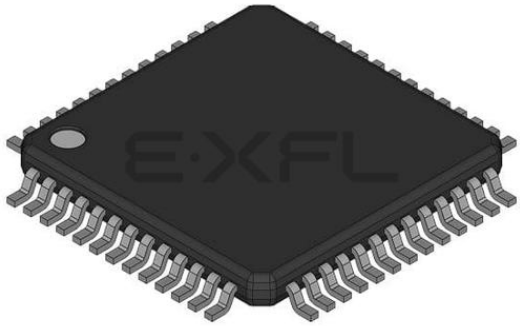


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	HC11
Core Size	8-Bit
Speed	2MHz
Connectivity	SCI, SPI
Peripherals	POR, WDT
Number of I/O	38
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LQFP
Supplier Device Package	52-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc11e1cpbe2

MC68HC11E Family

Data Sheet

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://freescale.com/

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Table 1-1. Port Signal Functions

Port/Bit	Single-Chip and Bootstrap Modes	Expanded and Test Modes
PA0	PA0/IC3	
PA1	PA1/IC2	
PA2	PA2/IC1	
PA3	PA3/OC5/IC4/OC1	
PA4	PA4/OC4/OC1	
PA5	PA5/OC3/OC1	
PA6	PA6/OC2/OC1	
PA7	PA7/PAI/OC1	
PB0	PB0	ADDR8
PB1	PB1	ADDR9
PB2	PB2	ADDR10
PB3	PB3	ADDR11
PB4	PB4	ADDR12
PB5	PB5	ADDR13
PB6	PB6	ADDR14
PB7	PB7	ADDR15
PC0	PC0	ADDR0/DATA0
PC1	PC1	ADDR1/DATA1
PC2	PC2	ADDR2/DATA2
PC3	PC3	ADDR3/DATA3
PC4	PC4	ADDR4/DATA4
PC5	PC5	ADDR5/DATA5
PC6	PC6	ADDR6/DATA6
PC7	PC7	ADDR7/DATA7
PD0	PD0/RxD	
PD1	PD1/TxD	
PD2	PD2/MISO	
PD3	PD3/MOSI	
PD4	PD4/SCK	
PD5	PD5/$\overline{SS}$	
—	STRA	AS
—	STRB	R/$\overline{W}$
PE0	PE0/AN0	
PE1	PE1/AN1	
PE2	PE3/AN2	
PE3	PE3/AN3	
PE4	PE4/AN4	
PE5	PE5/AN5	
PE6	PE6/AN6	
PE7	PE7/AN7	

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)

NOSEC — Security Disable Bit

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

0 = Security enabled
1 = Security disabled

NOCOP — COP System Disable Bit

Refer to Chapter 5 Resets and Interrupts.

1 = COP disabled
0 = COP enabled

ROMON — ROM/EPROM/OTPROM Enable Bit

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

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

EEON — EEPROM Enable Bit

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

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

2.3.3.2 RAM and I/O Mapping Register

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

Address: $103D

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

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

RAM[3:0] — RAM Map Position Bits

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

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

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

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

Refer to Chapter 5 Resets and Interrupts.

DLY — Enable Oscillator Startup Delay Bit

0 = The oscillator startup delay coming out of stop mode is bypassed and the MCU resumes processing within about four bus cycles.

1 = A delay of approximately 4000 E-clock cycles is imposed as the MCU is started up from the stop power-saving mode. This delay allows the crystal oscillator to stabilize.

CME — Clock Monitor Enable Bit

Refer to Chapter 5 Resets and Interrupts.

Bit 2 — Not implemented

Always reads 0

CR[1:0] — COP Timer Rate Select Bits

The internal E clock is divided by 2^{15} before it enters the COP watchdog system. These control bits determine a scaling factor for the watchdog timer. Refer to Chapter 5 Resets and Interrupts.

2.4 EPROM/OTPROM

Certain devices in the M68HC11 E series include on-chip EPROM/OTPROM. For instance:

- The MC68HC711E9 devices contain 12 Kbytes of on-chip EPROM (OTPROM in non-windowed package).
- The MC68HC711E20 has 20 Kbytes of EPROM (OTPROM in non-windowed package).
- The MC68HC711E32 has 32 Kbytes of EPROM (OTPROM in non-windowed package).

Standard MC68HC71E9 and MC68HC711E20 devices are shipped with the EPROM/OTPROM contents erased (all 1s). The programming operation programs zeros. Windowed devices must be erased using a suitable ultraviolet light source before reprogramming. Depending on the light source, erasing can take from 15 to 45 minutes.

Using the on-chip EPROM/OTPROM programming feature requires an external 12-volt nominal power supply (V_{PPE}). Normal programming is accomplished using the EPROM/OTPROM programming register (PPROG).

PPROG is the combined EPROM/OTPROM and EEPROM programming register on all devices with EPROM/OTPROM except the MC68HC711E20. For the MC68HC711E20, there is a separate register for EPROM/OTPROM programming called the EPROG register.

As described in the following subsections, these two methods of programming and verifying EPROM are possible:

1. Programming an individual EPROM address
2. Programming the EPROM with downloaded data

2.5.1.2 EPROM and EEPROM Programming Control Register

The EPROM and EEPROM programming control register (PPROG) selects and controls the EEPROM programming function. Bits in PPROG enable the programming voltage, control the latching of data to be programmed, and select the method of erasure (for example, byte, row, etc.).

Address: $103B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ODD	EVEN	ELAT(1)	BYTE	ROW	ERASE	EELAT	EPGM
Reset:	0	0	0	0	0	0	0	0

1. MC68HC711E9 only

Figure 2-17. EPROM and EEPROM Programming Control Register (PPROG)

ODD — Program Odd Rows in Half of EEPROM (Test) Bit

EVEN — Program Even Rows in Half of EEPROM (Test) Bit

ELAT — EPROM/OTPROM Latch Control Bit

For the MC68HC711E9, EPGM enables the high voltage necessary for both EPROM/OTPROM and EEPROM programming.

For MC68HC711E9, ELAT and EELAT are mutually exclusive and cannot both equal 1.

0 = EPROM address and data bus configured for normal reads
1 = EPROM address and data bus configured for programming

BYTE — Byte/Other EEPROM Erase Mode Bit

This bit overrides the ROW bit.

0 = Row or bulk erase
1 = Erase only one byte

ROW — Row/All EEPROM Erase Mode Bit

If BYTE is 1, ROW has no meaning.

0 = Bulk erase
1 = Row erase

Table 2-8. EEPROM Erase

BYTE	ROW	Action
0	0	Bulk erase (entire array)
0	1	Row erase (16 bytes)
1	0	Byte erase
1	1	Byte erase

ERASE — Erase Mode Select Bit

0 = Normal read or program mode
1 = Erase mode

EELAT — EEPROM Latch Control Bit

0 = EEPROM address and data bus configured for normal reads and cannot be programmed
1 = EEPROM address and data bus configured for programming or erasing and cannot be read

Chapter 3
Analog-to-Digital (A/D) Converter

3.1 Introduction

The analog-to-digital (A/D) system, a successive approximation converter, uses an all-capacitive charge redistribution technique to convert analog signals to digital values.

3.2 Overview

The A/D system is an 8-channel, 8-bit, multiplexed-input converter. The converter does not require external sample and hold circuits because of the type of charge redistribution technique used. A/D converter timing can be synchronized to the system E clock or to an internal resistor capacitor (RC) oscillator.

The A/D converter system consists of four functional blocks: multiplexer, analog converter, digital control, and result storage. Refer to Figure 3-1.

3.2.1 Multiplexer

The multiplexer selects one of 16 inputs for conversion. Input selection is controlled by the value of bits CD:CA in the ADCTL register. The eight port E pins are fixed-direction analog inputs to the multiplexer, and additional internal analog signal lines are routed to it.

Port E pins also can be used as digital inputs. Digital reads of port E pins are not recommended during the sample portion of an A/D conversion cycle, when the gate signal to the N-channel input gate is on. Because no P-channel devices are directly connected to either input pins or reference voltage pins, voltages above V_{DD} do not cause a latchup problem, although current should be limited according to maximum ratings. Refer to Figure 3-2, which is a functional diagram of an input pin.

3.2.2 Analog Converter

Conversion of an analog input selected by the multiplexer occurs in this block. It contains a digital-to-analog capacitor (DAC) array, a comparator, and a successive approximation register (SAR). Each conversion is a sequence of eight comparison operations, beginning with the most significant bit (MSB). Each comparison determines the value of a bit in the successive approximation register.

The DAC array performs two functions. It acts as a sample and hold circuit during the entire conversion sequence and provides comparison voltage to the comparator during each successive comparison.

The result of each successive comparison is stored in the SAR. When a conversion sequence is complete, the contents of the SAR are transferred to the appropriate result register.

A charge pump provides switching voltage to the gates of analog switches in the multiplexer. Charge pump output must stabilize between 7 and 8 volts within up to 100 μs before the converter can be used. The charge pump is enabled by the ADPU bit in the OPTION register.

6.3 Port B

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

Address: $1004

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

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

6.4 Port C

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

Address: $1003

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

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

Address: $1005

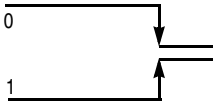
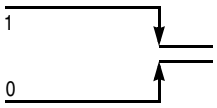
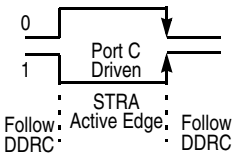
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PCL7	PCL6	PCL5	PCL4	PCL3	PCL2	PCL1	PCL0
Reset:	Indeterminate after reset							

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

6.8 Parallel I/O Control Register

The parallel handshake functions are available only in the single-chip operating mode. PIOC is a read/write register except for bit 7, which is read only. Table 6-2 shows a summary of handshake operations.

Table 6-2. Parallel I/O Control

	STAF Clearing Sequence	HNDS	OIN	PLS	EGA	Port B	Port C
Simple strobed mode	Read PIOC with STAF = 1 then read PORTCL	0	X	X		Inputs latched into PORTCL on any active edge on STRA	STRB pulses on writes to PORTB
Full-input hand-shake mode	Read PIOC with STAF = 1 then read PORTCL	1	0	0 = STRB active level 1 = STRB active pulse		Inputs latched into PORTCL on any active edge on STRA	Normal output port, unaffected in handshake modes
Full-output hand-shake mode	Read PIOC with STAF = 1 then write PORTCL	1	1	0 = STRB active level 1 = STRB active pulse		Driven as outputs if STRA at active level; follows DDRC if STRA not at active level	Normal output port, unaffected in handshake modes

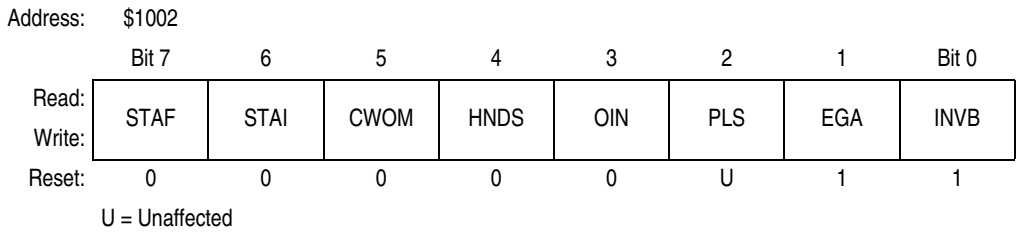


Figure 6-10. Parallel I/O Control Register (PIOC)

STAF — Strobe A Interrupt Status Flag

STAF is set when the selected edge occurs on strobe A. This bit can be cleared by a read of PIOC with STAF set followed by a read of PORTCL (simple strobed or full input handshake mode) or a write to PORTCL (output handshake mode).

0 = No edge on strobe A
1 = Selected edge on strobe A

STAI — Strobe A Interrupt Enable Mask Bit

0 = STAF does not request interrupt
1 = STAF requests interrupt

7.4 Receive Operation

During receive operations, the transmit sequence is reversed. The serial shift register receives data and transfers it to a parallel receive data register (SCDR) as a complete word. This double buffered operation allows a character to be shifted in serially while another character is already in the SCDR. An advanced data recovery scheme distinguishes valid data from noise in the serial data stream. The data input is selectively sampled to detect receive data, and a majority voting circuit determines the value and integrity of each bit. See Figure 7-2.

7.5 Wakeup Feature

The wakeup feature reduces SCI service overhead in multiple receiver systems. Software for each receiver evaluates the first character of each message. The receiver is placed in wakeup mode by writing a 1 to the RWU bit in the SCCR2 register. While RWU is 1, all of the receiver-related status flags (RDRF, IDLE, OR, NF, and FE) are inhibited (cannot become set). Although RWU can be cleared by a software write to SCCR2, to do so would be unusual. Normally, RWU is set by software and is cleared automatically with hardware. Whenever a new message begins, logic alerts the sleeping receivers to wake up and evaluate the initial character of the new message.

Two methods of wakeup are available:

- Idle-line wakeup
- Address-mark wakeup

During idle-line wakeup, a sleeping receiver awakens as soon as the RxD line becomes idle. In the address-mark wakeup, logic 1 in the most significant bit (MSB) of a character wakes up all sleeping receivers.

7.5.1 Idle-Line Wakeup

To use the receiver wakeup method, establish a software addressing scheme to allow the transmitting devices to direct a message to individual receivers or to groups of receivers. This addressing scheme can take any form as long as all transmitting and receiving devices are programmed to understand the same scheme. Because the addressing information is usually the first frame(s) in a message, receivers that are not part of the current task do not become burdened with the entire set of addressing frames. All receivers are awake (RWU = 0) when each message begins. As soon as a receiver determines that the message is not intended for it, software sets the RWU bit (RWU = 1), which inhibits further flag setting until the RxD line goes idle at the end of the message. As soon as an idle line is detected by receiver logic, hardware automatically clears the RWU bit so that the first frame of the next message can be received. This type of receiver wakeup requires a minimum of one idle-line frame time between messages and no idle time between frames in a message.

Table 9-1. Timer Summary

Control Bits PR1, PR0	XTAL Frequencies			
	4.0 MHz	8.0 MHz	12.0 MHz	Other Rates
	1.0 MHz	2.0 MHz	3.0 MHz	(E)
	1000 ns	500 ns	333 ns	(1/E)
	Main Timer Count Rates			
0 0 1 count — overflow —	1000 ns 65.536 ms	500 ns 32.768 ms	333 ns 21.845 ms	(E/1) ($E/2^{16}$)
0 1 1 count — overflow —	4.0 µs 262.14 ms	2.0 µs 131.07 ms	1.333 µs 87.381 ms	(E/4) ($E/2^{18}$)
1 0 1 count — overflow —	8.0 µs 524.29 ms	4.0 µs 262.14 ms	2.667 µs 174.76 ms	(E/8) ($E/2^{19}$)
1 1 1 count — overflow —	16.0 µs 1.049 s	8.0 µs 524.29 ms	5.333 µs 349.52 ms	(E/16) ($E/2^{20}$)

9.2 Timer Structure

Figure 9-2 shows the capture/compare system block diagram. The port A pin control block includes logic for timer functions and for general-purpose I/O. For pins PA3, PA2, PA1, and PA0, this block contains both the edge-detection logic and the control logic that enables the selection of which edge triggers an input capture. The digital level on PA[3:0] can be read at any time (read PORTA register), even if the pin is being used for the input capture function. Pins PA[6:3] are used for either general-purpose I/O, or as output compare pins. When one of these pins is being used for an output compare function, it cannot be written directly as if it were a general-purpose output. Each of the output compare functions (OC[5:2]) is related to one of the port A output pins. Output compare one (OC1) has extra control logic, allowing it optional control of any combination of the PA[7:3] pins. The PA7 pin can be used as a general-purpose I/O pin, as an input to the pulse accumulator, or as an OC1 output pin.

9.3 Input Capture

The input capture function records the time an external event occurs by latching the value of the free-running counter when a selected edge is detected at the associated timer input pin. Software can store latched values and use them to compute the periodicity and duration of events. For example, by storing the times of successive edges of an incoming signal, software can determine the period and pulse width of a signal. To measure period, two successive edges of the same polarity are captured. To measure pulse width, two alternate polarity edges are captured.

In most cases, input capture edges are asynchronous to the internal timer counter, which is clocked relative to an internal clock (PH2). These asynchronous capture requests are synchronized to PH2 so that the latching occurs on the opposite half cycle of PH2 from when the timer counter is being incremented. This synchronization process introduces a delay from when the edge occurs to when the counter value is detected. Because these delays offset each other when the time between two edges is being measured, the delay can be ignored. When an input capture is being used with an output compare, there is a similar delay between the actual compare point and when the output pin changes state.

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.

9.4.1 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.

Register name: Timer Output Compare 1 Register (High) Address: $1016

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

Register name: Timer Output Compare 1 Register (Low) Address: $1017

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

Figure 9-8. Timer Output Compare 1 Register Pair (TOC1)

Register name: Timer Output Compare 2 Register (High) Address: $1018

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

Register name: Timer Output Compare 2 Register (Low) Address: $1019

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

Figure 9-9. Timer Output Compare 2 Register Pair (TOC2)

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.

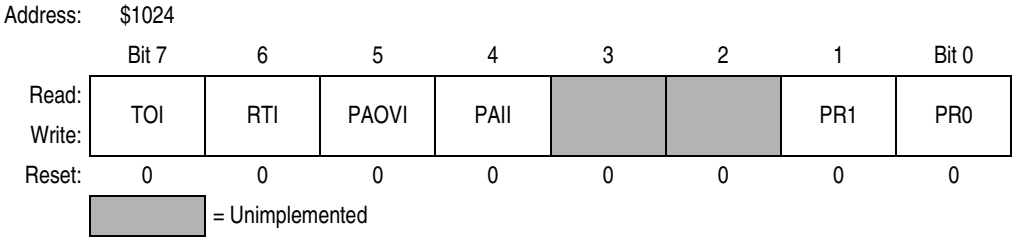


Figure 9-21. Timer Interrupt Mask 2 Register (TMSK2)

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.

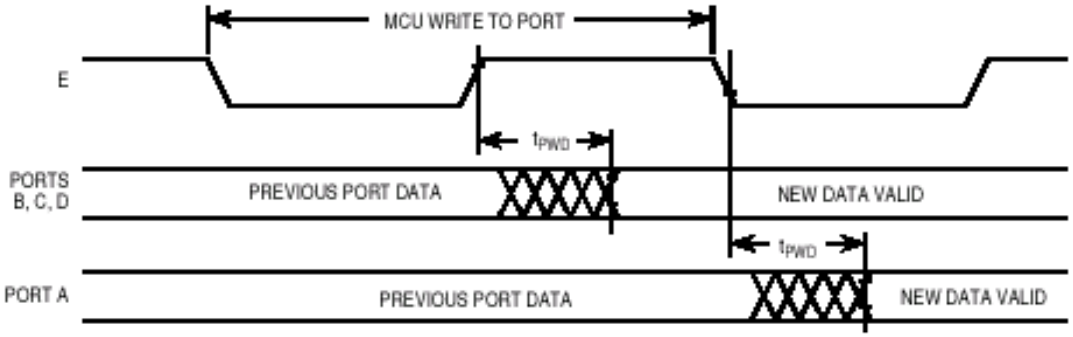


Figure 10-8. Port Write Timing Diagram

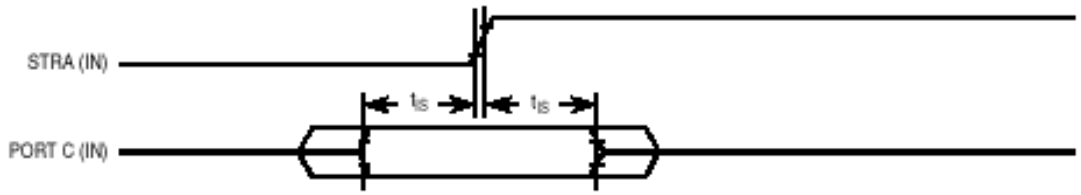


Figure 10-9. Simple Input Strobe Timing Diagram

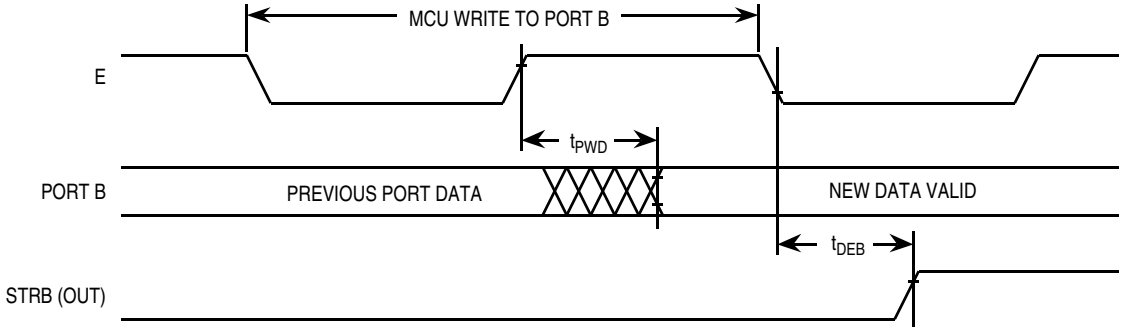
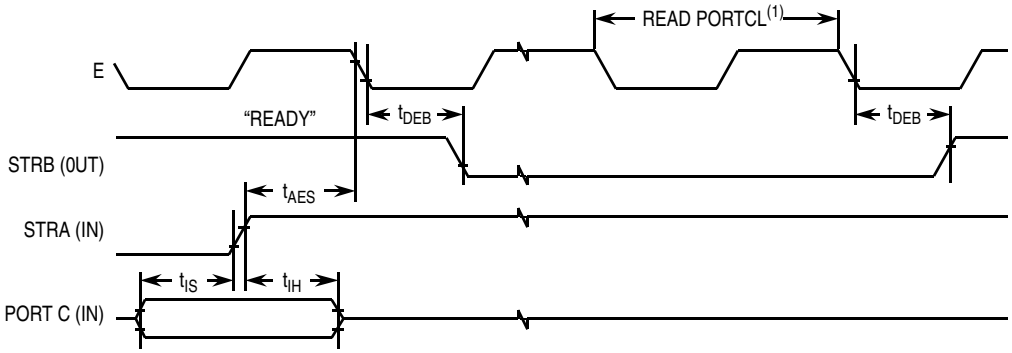


Figure 10-10. Simple Output Strobe Timing Diagram



Notes:
1. After reading PIOC with STAF set
2. Figure shows rising edge STRA (EGA = 1) and high true STRB (INVB = 1).

Figure 10-11. Port C Input Handshake Timing Diagram

Freescale Semiconductor
Application Note

AN1060
Rev. 1.1, 07/2005

M68HC11 Bootstrap Mode

By Jim Sibigtroth
Mike Rhoades
John Langan
Austin, Texas

Introduction

The M68HC11 Family of MCUs (microcontroller units) has a bootstrap mode that allows a user-defined program to be loaded into the internal random-access memory (RAM) by way of the serial communications interface (SCI); the M68HC11 then executes this loaded program. The loaded program can do anything a normal user program can do as well as anything a factory test program can do because protected control bits are accessible in bootstrap mode. Although the bootstrap mode is a single-chip mode of operation, expanded mode resources are accessible because the mode control bits can be changed while operating in the bootstrap mode.

This application note explains the operation and application of the M68HC11 bootstrap mode. Although basic concepts associated with this mode are quite simple, the more subtle implications of these functions require careful consideration. Useful applications of this mode are overlooked due to an incomplete understanding of bootstrap mode. Also, common problems associated with bootstrap mode could be avoided by a more complete understanding of its operation and implications.

Topics discussed in this application note include:

- Basic operation of the M68HC11 bootstrap mode
- General discussion of bootstrap mode uses
- Detailed explanation of on-chip bootstrap logic
- Detailed explanation of bootstrap firmware
- Bootstrap firmware vs. EEPROM security
- Incorporating the bootstrap mode into a system
- Driving bootstrap mode from another M68HC11
- Driving bootstrap mode from a personal computer
- Common bootstrap mode problems
- Variations for specific versions of M68HC11
- Commented listings for selected M68HC11 bootstrap ROMs

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 bootload 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.

To understand the detailed operation of the EPROM programming utility, refer to Figure 4 during the following discussion. Figure 4 is composed of three interrelated parts. The upper-left portion shows the flowchart of the PROGRAM utility running in the boot ROM of the MCU. The upper-right portion shows the flowchart for the user-supplied driver program running in the host computer. The lower portion of Figure 4 is a timing sequence showing the relationship of operations between the MCU and the host computer. Reference numbers in the flowcharts in the upper half of Figure 4 have matching numbers in the lower half to help the reader relate the three parts of the figure.

The shaded area [1] refers to the software and hardware latency in the MCU leading to the transmission of a character (in this case, the $FF). The shaded area [2] refers to a similar latency in the host computer (in this case, leading to the transmission of the first data character to the MCU).

The overall operation begins when the MCU sends the first character ($FF) to the host computer, indicating that it is ready for the first data character. The host computer sends the first data byte [3] and enters its main loop. The second data character is sent [4], and the host then waits [5] for the first verify byte to come back from the MCU.

After the MCU sends $FF [8], it enters the WAIT1 loop [9] and waits for the first data character from the host. When this character is received [10], the MCU programs it into the address pointed to by the Y index register. When the programming time delay is over, the MCU reads the programmed data, transmits it to the host for verification [11], and returns to the top of the WAIT1 loop to wait for the next data character [12]. Because the host previously sent the second data character, it is already waiting in the SCI receiver of the MCU. Steps [13], [14], and [15] correspond to the second pass through the WAIT1 loop.

Back in the host, the first verify character is received, and the third data character is sent [6]. The host then waits for the second verify character [7] to come back from the MCU. The sequence continues as long as the host continues to send data to the MCU. Since the WAIT1 loop in the PROGRAM utility is an indefinite loop, reset is used to end the process in the MCU after the host has finished sending data to be programmed.

Allowing for Bootstrap Mode

Since bootstrap mode requires few connections to the MCU, it is easy to design systems that accommodate bootstrap mode.

Bootstrap mode is useful for diagnosing or repairing systems that have failed due to changes in the CONFIG register or failures of the expansion address/data buses, (rendering programs in external memory useless). Bootstrap mode can also be used to load information into the EPROM or EEPROM of an M68HC11 after final assembly of a module. Bootstrap mode is also useful for performing system checks and calibration routines. The following paragraphs explain system requirements for use of bootstrap mode in a product.

Mode Select Pins

It must be possible to force the MODA and MODB pins to logic 0, which implies that these two pins should be pulled up to V_{DD} through resistors rather than being tied directly to V_{DD}. If mode pins are connected directly to V_{DD}, it is not possible to force a mode other than the one the MCU is hard wired for. It is also good practice to use pulldown resistors to V_{SS} rather than connecting mode pins directly to V_{SS} because it is sometimes a useful debug aid to attempt reset in modes other than the one the system was primarily designed for. Physically, this requirement sometimes calls for the addition of a test point or a wire connected to one or both mode pins. Mode selection only uses the mode pins while $\overline{\text{RESET}}$ is active.

$\overline{\text{RESET}}$

It must be possible to initiate a reset while the mode select pins are held low. In systems where there is no provision for manual reset, it is usually possible to generate a reset by turning power off and back on.

RxD Pin

It must be possible to drive the PD0/RxD pin with serial data from a host computer (or another MCU). In many systems, this pin is already used for SCI communications; thus no changes are required.

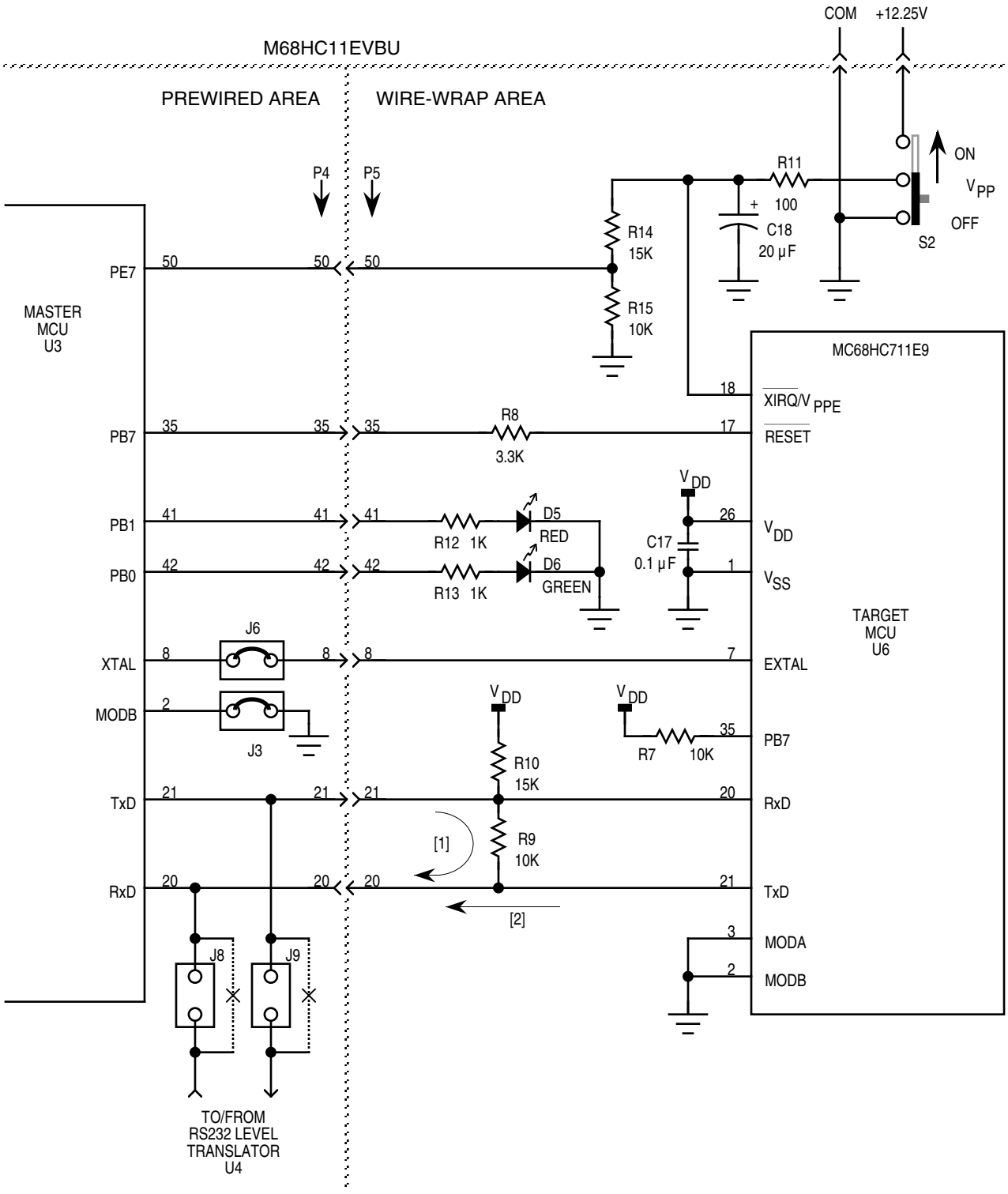


Figure 6. MCU-to-MCU EPROM Duplicator Schematic