rr	3	—	—	—	—	—	—	—	—
BHS (rel)	Branch if Higher or Same	? C = 0	REL	24	rr	3	—	—	—	—	—	—	—	—
BITA (opr)	Bit(s) Test A with Memory	A • M	A IMM A DIR A EXT A IND,X A IND,Y	85 95 B5 A5 18 A5	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
BITB (opr)	Bit(s) Test B with Memory	B • M	B IMM B DIR B EXT B IND,X B IND,Y	C5 D5 F5 E5 18 E5	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
BLE (rel)	Branch if Δ Zero	? Z + (N ⊕ V) = 1	REL	2F	rr	3	—	—	—	—	—	—	—	—
BLO (rel)	Branch if Lower	? C = 1	REL	25	rr	3	—	—	—	—	—	—	—	—
BLS (rel)	Branch if Lower or Same	? C + Z = 1	REL	23	rr	3	—	—	—	—	—	—	—	—
BLT (rel)	Branch if < Zero	? N ⊕ V = 1	REL	2D	rr	3	—	—	—	—	—	—	—	—
BMI (rel)	Branch if Minus	? N = 1	REL	2B	rr	3	—	—	—	—	—	—	—	—
BNE (rel)	Branch if not = Zero	? Z = 0	REL	26	rr	3	—	—	—	—	—	—	—	—
BPL (rel)	Branch if Plus	? N = 0	REL	2A	rr	3	—	—	—	—	—	—	—	—
BRA (rel)	Branch Always	? 1 = 1	REL	20	rr	3	—	—	—	—	—	—	—	—
BRCLR(opr) (msk) (rel)	Branch if Bit(s) Clear	? M • mm = 0	DIR IND,X IND,Y	13 1F 18 1F	dd mm rr ff mm rr ff mm rr	6 7 8	—	—	—	—	—	—	—	—
BRN (rel)	Branch Never	? 1 = 0	REL	21	rr	3	—	—	—	—	—	—	—	—
BRSET(opr) (msk) (rel)	Branch if Bit(s) Set	? (M) • mm = 0	DIR IND,X IND,Y	12 1E 18 1E	dd mm rr ff mm rr ff mm rr	6 7 8	—	—	—	—	—	—	—	—
BSET (opr) (msk)	Set Bit(s)	M + mm ⇒ M	DIR IND,X IND,Y	14 1C 18 1C	dd mm ff mm ff mm	6 7 8	—	—	—	—	Δ	Δ	0	—
BSR (rel)	Branch to Subroutine	See Figure 3–2	REL	8D	rr	6	—	—	—	—	—	—	—	—
BVC (rel)	Branch if Overflow Clear	? V = 0	REL	28	rr	3	—	—	—	—	—	—	—	—
BVS (rel)	Branch if Overflow Set	? V = 1	REL	29	rr	3	—	—	—	—	—	—	—	—
CBA	Compare A to B	A – B	INH	11	—	2	—	—	—	—	Δ	Δ	Δ	Δ
CLC	Clear Carry Bit	0 ⇒ C	INH	0C	—	2	—	—	—	—	—	—	—	0
CLI	Clear Interrupt Mask	0 ⇒ I	INH	0E	—	2	—	—	—	0	—	—	—	—
CLR (opr)	Clear Memory Byte	0 ⇒ M	EXT IND,X IND,Y	7F 6F 18 6F	hh ll ff ff	6 6 7	—	—	—	—	0	1	0	0
CLRA	Clear Accumulator A	0 ⇒ A	A INH	4F	—	2	—	—	—	—	0	1	0	0
CLRB	Clear Accumulator B	0 ⇒ B	B INH	5F	—	2	—	—	—	—	0	1	0	0
CLV	Clear Overflow Flag	0 ⇒ V	INH	0A	—	2	—	—	—	—	—	—	0	—
CMPA (opr)	Compare A to Memory	A – M	A IMM A DIR A EXT A IND,X A IND,Y	81 91 B1 A1 18 A1	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	Δ	Δ
CMPB (opr)	Compare B to Memory	B – M	B IMM B DIR B EXT B IND,X B IND,Y	C1 D1 F1 E1 18 E1	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	Δ	Δ

SECTION 4OPERATING MODES AND ON-CHIP MEMORY

This section contains information about the modes that define MC68HC11F1 operating conditions, and about the on-chip memory that allows the MCU to be configured for various applications.

4.1 Operating Modes

The values of the mode select inputs MODB and MODA during reset determine the operating mode. Single chip and expanded modes are the normal modes. In single-chip mode only on-board resources are available. Expanded mode, however, allows access to external memory or peripheral devices. Each of these two normal modes is paired with a special mode. Bootstrap mode, a variation of the single-chip mode, executes a bootloader program from an internal bootstrap ROM. Test mode allows privileged access to internal resources.

4.1.1 Single-Chip Operating Mode

In single-chip operating mode, the MC68HC11F1 has no external address or data bus. Ports B, C, and F are available for general-purpose I/O.

4.1.2 Expanded Operating Mode

In expanded operating mode, the MCU can access a 64-Kbyte physical address space. The address space includes the same on-chip memory addresses used for single-chip mode, in addition to external memory and peripheral devices.

The expansion bus is made up of ports B, C, F and the R/$\overline{W}$ signal. In expanded mode, high order address bits are output on the port B pins, low order address bits on the port F pins, and the data bus on port C. The R/$\overline{W}$ pin indicates the direction of data transfer on the port C bus.

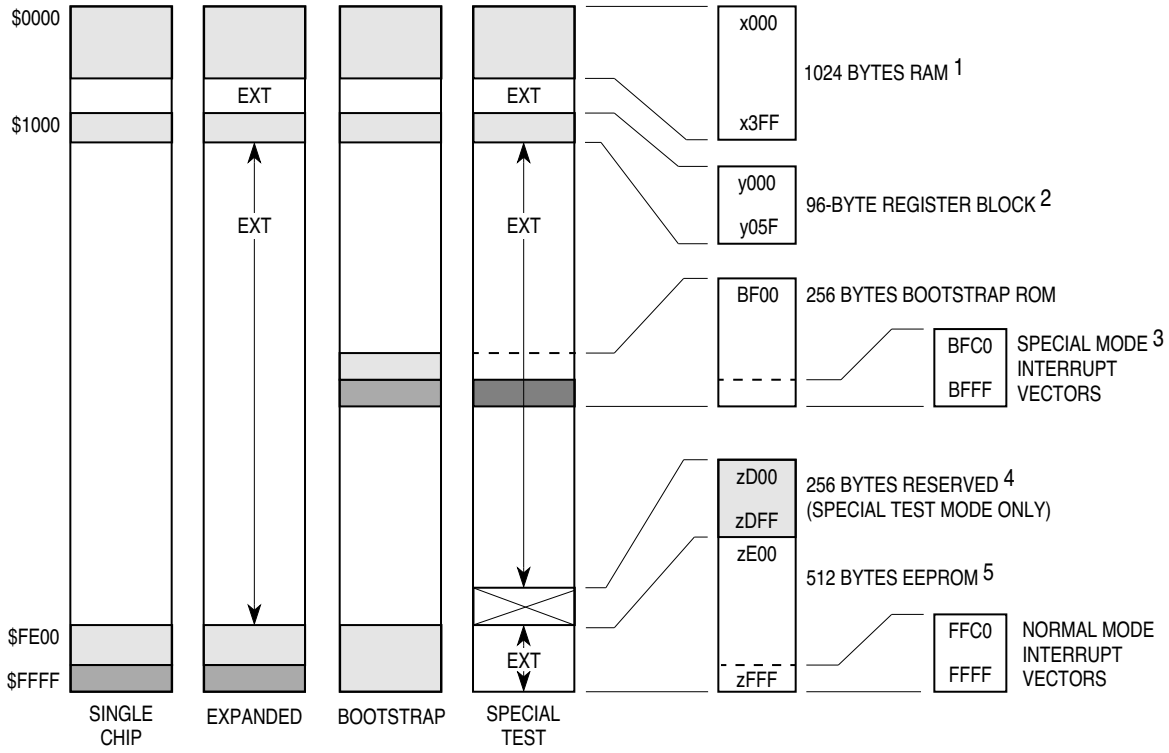
4.1.3 Special Test Mode

Special test mode, a variation of the expanded mode, is primarily used during Motorola's internal production testing; however, it is accessible for programming the CONFIG register, programming calibration data into EEPROM, and supporting emulation and debugging during development.

4.1.4 Special Bootstrap Mode

Bootstrap mode is a special variation of the single-chip mode. Bootstrap mode allows special-purpose programs to be entered into internal RAM. When boot mode is selected at reset, a small bootstrap ROM becomes present in the memory map. Reset and interrupt vectors are located in bootstrap ROM at $BFC0–$BFFF. The MCU fetches the reset vector, then executes the bootloader.

4.2.2 Memory Map



NOTES:
1. RAM can be remapped to any 4-Kbyte boundary ($x000). "x" represents the value contained in RAM[3:0] in the init register.
2. The register block can be remapped to any 4-Kbyte boundary ($y000). "y" represents the value contained in reg[3:0] in the init register.
3. Special test mode vectors are externally addressed.
4. In special test mode the address locations $zD00–$zDFF are not externally addressable. "z" represents the value of bits EE[3:0] in the config register.
5. EEPROM can be remapped to any 4-Kbyte boundary ($z000). "z" represents the value contained in EE[3:0] in the config register.

Figure 4-1 MC68HC11F1 Memory Map

4.2.2.1 RAM

The MC68HC11F1 microcontroller has 1024 bytes of fully static RAM that can be used for storing instructions, variables, and temporary data during program execution. RAM can be placed at any 4-Kbyte boundary in the 64 Kbyte address space by writing an appropriate value to the INIT register.

RAM is initially located at $0000 in the memory map upon reset. Direct addressing mode can access the first 256 locations of RAM using a one-byte address operand. Direct mode accesses save program memory space and execution time.

The on-chip RAM is a fully static memory. RAM contents can be preserved during periods of processor inactivity by either of two methods, both of which reduce power consumption.

Table 4-2 Write Access Limited Registers

Register Address	Register Name	Must be Written in First 64 Cycles	Write One Time Only
$x024	Timer Interrupt Mask 2 (TMSK2)	Note 1	—
$x035	Block Protect Register (BPROT)	Note 2	—
$x038	System Configuration Options 2 (OPT2)	No	Note 4
$x039	System Configuration Options (OPTION)	Note 3	—
$x03C	Highest Priority I-bit and Miscellaneous (HPRIO)	No	Note 5
$x03D	RAM and I/O Map Register (INIT)	Yes	Note 6

Notes:
1. Bits 1 and 0 can be written once only in first 64 cycles. When SMOD = 1, these bits can be written any time. All other bits can be written at any time.
2. Bits can be written to zero (protection disabled) once only in first 64 cycles or at any time in special modes. Bits can be set to one at any time.
3. Bits 5, 4, 2, 1, and 0 can be written once only in first 64 cycles. When SMOD = 1, bits 5, 4, 2, 1, and 0 can be written at any time. All other bits can be written at any time
4. Bit 5 (CLK4X) can be written only one time.
5. Bit 4 (IRV) can be written only one time.
6. Can be written once in first 64 cycles after reset in normal modes or at any time in special modes.

4.3.1 Mode Selection

The four mode variations are selected by the logic levels present on the MODA and MODB pins at the rising edge of $\overline{\text{RESET}}$. The MODA and MODB logic levels determine the logic state of SMOD and MDA control bits in the HPRIO register.

After reset is released, the mode select pins no longer influence the MCU operating mode. In single-chip operating mode, the MODA pin is connected to a logic level zero. In expanded mode, MODA should be connected to V_{DD} through a pull-up resistor of 4.7 kΩ. The MODA pin also functions as the load instruction register ($\overline{\text{LIR}}$) pin when the MCU is not in reset. The open-drain active low $\overline{\text{LIR}}$ output pin drives low during the first E cycle of each instruction (opcode fetch). The MODB pin also functions as stand-by power input (V_{STBY}), which allows RAM contents to be maintained in absence of V_{DD}. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for V_{STBY} voltage requirements.

Refer to **Table 4-3**, which is a summary of mode pin operation, the mode control bits, and the four operating modes.

Table 4-3 Hardware Mode Select Summary

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

```
ROWE    LDAB    #$0E        ROW=1, ERASE=1, EELAT=1, EEPGM=0
        STAB    $103B       Set to ROW erase mode
        STAB    0,X         Store any data to any address in ROW
        LDAB    #$0F        ROW=1, ERASE=1, EELAT=1, EEPGM=1
        STAB    $103B       Turn on high voltage
        JSR     DLY10       Delay 10 ms
        CLR     $103B       Turn off high voltage and set to READ mode
```

4.4.1.4 EEPROM Byte Erase

The following is an example of how to erase a single byte of EEPROM and assumes that index register X contains the address of the byte to be erased.

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

4.4.2 PPROG EEPROM Programming Control Register

Bits in PPROG register control parameters associated with EEPROM programming.

PPROG — EEPROM Programming Control **$103B**

	Bit 7	6	5	4	3	2	1	Bit 0
	ODD	EVEN	—	BYTE	ROW	ERASE	EELAT	EEPGM
RESET:	0	0	0	0	0	0	0	0

ODD — Program Odd Rows in Half of EEPROM (TEST)

EVEN — Program Even Rows in Half of EEPROM (TEST)

Bit 5 — Not implemented
Always reads zero

BYTE — Byte/Other EEPROM Erase Mode
0 = Row or bulk erase mode used
1 = Erase only one byte of EEPROM

ROW — Row/All EEPROM Erase Mode (only valid when BYTE = 0)
0 = All 512 bytes of EEPROM erased
1 = Erase only one 16-byte row of EEPROM

For a description of the bits contained in the CONFIG register refer to **4.3.2.1 CONFIG Register**.

4.5 Chip Selects

The function of the chip selects is to minimize the amount of external glue logic needed to interface the MCU to external devices. The MC68HC11F1 has four software configured chip selects that can be enabled in expanded modes. The chip selects for I/O (CSIO1 and CSIO2) are used for I/O expansion. The program chip select ($\overline{\text{CSPROG}}$) is used with an external memory that contains the program code and reset vectors. The general-purpose chip select (CSGEN) is the most flexible and is used to enable external devices.

Such factors as polarity, block size, base address and clock stretching can be controlled using the four chip-select control registers. When a port G pin is not used for chip select functions it can be used for general-purpose I/O.

When enabled, a chip select signal is asserted whenever the CPU makes an access to a designated range of addresses. Bus control signals and chip select signals are synchronous with the external E clock signal. For more information refer to Table A–7. Expansion Bus Timing in **APPENDIX A ELECTRICAL CHARACTERISTICS**. The length of the external E clock cycle to which the external device is synchronized can be stretched to accommodate devices that are slower than the MCU.

4.5.1 Program Chip Select

The program chip select ($\overline{\text{CSPROG}}$) is active in the range of memory where the main program exists. Refer to **Figure 4-3**.

When enabled, the $\overline{\text{CSPROG}}$ is active during address valid time and is an active-low signal. Although the general-purpose chip select has priority over the program chip select, $\overline{\text{CSPROG}}$ can be raised to a higher priority level by setting the GCSPR bit in CSCTL register. Bits in CSCTL enable the program chip select and determine its address range and priority level. Bits in CSSTRH select from zero to three clock cycles of delay.

4.5.2 I/O Chip Selects

The I/O chip selects (CSIO1 and CSIO2) are fixed in size and fill the remainder of the 4-Kbyte block occupied by the register block. CSIO1 is mapped at $x060–$x7FF and CSIO2 is mapped at $x800–$xFFF, where “x” corresponds to the high-order nibble of the register block base address, represented by the value contained in REG[3:0] in the INIT register.

Bits in the CSCTL register determine the polarity of the active state and enable both I/O chip selects. Bits in CSGSIZ select whether each chip select is active for address-valid or E-valid time. Bits in CSSTRH select from zero to three clock cycles of delay. Refer to **Figure 4-3**.

CR[1:0] — COP Timer Rate Select

The internal E clock is first divided by 2^{15} before it enters the COP watchdog system. These control bits determine a scaling factor for the watchdog timer. Refer to **Table 5-1**.

5.1.6 CONFIG Register

CONFIG — System Configuration Register **$103F**

	Bit 7	6	5	4	3	2	1	Bit 0	
	EE3	EE2	EE1	EE0	—	NOCOP	—	EEON	
RESET:	1	1	1	1	1	P	1	1	Single Chip
	1	1	1	1	1	P(L)	1	1	Bootstrap
	P	P	P	P	1	P	1	P	Expanded
	P	P	P	P	1	P(L)	1	0	Special Test

P indicates a previously programmed bit. P(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 DISR in TEST1 register.

EE[3:0] — EEPROM Mapping Control

Refer to **SECTION 4 OPERATING MODES AND ON-CHIP MEMORY**.

Bit 3 — Not implemented

Always reads one

NOCOP — COP System Disable

0 = COP system enabled (forces reset on time-out)

1 = COP system disabled

Bit 1 — Not implemented

Always reads one

EEON — EEPROM Enable

Refer to **SECTION 4 OPERATING MODES AND ON-CHIP MEMORY**.

5.2 Effects of Reset

When a reset condition is recognized, the internal registers and control bits are forced to an initial state. Depending on the cause of the reset and the operating mode, the reset vector can be fetched from any of six possible locations. Refer to **Table 5-2**.

Table 5-2 Reset Cause, Operating Mode, and Reset Vector

Cause of Reset	Normal Mode Vector	Special Test or Bootstrap
POR or RESET Pin	$FFFE, FFFF	$BFFE, $BFFF
Clock Monitor Failure	$FFFC, FFFD	$BFFC, $BFFD
COP Watchdog Time-out	$FFFA, FFFB	$BFFA, $BFFB

These initial states then control on-chip peripheral systems to force them to known start-up states, as follows:

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.

5.4.4 Software Interrupt

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

5.4.5 Maskable Interrupts

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

5.4.6 Reset and Interrupt Processing

Figure 5-1 and **Figure 5-3** illustrate the reset and interrupt process. **Figure 5-1** illustrates how the CPU begins from a reset and how interrupt detection relates to normal opcode fetches. **Figure 5-3** is an expansion of a block in **Figure 5-1** and illustrates interrupt priorities. **Figure 5-5** shows the resolution of interrupt sources within the SCI subsystem.

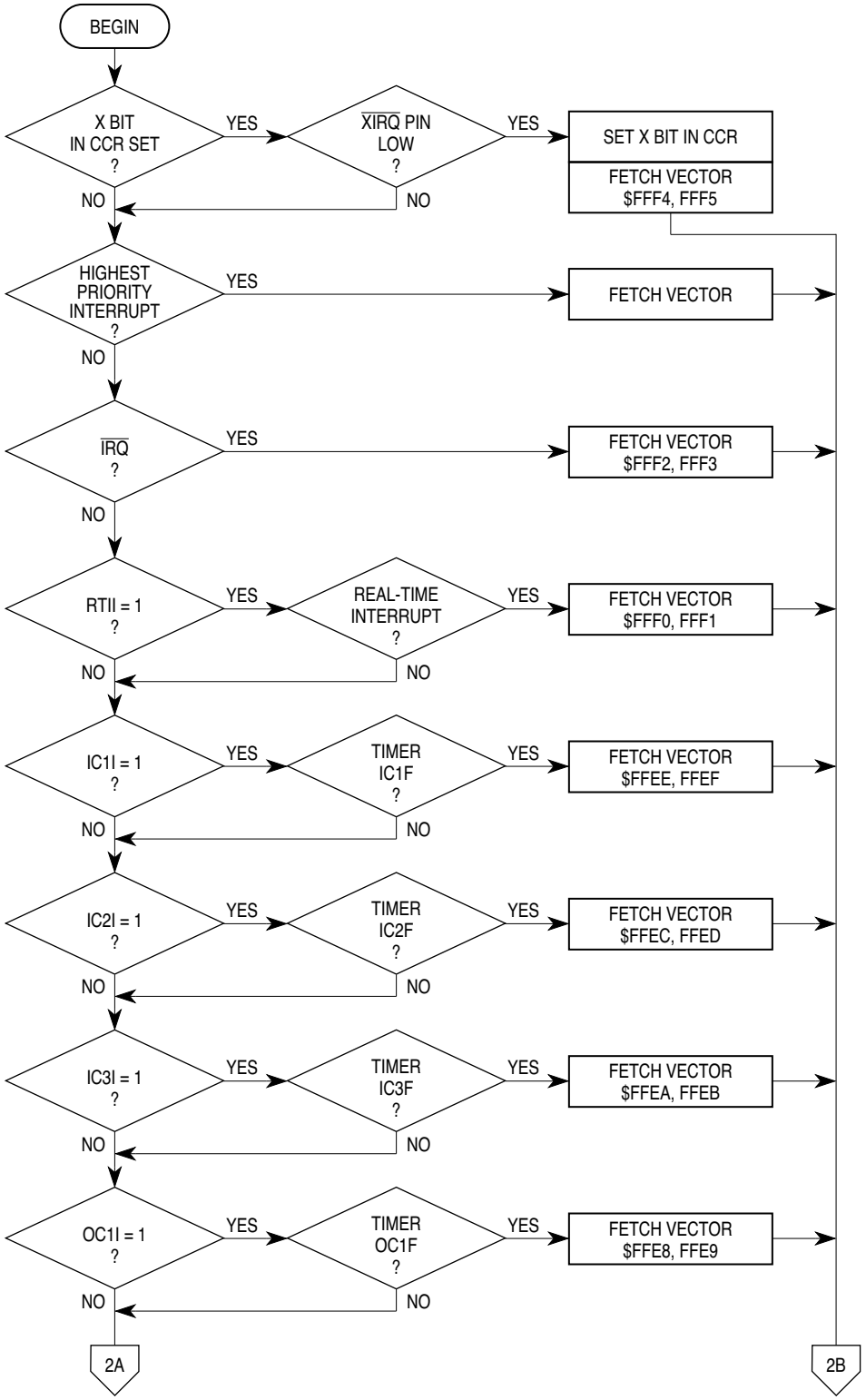


Figure 5-3 Interrupt Priority Resolution (1 of 2)

SECTION 7 SERIAL COMMUNICATIONS INTERFACE

The serial communications interface (SCI) is a universal asynchronous receiver transmitter (UART), one of two independent serial I/O subsystems in the MC68HC11F1 MCU. It has a standard nonreturn to zero (NRZ) format (one start bit, eight or nine data bits, and one stop bit). Several baud rates are available. The SCI transmitter and receiver are independent, but use the same data format and bit rate.

7.1 Data Format

The serial data format requires the following conditions:

1. An idle-line in the high state before transmission or reception of a message.
2. A start bit, logic zero, transmitted or received, that indicates the start of each character.
3. Data that is transmitted and received least significant bit (LSB) first.
4. A stop bit, logic one, used to indicate the end of a frame. (A frame consists of a start bit, a character of eight or nine data bits, and a stop bit.)
5. A break (defined as the transmission or reception of a logic zero for some multiple number of frames).

Selection of the word length is controlled by the M bit of SCI control register SCCR1.

7.2 Transmit Operation

The SCI transmitter includes a parallel transmit data register (SCDR) and a serial shift register. The contents of the serial shift register can only be written through the SCDR. This double buffered operation allows a character to be shifted out serially while another character is waiting in the SCDR to be transferred into the serial shift register. The output of the serial shift register is applied to TxD as long as transmission is in progress or the transmit enable (TE) bit of serial communication control register 2 (SCCR2) is set. The block diagram, **Figure 7-1**, shows the transmit serial shift register and the buffer logic at the top of the figure.

When the SPI system is configured as a master and the $\overline{SS}$ input line goes to active low, a mode fault error has occurred — usually because two devices have attempted to act as master at the same time. In cases where more than one device is concurrently configured as a master, there is a chance of contention between two pin drivers. For push-pull CMOS drivers, this contention can cause permanent damage. The mode fault mechanism attempts to protect the device by disabling the drivers. The MSTR control bit in the SPCR and all four DDRD control bits associated with the SPI are cleared and an interrupt is generated subject to masking by the SPIE control bit and the I bit in the CCR.

Other precautions may need to be taken to prevent driver damage. If two devices are made masters at the same time, mode fault does not help protect either one unless one of them selects the other as slave. The amount of damage possible depends on the length of time both devices attempt to act as master.

A write collision error occurs if the SPDR is written while a transfer is in progress. Because the SPDR is not double buffered in the transmit direction, writes to SPDR cause data to be written directly into the SPI shift register. Because this write corrupts any transfer in progress, a write collision error is generated. The transfer continues undisturbed, and the write data that caused the error is not written to the shifter.

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

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

8.5 SPI Registers

The three SPI registers, SPCR, SPSR, and SPDR, provide control, status, and data storage functions. Refer to the following information for a description of how these registers are organized.

8.5.1 Serial Peripheral Control

SPCR — Serial Peripheral Control Register **$1028**

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

To produce a pulse of a specific duration, write a value to the output compare register that represents the time the leading edge of the pulse is to occur. The output compare circuit is configured to set the appropriate output either high or low, depending on the polarity of the pulse being produced. After a match occurs, the output compare register is reprogrammed to change the output pin back to its inactive level at the next match. A value representing the width of the pulse is added to the original value, and then written to the output compare register. Because the pin state changes occur at specific values of the free-running counter, the pulse width can be controlled accurately at the resolution of the free-running counter, independent of software latencies. To generate an output signal of a specific frequency and duty cycle, repeat this pulse-generating procedure.

There are four 16-bit read/write output compare registers: TOC1, TOC2, TOC3, and TOC4, and the TI4/O5 register, which functions under software control as either IC4 or OC5. Each of the OC registers is set to $FFFF on reset. A value written to an OC register is compared to the free-running counter value during each E-clock cycle. If a match is found, the particular output compare flag is set in timer interrupt flag register 1 (TFLG1). If that particular interrupt is enabled in the timer interrupt mask register 1 (TMSK1), an interrupt is generated. In addition to an interrupt, a specified action can be initiated at one or more timer output pins. For OC[5:2], the pin action is controlled by pairs of bits (OMx and OLx) in the TCTL1 register. The output action is taken on each successful compare, regardless of whether or not the OCxF flag in the TFLG1 register was previously cleared.

OC1 is different from the other output compares in that a successful OC1 compare can affect any or all five of the OC pins. The OC1 output action taken when a match is found is controlled by two 8-bit registers with three bits unimplemented: the output compare 1 mask register, OC1M, and the output compare 1 data register, OC1D. OC1M specifies which port A outputs are to be used, and OC1D specifies what data is placed on these port pins.

9.3.1 Timer Output Compare Registers

All output compare registers are 16-bit read-write. Each is initialized to $FFFF at reset. If an output compare register is not used for an output compare function, it can be used as a storage location. A write to the high-order byte of an output compare register pair inhibits the output compare function for one bus cycle. This inhibition prevents inappropriate subsequent comparisons. Coherency requires a complete 16-bit read or write. However, if coherency is not needed, byte accesses can be used.

For output compare functions, write a comparison value to output compare registers TOC1–TOC4 and TI4/O5. When TCNT value matches the comparison value, specified pin actions occur.

OC1M — Output Compare 1 Mask **$100C**

	Bit 7	6	5	4	3	2	1	Bit 0
	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	—	—	—
RESET:	0	0	0	0	0	0	0	0

OC1M[7:3] — Output Compare Masks
0 = OC1 is disabled.
1 = OC1 is enabled to control the corresponding pin of port A

Bits [2:0] — Not implemented
Always read zero

9.3.4 Output Compare Data Register

Use this register with OC1 to specify the data that is to be stored on the affected pin of port A after a successful OC1 compare. When a successful OC1 compare occurs, a data bit in OC1D is stored in the corresponding bit of port A for each bit that is set in OC1M.

OC1D — Output Compare 1 Data **$100D**

	Bit 7	6	5	4	3	2	1	Bit 0
	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	—	—	—
RESET:	0	0	0	0	0	0	0	0

If OC1Mx is set, data in OC1Dx is output to port A bit x on successful OC1 compares.

Bits [2:0] — Not implemented
Always read zero

9.3.5 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.

TCNT — Timer Counter **$100E, $100F**

$100E	Bit 15	14	13	12	11	10	9	Bit 8	TCNT (High)
$100F	Bit 7	6	5	4	3	2	1	Bit 0	TCNT (Low)

TCNT resets to $0000. In normal modes, TCNT is a read-only register.

9.3.6 Timer Control Register 1

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

Table 9-3 Timer Prescaler Selection

PR[1:0]	Prescaler
0 0	1
0 1	4
1 0	8
1 1	16

NOTE

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

9.3.10 Timer Interrupt Flag Register 2

Bits in this register indicate when certain timer system events have occurred. Coupled with the four high-order bits of TMSK2, the bits of TFLG2 allow the timer subsystem to operate in either a polled or interrupt driven system. Each bit of TFLG2 corresponds to a bit in TMSK2 in the same position.

TFLG2 — Timer Interrupt Flag 2 **$1025**

	Bit 7	6	5	4	3	2	1	Bit 0
	TOF	RTIF	PAOVF	PAIF	—	—	—	—
RESET:	0	0	0	0	0	0	0	0

Clear flags by writing a one to the corresponding bit position(s).

TOF — Timer Overflow Interrupt Flag
Set when TCNT changes from $FFFF to $0000

RTIF — Real-Time (Periodic) Interrupt Flag
Refer to **9.4 Real-Time Interrupt**.

PAOVF — Pulse Accumulator Overflow Interrupt Flag
Refer to **9.6 Pulse Accumulator**.

PAIF — Pulse Accumulator Input Edge Interrupt Flag
Refer to **9.6 Pulse Accumulator**.

Bits [3:0] — Not implemented
Always read zero

9.4 Real-Time Interrupt

The real-time interrupt (RTI) feature, used to generate hardware interrupts at a fixed periodic rate, is controlled and configured by two bits (RTR1 and RTR0) in the pulse accumulator control (PACTL) register. The RTII bit in the TMSK2 register enables the interrupt capability. The four different rates available are a product of the MCU oscillator frequency and the value of bits RTR[1:0]. Refer to **Table 9-4**, which shows the periodic real-time interrupt rates.

9.4.2 Timer Interrupt Flag Register 2

Bits of this register indicate the occurrence of timer system events. Coupled with the four high-order bits of TMSK2, the bits of TFLG2 allow the timer subsystem to operate in either a polled or interrupt driven system. Each bit of TFLG2 corresponds to a bit in TMSK2 in the same position.

TFLG2 — Timer Interrupt Flag 2 **$1025**

	Bit 7	6	5	4	3	2	1	Bit 0
	TOF	RTIF	PAOVF	PAIF	—	—	—	—
RESET:	0	0	0	0	0	0	0	0

Clear flags by writing a one to the corresponding bit position(s).

TOF — Timer Overflow Interrupt Flag
Set when TCNT changes from $FFFF to $0000

RTIF — Real-Time Interrupt Flag
The RTIF status bit is automatically set to one at the end of every RTI period. To clear RTIF, write a byte to TFLG2 with bit 6 set.

PAOVF — Pulse Accumulator Overflow Interrupt Flag
Refer to **9.6 Pulse Accumulator**.

PAIF — Pulse Accumulator Input Edge Interrupt Flag
Refer to **9.6 Pulse Accumulator**.

Bits [3:0] — Not implemented
Always read zero

9.4.3 Pulse Accumulator Control Register

Bits RTR[1:0] of this register select the rate for the RTI system. The remaining bits control the pulse accumulator and IC4/OC5 functions.

PACTL — Pulse Accumulator Control **$1026**

	Bit 7	6	5	4	3	2	1	Bit 0
	—	PAEN	PAMOD	PEDGE	—	I4/O5	RTR1	RTR0
RESET:	0	0	0	0	0	0	0	0

Bit 7 — Not implemented
Always reads zero

PAEN — Pulse Accumulator System Enable
Refer to **9.6 Pulse Accumulator**.

PAMOD — Pulse Accumulator Mode
Refer to **9.6 Pulse Accumulator**.