M68HC11E Family

Data Sheet

HC11
Microcontrollers

M68HC11E
Rev. 5.1
07/2005

freescale.com

freescale™
semiconductor

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.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
© Freescale Semiconductor, Inc., 2005. All rights reserved.

Revision History

Date	Revision Level	Description	Page Number(s)
May, 2001	3.1	2.3.3.1 System Configuration Register — Addition to NOCOP bit description	44
		Added 10.21 EPROM Characteristics	175
June, 2001	3.2	10.21 EPROM Characteristics — For clarity, addition to note 2 following the table	175
December, 2001	3.3	7.7.2 Serial Communications Control Register 1 — SCCR1 bit 4 (M) description corrected	110
July, 2002	4	10.7 MC68L11E9/E20 DC Electrical Characteristics — Title changed to include the MC68L11E20	153
		10.8 MC68L11E9/E20 Supply Currents and Power Dissipation — Title changed to include the MC68L11E20	154
		10.10 MC68L11E9/E20 Control Timing — Title changed to include the MC68L11E20	157
		10.12 MC68L11E9/E20 Peripheral Port Timing — Title changed to include the MC68L11E20	163
		10.14 MC68L11E9/E20 Analog-to-Digital Converter Characteristics — Title changed to include the MC68L11E20	167
		10.16 MC68L11E9/E20 Expansion Bus Timing Characteristics — Title changed to include the MC68L11E20	169
		10.18 MC68L11E9/E20 Serial Peirpheral Interface Characteristics — Title changed to include the MC68L11E20	172
		— Title changed to include the MC68L11E20	175
		11.4 Extended Voltage Device Ordering Information (3.0 Vdc to 5.5 Vdc) — Updated table to include MC68L1120	181
June, 2003	5	Format updated to current publications standards	Throughout
		1.4.6 Non-Maskable Interrupt (XIRQ/VPPE) — Added Caution note pertaining to EPROM programming of the MC68HC711E9 device only.	23
		6.4 Port C — Clarified description of DDRC[7:0] bits	100
		10.21 EPROM Characteristics — Added note pertaining to EPROM programming of the MC68HC711E9 device only.	175
July, 2005	5.1	Updated to meet Freescale identity guidelines.	Throughout

List of Chapters

Chapter 1 General Description. 13

Chapter 2 Operating Modes and On-Chip Memory 29

Chapter 3 Analog-to-Digital (A/D) Converter 57

Chapter 4 Central Processor Unit (CPU) 65

Chapter 5 Resets and Interrupts 79

Chapter 6 Parallel Input/Output (I/O) Ports 97

Chapter 7 Serial Communications Interface (SCI) 105

Chapter 8 Serial Peripheral Interface (SPI) 119

Chapter 9 Timing Systems 127

Chapter 10 Electrical Characteristics. 149

Chapter 11 Ordering Information and Mechanical Specifications 177

Appendix A Development Support 187

Appendix B EVBU Schematic 191

AN1060 — M68HC11 Bootstrap Mode 193

EB184 — Enabling the Security Feature on the MC68HC711E9 Devices with PCbug11 on the M68HC711E9PGMR 229

EB188 — Enabling the Security Feature on M68HC811E2 Devices with PCbug11 on the M68HC711E9PGMR 233

EB296 — Programming MC68HC711E9 Devices with PCbug11 and the M68HC11EVBU 237

Table of Contents

Chapter 1
General Description

1.1 Introduction . . . 13
1.2 Features . . . 13
1.3 Structure . . . 14
1.4 Pin Descriptions . . . 14
1.4.1 V_{DD} and V_{SS} . . . 21
1.4.2 $\overline{\text{RESET}}$ . . . 22
1.4.3 Crystal Driver and External Clock Input (XTAL and EXTAL) . . . 22
1.4.4 E-Clock Output (E) . . . 23
1.4.5 Interrupt Request ($\overline{\text{IRQ}}$) . . . 23
1.4.6 Non-Maskable Interrupt ($\overline{\text{XIRQ}}$/V_{PPE}) . . . 23
1.4.7 MODA and MODB (MODA/$\overline{\text{LIR}}$ and MODB/V_{STBY}) . . . 24
1.4.7.1 V_{RL} and V_{RH} . . . 24
1.4.8 STRA/AS . . . 25
1.4.9 STRB/R/$\overline{\text{W}}$ . . . 25
1.4.10 Port Signals . . . 25
1.4.10.1 Port A . . . 25
1.4.10.2 Port B . . . 27
1.4.10.3 Port C . . . 27
1.4.10.4 Port D . . . 28
1.4.10.5 Port E . . . 28

Chapter 2
Operating Modes and On-Chip Memory

2.1 Introduction . . . 29
2.2 Operating Modes . . . 29
2.2.1 Single-Chip Mode . . . 29
2.2.2 Expanded Mode . . . 29
2.2.3 Test Mode . . . 30
2.2.4 Bootstrap Mode . . . 30
2.3 Memory Map . . . 31
2.3.1 RAM and Input/Output Mapping . . . 39
2.3.2 Mode Selection . . . 40
2.3.3 System Initialization . . . 42
2.3.3.1 System Configuration Register . . . 43
2.3.3.2 RAM and I/O Mapping Register . . . 45
2.3.3.3 System Configuration Options Register . . . 46
2.4 EPROM/OTPROM . . . 47
2.4.1 Programming an Individual EPROM Address . . . 48
2.4.2 Programming the EPROM with Downloaded Data . . . 48

2.4.3 EPROM and EEPROM Programming Control Register . . . 49
2.5 EEPROM . . . 51
2.5.1 EEPROM and CONFIG Programming and Erasure . . . 51
2.5.1.1 Block Protect Register . . . 51
2.5.1.2 EPROM and EEPROM Programming Control Register . . . 53
2.5.1.3 EEPROM Bulk Erase . . . 54
2.5.1.4 EEPROM Row Erase . . . 54
2.5.1.5 EEPROM Byte Erase . . . 55
2.5.1.6 CONFIG Register Programming . . . 55
2.5.2 EEPROM Security . . . 55

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

3.1 Introduction . . . 57
3.2 Overview . . . 57
3.2.1 Multiplexer . . . 57
3.2.2 Analog Converter . . . 57
3.2.3 Digital Control . . . 59
3.2.4 Result Registers . . . 59
3.2.5 A/D Converter Clocks . . . 59
3.2.6 Conversion Sequence . . . 59
3.3 A/D Converter Power-Up and Clock Select . . . 60
3.4 Conversion Process . . . 61
3.5 Channel Assignments . . . 61
3.6 Single-Channel Operation . . . 61
3.7 Multiple-Channel Operation . . . 62
3.8 Operation in Stop and Wait Modes . . . 62
3.9 A/D Control/Status Register . . . 62
3.10 A/D Converter Result Registers . . . 64

Chapter 4
Central Processor Unit (CPU)

4.1 Introduction . . . 65
4.2 CPU Registers . . . 65
4.2.1 Accumulators A, B, and D . . . 66
4.2.2 Index Register X (IX) . . . 66
4.2.3 Index Register Y (IY) . . . 66
4.2.4 Stack Pointer (SP) . . . 66
4.2.5 Program Counter (PC) . . . 68
4.2.6 Condition Code Register (CCR) . . . 68
4.2.6.1 Carry/Borrow (C) . . . 68
4.2.6.2 Overflow (V) . . . 68
4.2.6.3 Zero (Z) . . . 68
4.2.6.4 Negative (N) . . . 68
4.2.6.5 Interrupt Mask (I) . . . 69
4.2.6.6 Half Carry (H) . . . 69
4.2.6.7 X Interrupt Mask (X) . . . 69
4.2.6.8 STOP Disable (S) . . . 69

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

Chapter 5
Resets and Interrupts

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

Chapter 6
Parallel Input/Output (I/O) Ports

6.1 Introduction . . . 97
6.2 Port A . . . 98
6.3 Port B . . . 99
6.4 Port C . . . 99
6.5 Port D . . . 100
6.6 Port E . . . 101
6.7 Handshake Protocol . . . 101
6.8 Parallel I/O Control Register . . . 102

Chapter 7
Serial Communications Interface (SCI)

7.1 Introduction . . . 105
7.2 Data Format . . . 105
7.3 Transmit Operation . . . 105
7.4 Receive Operation . . . 107
7.5 Wakeup Feature . . . 107
7.5.1 Idle-Line Wakeup . . . 107
7.5.2 Address-Mark Wakeup . . . 109
7.6 SCI Error Detection . . . 109
7.7 SCI Registers . . . 109
7.7.1 Serial Communications Data Register . . . 110
7.7.2 Serial Communications Control Register 1 . . . 110
7.7.3 Serial Communications Control Register 2 . . . 111
7.7.4 Serial Communication Status Register . . . 112
7.7.5 Baud Rate Register . . . 113
7.8 Status Flags and Interrupts . . . 116
7.9 Receiver Flags . . . 117

Chapter 8
Serial Peripheral Interface (SPI)

8.1 Introduction . . . 119
8.2 Functional Description . . . 119
8.3 SPI Transfer Formats . . . 119
8.4 Clock Phase and Polarity Controls . . . 120
8.5 SPI Signals . . . 121
8.5.1 Master In/Slave Out . . . 121
8.5.2 Master Out/Slave In . . . 121
8.5.3 Serial Clock . . . 122
8.5.4 Slave Select . . . 122
8.6 SPI System Errors . . . 122
8.7 SPI Registers . . . 123
8.7.1 Serial Peripheral Control Register . . . 123
8.7.2 Serial Peripheral Status Register . . . 124
8.7.3 Serial Peripheral Data I/O Register . . . 125

Chapter 9
Timing Systems

9.1 Introduction . . . 127
9.2 Timer Structure . . . 129
9.3 Input Capture . . . 129
9.3.1 Timer Control Register 2 . . . 131
9.3.2 Timer Input Capture Registers . . . 131
9.3.3 Timer Input Capture 4/Output Compare 5 Register . . . 133
9.4 Output Compare . . . 133
9.4.1 Timer Output Compare Registers . . . 134
9.4.2 Timer Compare Force Register . . . 135
9.4.3 Output Compare Mask Register . . . 136
9.4.4 Output Compare Data Register . . . 136
9.4.5 Timer Counter Register . . . 137
9.4.6 Timer Control Register 1 . . . 137
9.4.7 Timer Interrupt Mask 1 Register . . . 138
9.4.8 Timer Interrupt Flag 1 Register . . . 138
9.4.9 Timer Interrupt Mask 2 Register . . . 139
9.4.10 Timer Interrupt Flag Register 2 . . . 140
9.5 Real-Time Interrupt (RTI) . . . 140
9.5.1 Timer Interrupt Mask Register 2 . . . 141
9.5.2 Timer Interrupt Flag Register 2 . . . 142
9.5.3 Pulse Accumulator Control Register . . . 142
9.6 Computer Operating Properly (COP) Watchdog Function . . . 143
9.7 Pulse Accumulator . . . 143
9.7.1 Pulse Accumulator Control Register . . . 145
9.7.2 Pulse Accumulator Count Register . . . 146
9.7.3 Pulse Accumulator Status and Interrupt Bits . . . 146

Chapter 10
Electrical Characteristics

10.1 Introduction . . . 149
10.2 Maximum Ratings for Standard and Extended Voltage Devices . . . 149
10.3 Functional Operating Range . . . 150
10.4 Thermal Characteristics . . . 150
10.5 DC Electrical Characteristics . . . 151
10.6 Supply Currents and Power Dissipation . . . 152
10.7 MC68L11E9/E20 DC Electrical Characteristics . . . 153
10.8 MC68L11E9/E20 Supply Currents and Power Dissipation . . . 154
10.9 Control Timing . . . 156
10.10 MC68L11E9/E20 Control Timing . . . 157
10.11 Peripheral Port Timing . . . 162
10.12 MC68L11E9/E20 Peripheral Port Timing . . . 163
10.13 Analog-to-Digital Converter Characteristics . . . 166
10.14 MC68L11E9/E20 Analog-to-Digital Converter Characteristics . . . 167

10.15 Expansion Bus Timing Characteristics . . . 168
10.16 MC68L11E9/E20 Expansion Bus Timing Characteristics . . . 169
10.17 Serial Peripheral Interface Timing Characteristics . . . 171
10.18 MC68L11E9/E20 Serial Peirpheral Interface Characteristics . . . 172
10.19 EEPROM Characteristics . . . 175
10.20 MC68L11E9/E20 EEPROM Characteristics . . . 175
10.21 EPROM Characteristics . . . 175

Chapter 11
Ordering Information and Mechanical Specifications

11.1 Introduction . . . 177
11.2 Standard Device Ordering Information . . . 177
11.3 Custom ROM Device Ordering Information . . . 179
11.4 Extended Voltage Device Ordering Information (3.0 Vdc to 5.5 Vdc) . . . 181
11.5 52-Pin Plastic-Leaded Chip Carrier (Case 778) . . . 182
11.6 52-Pin Windowed Ceramic-Leaded Chip Carrier (Case 778B) . . . 183
11.7 64-Pin Quad Flat Pack (Case 840C) . . . 184
11.8 52-Pin Thin Quad Flat Pack (Case 848D) . . . 185
11.9 56-Pin Dual in-Line Package (Case 859) . . . 186
11.10 48-Pin Plastic DIP (Case 767) . . . 186

Appendix A
Development Support

A.1 Introduction . . . 187
A.2 M68HC11 E-Series Development Tools . . . 187
A.3 EVS — Evaluation System . . . 187
A.4 Modular Development System (MMDS11) . . . 188
A.5 SPGMR11 — Serial Programmer for M68HC11 MCUs . . . 189

Appendix B
EVBU Schematic

AN1060 — M68HC11 Bootstrap Mode . . . 193

EB184 — Enabling the Security Feature on the MC68HC711E9 Devices with PCbug11 on the M68HC711E9PGMR . . . 229

EB188 — Enabling the Security Feature on M68HC811E2 Devices with PCbug11 on the M68HC711E9PGMR . . . 233

EB296 — Programming MC68HC711E9 Devices with PCbug11 and the M68HC11EVBU . . . 237

Chapter 1
General Description

1.1 Introduction

This document contains a detailed description of the M68HC11 E series of 8-bit microcontroller units (MCUs). These MCUs all combine the M68HC11 central processor unit (CPU) with high-performance, on-chip peripherals.

The E series is comprised of many devices with various configurations of:

- Random-access memory (RAM)
- Read-only memory (ROM)
- Erasable programmable read-only memory (EPROM)
- Electrically erasable programmable read-only memory (EEPROM)
- Several low-voltage devices are also available.

With the exception of a few minor differences, the operation of all E-series MCUs is identical. A fully static design and high-density complementary metal-oxide semiconductor (HCMOS) fabrication process allow the E-series devices to operate at frequencies from 3 MHz to dc with very low power consumption.

1.2 Features

Features of the E-series devices include:

- M68HC11 CPU
- Power-saving stop and wait modes
- Low-voltage devices available (3.0–5.5 Vdc)
- 0, 256, 512, or 768 bytes of on-chip RAM, data retained during standby
- 0, 12, or 20 Kbytes of on-chip ROM or EPROM
- 0, 512, or 2048 bytes of on-chip EEPROM with block protect for security
- 2048 bytes of EEPROM with selectable base address in the MC68HC811E2
- Asynchronous non-return-to-zero (NRZ) serial communications interface (SCI)
- Additional baud rates available on MC68HC(7)11E20
- Synchronous serial peripheral interface (SPI)
- 8-channel, 8-bit analog-to-digital (A/D) converter
- 16-bit timer system:
  - Three input capture (IC) channels
  - Four output compare (OC) channels
  - One additional channel, selectable as fourth IC or fifth OC
- 8-bit pulse accumulator
- Real-time interrupt circuit

- Computer operating properly (COP) watchdog system
- 38 general-purpose input/output (I/O) pins:
  - 16 bidirectional I/O pins
  - 11 input-only pins
  - 11 output-only pins
- Several packaging options:
  - 52-pin plastic-leaded chip carrier (PLCC)
  - 52-pin windowed ceramic leaded chip carrier (CLCC)
  - 52-pin plastic thin quad flat pack, 10 mm x 10 mm (TQFP)
  - 64-pin quad flat pack (QFP)
  - 48-pin plastic dual in-line package (DIP), MC68HC811E2 only
  - 56-pin plastic shrink dual in-line package, .070-inch lead spacing (SDIP)

1.3 Structure

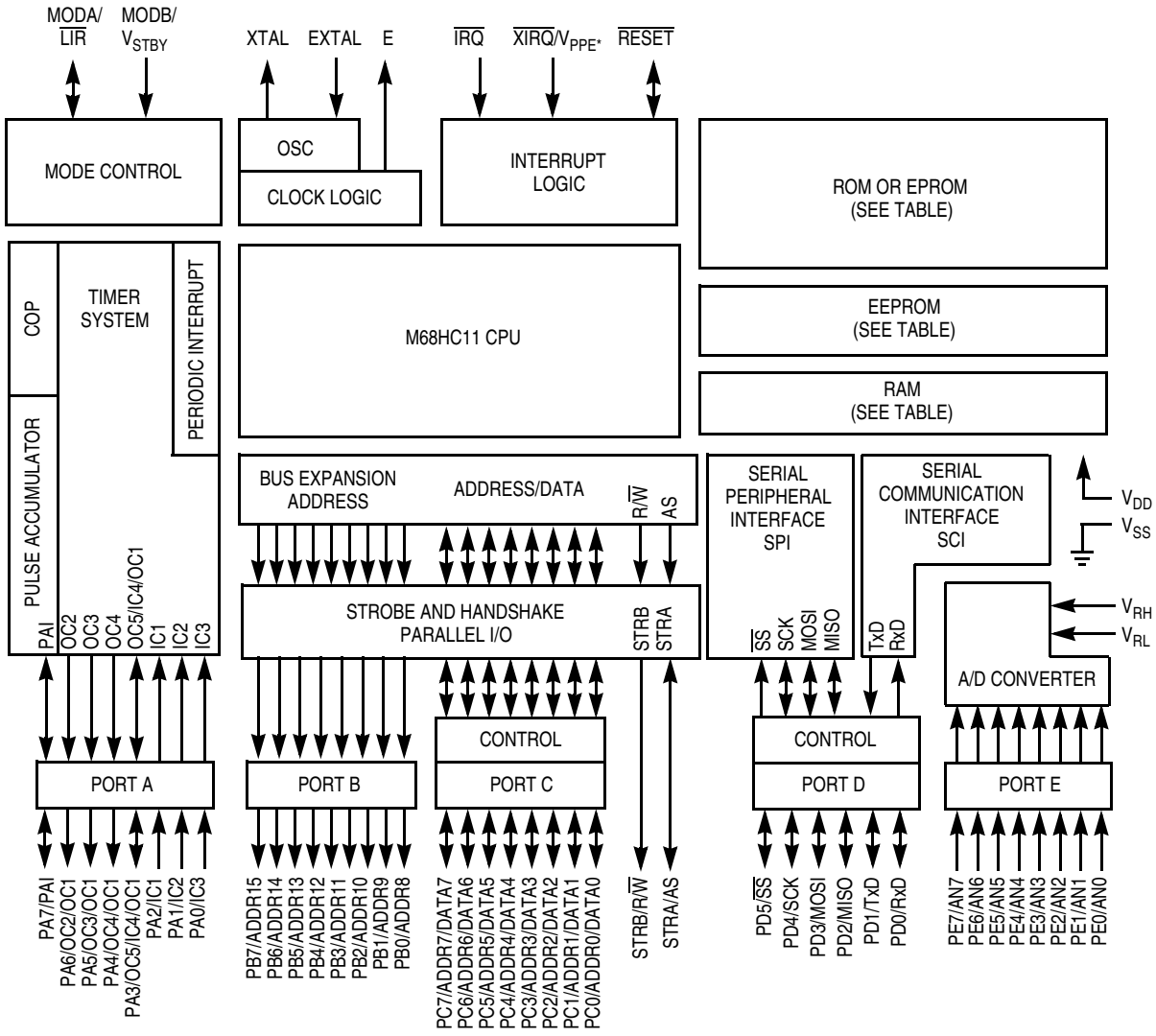
See Figure 1-1 for a functional diagram of the E-series MCUs. Differences among devices are noted in the table accompanying Figure 1-1.

1.4 Pin Descriptions

M68HC11 E-series MCUs are available packaged in:

- 52-pin plastic-leaded chip carrier (PLCC)
- 52-pin windowed ceramic leaded chip carrier (CLCC)
- 52-pin plastic thin quad flat pack, 10 mm x 10 mm (TQFP)
- 64-pin quad flat pack (QFP)
- 48-pin plastic dual in-line package (DIP), MC68HC811E2 only
- 56-pin plastic shrink dual in-line package, .070-inch lead spacing (SDIP)

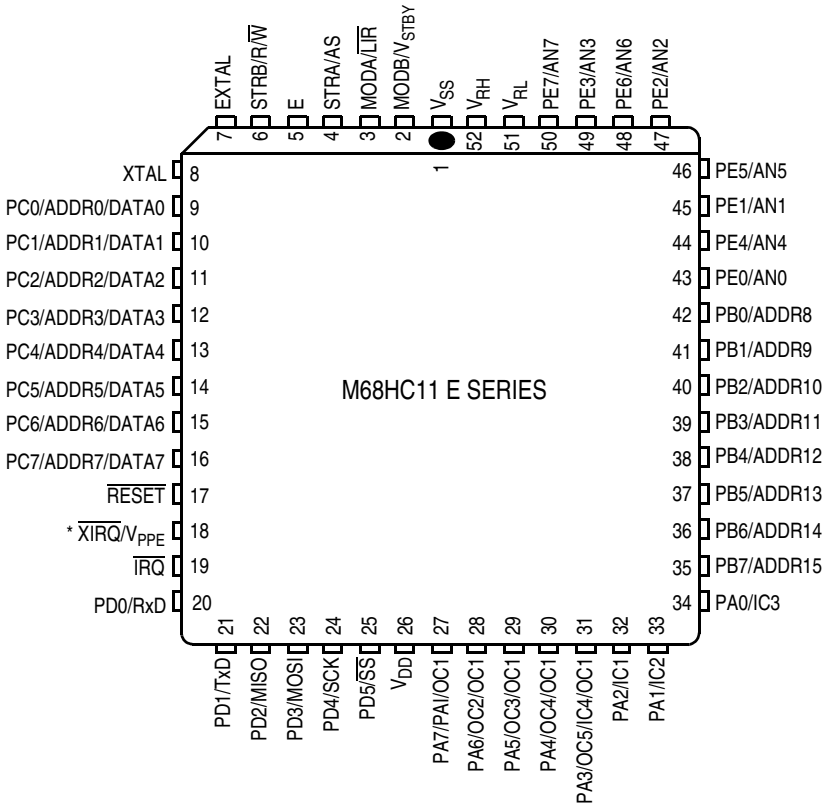
Most pins on these MCUs serve two or more functions, as described in the following paragraphs. Refer to Figure 1-2, Figure 1-3, Figure 1-4, Figure 1-5, and Figure 1-6 which show the M68HC11 E-series pin assignments for the PLCC/CLCC, QFP, TQFP, SDIP, and DIP packages.



DEVICE	RAM	ROM	EPROM	EEPROM
MC68HC11E0	512	—	—	—
MC68HC11E1	512	—	—	512
MC68HC11E9	512	12 K	—	512
MC68HC711E9	512	—	12 K	512
MC68HC11E20	768	20 K	—	512
MC68HC711E20	768	—	20 K	512
MC68HC811E2	256	—	—	2048

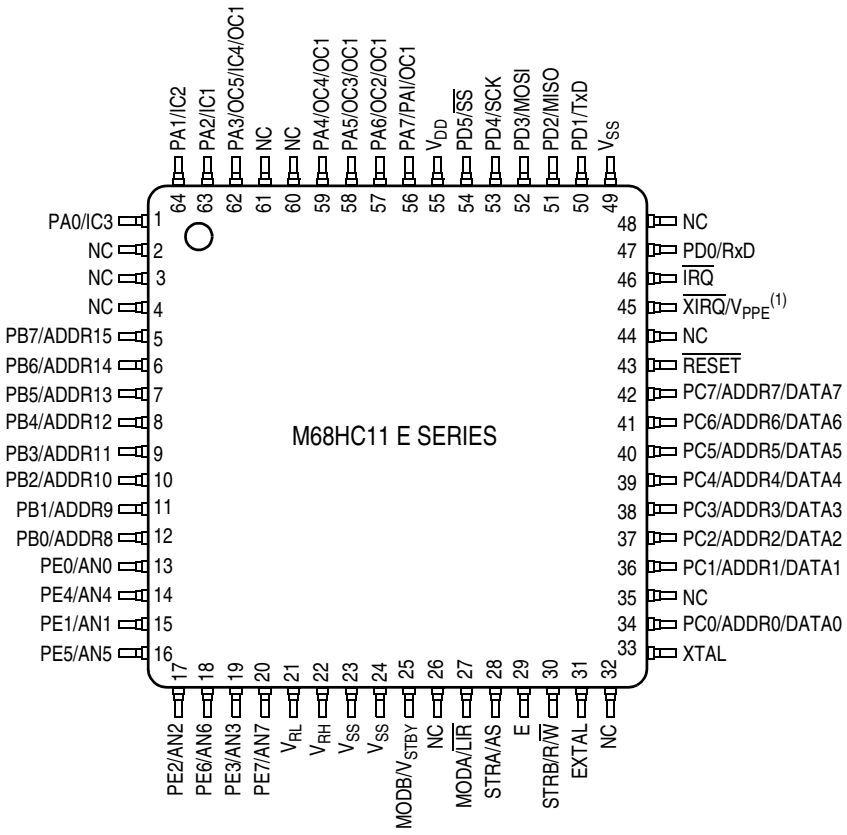
* V_{PPE} applies only to devices with EPROM/OTPROM.

Figure 1-1. M68HC11 E-Series Block Diagram



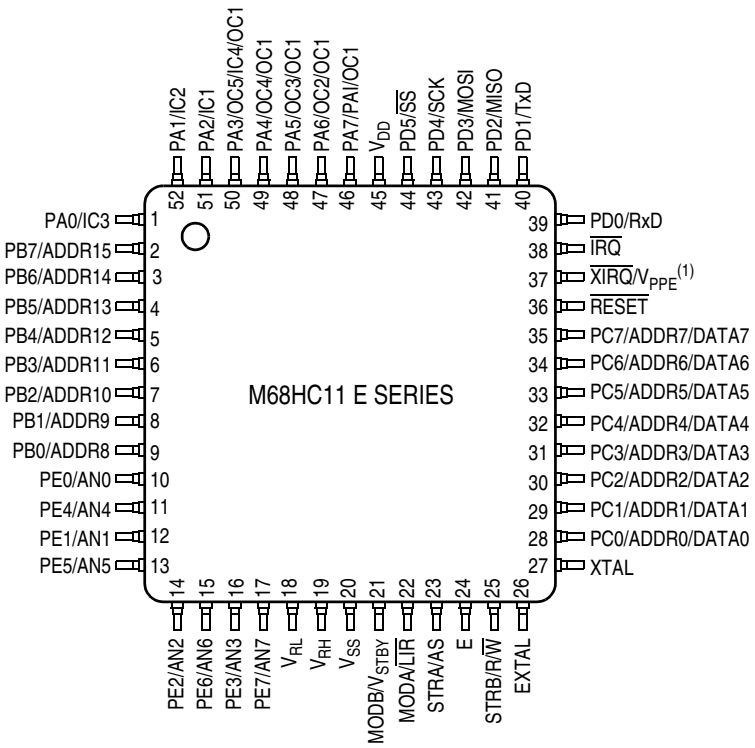
* V_{PPE} applies only to devices with EPROM/OTPROM.

Figure 1-2. Pin Assignments for 52-Pin PLCC and CLCC



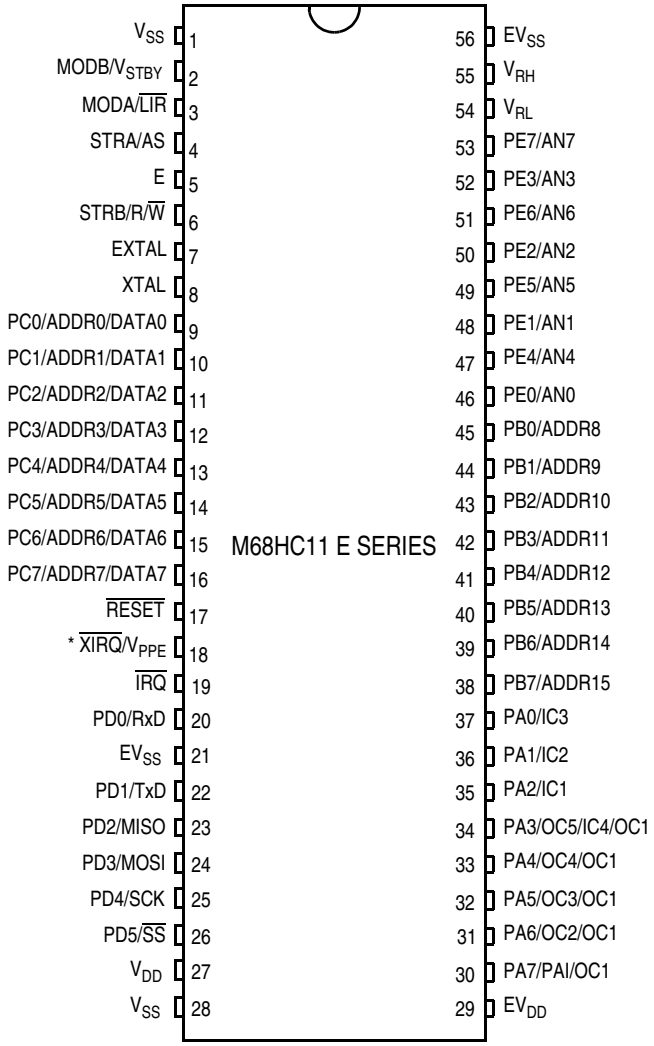
1. V_{PPE} applies only to devices with EPROM/OTPROM.

Figure 1-3. Pin Assignments for 64-Pin QFP



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

Figure 1-4. Pin Assignments for 52-Pin TQFP



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

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

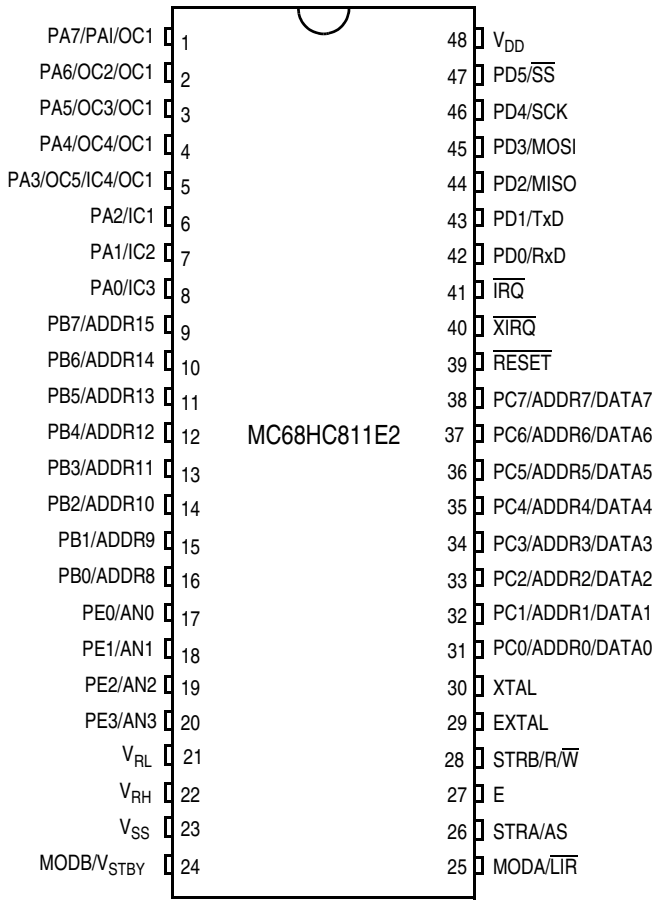


Figure 1-6. Pin Assignments for 48-Pin DIP (MC68HC811E2)

1.4.1 V_{DD} and V_{SS}

Power is supplied to the MCU through V_{DD} and V_{SS}. V_{DD} is the power supply, V_{SS} is ground. The MCU operates from a single 5-volt (nominal) power supply. Low-voltage devices in the E series operate at 3.0–5.5 volts.

Very fast signal transitions occur on the MCU pins. The short rise and fall times place high, short duration current demands on the power supply. To prevent noise problems, provide good power supply bypassing at the MCU. Also, use bypass capacitors that have good

high-frequency characteristics and situate them as close to the MCU as possible. Bypass requirements vary, depending on how heavily the MCU pins are loaded.

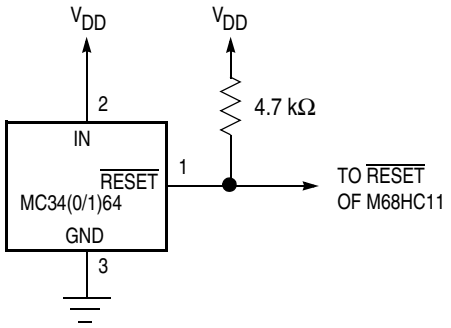


Figure 1-7. External Reset Circuit

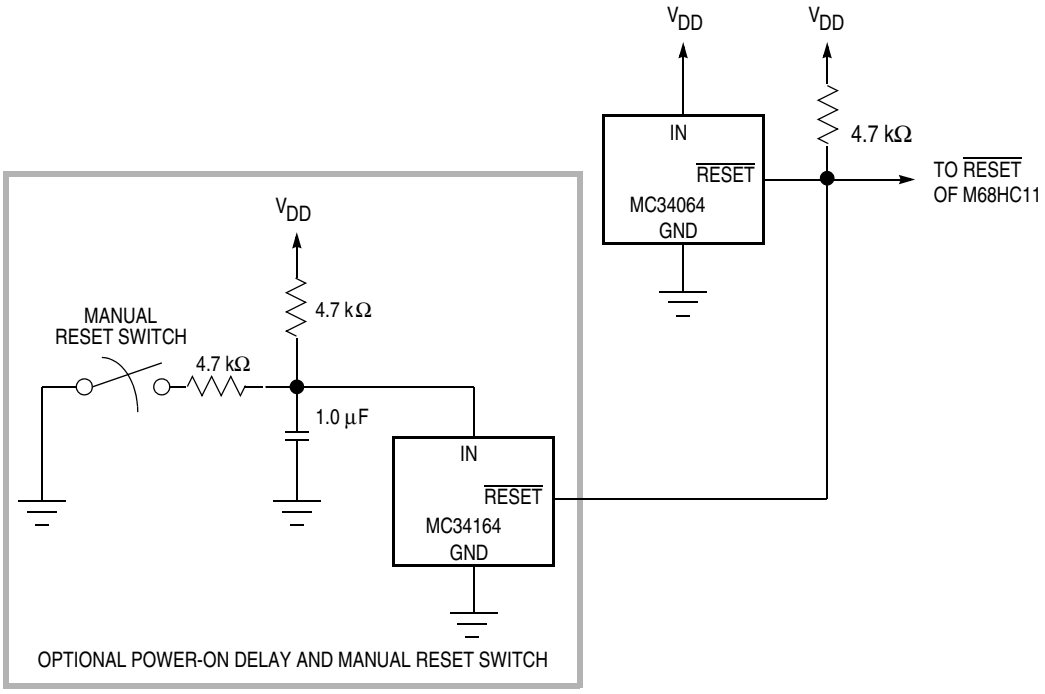


Figure 1-8. External Reset Circuit with Delay

1.4.2 $\overline{\text{RESET}}$

A bidirectional control signal, $\overline{\text{RESET}}$, acts as an input to initialize the MCU to a known startup state. It also acts as an open-drain output to indicate that an internal failure has been detected in either the clock monitor or computer operating properly (COP) watchdog circuit. 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 a reset has occurred. See Figure 1-7 and Figure 1-8.

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.

Because the CPU is not able to fetch and execute instructions properly when V_{DD} falls below the minimum operating voltage level, reset must be controlled. A low-voltage inhibit (LVI) circuit is required primarily for protection of EEPROM contents. However, since the configuration register (CONFIG) value is read from the EEPROM, protection is required even if the EEPROM array is not being used.

Presently, there are several economical ways to solve this problem. For example, two good external components for LVI reset are:

1. The Seiko S0854HN (or other S805 series devices):
   a. Extremely low power (2 µA)
   a. TO-92 package
   a. Limited temperature range, –20°C to +70°C
   a. Available in various trip-point voltage ranges
2. The Freescale MC34064:
   a. TO-92 or SO-8 package
   a. Draws about 300 µA
   a. Temperature range –40°C to 85°C
   a. Well controlled trip point
   a. Inexpensive

Refer to Chapter 5 Resets and Interrupts for further information.

1.4.3 Crystal Driver and External Clock Input (XTAL and EXTAL)

These two pins provide the interface for either a crystal or a CMOS- compatible clock to control the internal clock generator circuitry. The frequency applied to these pins is four times higher than the desired E-clock rate.

The XTAL pin must be left unterminated when an external CMOS- compatible clock input is connected to the EXTAL pin. The XTAL output is normally intended to drive only a crystal. Refer to Figure 1-9 and Figure 1-10.

CAUTION

In all cases, use caution around the oscillator pins. Load capacitances shown in the oscillator circuit are specified by the crystal manufacturer and should include all stray layout capacitances.

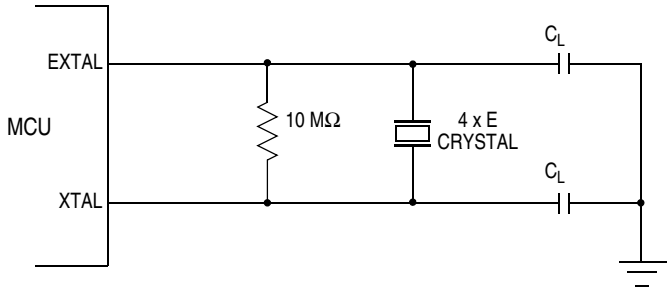


Figure 1-9. Common Parallel Resonant Crystal Connections

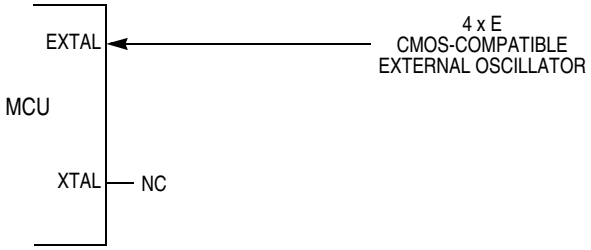


Figure 1-10. External Oscillator Connections

1.4.4 E-Clock Output (E)

E is the output connection for the internally generated E clock. The signal from E is used as a timing reference. The frequency of the E-clock output is one fourth that of the input frequency at the XTAL and EXTAL pins. When E-clock output is low, an internal process is taking place. When it is high, data is being accessed.

All clocks, including the E clock, are halted when the MCU is in stop mode. To reduce RFI emissions, the E-clock output of most E-series devices can be disabled while operating in single-chip modes.

The E-clock signal is always enabled on the MC68HC811E2.

1.4.5 Interrupt Request ($\overline{IRQ}$)

The $\overline{IRQ}$ input provides a means of applying asynchronous interrupt requests to the MCU. Either negative edge-sensitive triggering or level-sensitive triggering is program selectable (OPTION register). $\overline{IRQ}$ is always configured to level-sensitive triggering at reset. When using $\overline{IRQ}$ in a level-sensitive wired-OR configuration, connect an external pullup resistor, typically 4.7 kΩ, to V_{DD}.

1.4.6 Non-Maskable Interrupt ($\overline{XIRQ}/V_{PPE}$)

The $\overline{XIRQ}$ input provides a means of requesting a non-maskable interrupt after reset initialization. During reset, the X bit in the condition code register (CCR) is set and any interrupt is masked until MCU software enables it. Because the $\overline{XIRQ}$ input is level-sensitive, it can be connected to a multiple-source wired-OR network with an external pullup resistor to V_{DD}. $\overline{XIRQ}$ is often used as a power loss detect interrupt.

Whenever $\overline{XIRQ}$ or $\overline{IRQ}$ is used with multiple interrupt sources each source must drive the interrupt input with an open-drain type of driver to avoid contention between outputs.

NOTE

$\overline{IRQ}$ must be configured for level-sensitive operation if there is more than one source of $\overline{IRQ}$ interrupt.

There should be a single pullup resistor near the MCU interrupt input pin (typically 4.7 kΩ). There must also be an interlock mechanism at each interrupt source so that the source holds the interrupt line low until the MCU recognizes and acknowledges the interrupt request. If one or more interrupt sources are still pending after the MCU services a request, the interrupt line will still be held low and the MCU will be interrupted again as soon as the interrupt mask bit in the MCU is cleared (normally upon return from an interrupt). Refer to Chapter 5 Resets and Interrupts.

V_{PPE} is the input for the 12-volt nominal programming voltage required for EPROM/OTPROM programming. On devices without EPROM/OTPROM, this pin is only an $\overline{XIRQ}$ input.

CAUTION

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

1.4.7 MODA and MODB (MODA/$\overline{LIR}$ and MODB/V_{STBY})

During reset, MODA and MODB select one of the four operating modes:

- Single-chip mode
- Expanded mode
- Test mode
- Bootstrap mode

Refer to Chapter 2 Operating Modes and On-Chip Memory.

After the operating mode has been selected, the load instruction register ($\overline{LIR}$) pin provides an open-drain output to indicate that execution of an instruction has begun. A series of E-clock cycles occurs during execution of each instruction. The $\overline{LIR}$ signal goes low during the first E-clock cycle of each instruction (opcode fetch). This output is provided for assistance in program debugging.

The V_{STBY} pin is used to input random-access memory (RAM) standby power. When the voltage on this pin is more than one MOS threshold (about 0.7 volts) above the V_{DD} voltage, the internal RAM and part of the reset logic are powered from this signal rather than the V_{DD} input. This allows RAM contents to be retained without V_{DD} power applied to the MCU. Reset must be driven low before V_{DD} is removed and must remain low until V_{DD} has been restored to a valid level.

1.4.8 V_{RL} and V_{RH}

These two inputs provide the reference voltages for the analog-to-digital (A/D) converter circuitry:

- V_{RL} is the low reference, typically 0 Vdc.
- V_{RH} is the high reference.

For proper A/D converter operation:

- V_{RH} should be at least 3 Vdc greater than V_{RL}.
- V_{RL} and V_{RH} should be between V_{SS} and V_{DD}.

1.4.9 STRA/AS

The strobe A (STRA) and address strobe (AS) pin performs either of two separate functions, depending on the operating mode:

- In single-chip mode, STRA performs an input handshake (strobe input) function.
- In the expanded multiplexed mode, AS provides an address strobe function.

AS can be used to demultiplex the address and data signals at port C. Refer to Chapter 2 Operating Modes and On-Chip Memory.

1.4.10 STRB/R/$\overline{W}$

The strobe B (STRB) and read/write (R/$\overline{W}$) pin act as either an output strobe or as a data bus direction indicator, depending on the operating mode.

In single-chip operating mode, STRB acts as a programmable strobe for handshake with other parallel devices. Refer to Chapter 6 Parallel Input/Output (I/O) Ports for further information.

In expanded multiplexed operating mode, R/$\overline{W}$ is used to indicate the direction of transfers on the external data bus. A low on the R/$\overline{W}$ pin indicates data is being written to the external data bus. A high on this pin indicates that a read cycle is in progress. R/$\overline{W}$ stays low during consecutive data bus write cycles, such as a double-byte store. It is possible for data to be driven out of port C, if internal read visibility (IRV) is enabled and an internal address is read, even though R/$\overline{W}$ is in a high-impedance state. Refer to Chapter 2 Operating Modes and On-Chip Memory for more information about IRVNE (internal read visibility not E).

1.4.11 Port Signals

Port pins have different functions in different operating modes. Pin functions for port A, port D, and port E are independent of operating modes. Port B and port C, however, are affected by operating mode. Port B provides eight general-purpose output signals in single-chip operating modes. When the microcontroller is in expanded multiplexed operating mode, port B pins are the eight high-order address lines.

Port C provides eight general-purpose input/output signals when the MCU is in the single-chip operating mode. When the microcontroller is in the expanded multiplexed operating mode, port C pins are a multiplexed address/data bus.

Refer to Table 1-1 for a functional description of the 40 port signals within different operating modes. Terminate unused inputs and input/output (I/O) pins configured as inputs high or low.

1.4.12 Port A

In all operating modes, port A can be configured for three timer input capture (IC) functions and four timer output compare (OC) functions. An additional pin can be configured as either the fourth IC or the fifth OC. Any port A pin that is not currently being used for a timer function can be used as either a general-purpose input or output line. Only port A pins PA7 and PA3 have an associated data direction control bit that allows the pin to be selectively configured as input or output. Bits DDRA7 and DDRA3 located in PACTL register control data direction for PA7 and PA3, respectively. All other port A pins are fixed as either input or output.

PA7 can function as general-purpose I/O or as timer output compare for OC1. PA7 is also the input to the pulse accumulator, even while functioning as a general-purpose I/O or an OC1 output.

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	

PA6–PA4 serve as either general-purpose outputs, timer input captures, or timer output compare 2–4. In addition, PA6–PA4 can be controlled by OC1.

PA3 can be a general-purpose I/O pin or a timer IC/OC pin. Timer functions associated with this pin include OC1 and IC4/OC5. IC4/OC5 is software selectable as either a fourth input capture or a fifth output compare. PA3 can also be configured to allow OC1 edges to trigger IC4 captures.

PA2–PA0 serve as general-purpose inputs or as IC1–IC3.

PORTA can be read at any time. Reads of pins configured as inputs return the logic level present on the pin. Pins configured as outputs return the logic level present at the pin driver input. If written, PORTA stores the data in an internal latch, bits 7 and 3. It drives the pins only if they are configured as outputs. Writes to PORTA do not change the pin state when pins are configured for timer input captures or output compares. Refer to Chapter 6 Parallel Input/Output (I/O) Ports.

1.4.13 Port B

During single-chip operating modes, all port B pins are general-purpose output pins. During MCU reads of this port, the level sensed at the input side of the port B output drivers is read. Port B can also be used in simple strobed output mode. In this mode, an output pulse appears at the STRB signal each time data is written to port B.

In expanded multiplexed operating modes, all of the port B pins act as high order address output signals. During each MCU cycle, bits 15–8 of the address bus are output on the PB7–PB0 pins. The PORTB register is treated as an external address in expanded modes.

1.4.14 Port C

While in single-chip operating modes, all port C pins are general-purpose I/O pins. Port C inputs can be latched into an alternate PORTCL register by providing an input transition to the STRA signal. Port C can also be used in full handshake modes of parallel I/O where the STRA input and STRB output act as handshake control lines.

When in expanded multiplexed modes, all port C pins are configured as multiplexed address/data signals. During the address portion of each MCU cycle, bits 7–0 of the address are output on the PC7–PC0 pins. During the data portion of each MCU cycle (E high), PC7–PC0 are bidirectional data signals, DATA7–DATA0. The direction of data at the port C pins is indicated by the R/$\overline{W}$ signal.

The CWOM control bit in the PIOC register disables the port C P-channel output driver. CWOM simultaneously affects all eight bits of port C. Because the N-channel driver is not affected by CWOM, setting CWOM causes port C to become an open-drain type output port suitable for wired-OR operation.

In wired-OR mode:

- When a port C bit is at logic level 0, it is driven low by the N-channel driver.
- When a port C bit is at logic level 1, the associated pin has high-impedance, as neither the N-channel nor the P-channel devices are active.

It is customary to have an external pullup resistor on lines that are driven by open-drain devices. Port C can only be configured for wired-OR operation when the MCU is in single-chip mode. Refer to Chapter 6 Parallel Input/Output (I/O) Ports for additional information about port C functions.

1.4.15 Port D

Pins PD5–PD0 can be used for general-purpose I/O signals. These pins alternately serve as the serial communication interface (SCI) and serial peripheral interface (SPI) signals when those subsystems are enabled.

- PD0 is the receive data input (RxD) signal for the SCI.
- PD1 is the transmit data output (TxD) signal for the SCI.
- PD5–PD2 are dedicated to the SPI:
  - PD2 is the master in/slave out (MISO) signal.
  - PD3 is the master out/slave in (MOSI) signal.
  - PD4 is the serial clock (SCK) signal.
  - PD5 is the slave select ($\overline{SS}$) input.

1.4.16 Port E

Use port E for general-purpose or analog-to-digital (A/D) inputs.

CAUTION

If high accuracy is required for A/D conversions, avoid reading port E during sampling, as small disturbances can reduce the accuracy of that result.

Chapter 2
Operating Modes and On-Chip Memory

2.1 Introduction

This section contains information about the operating modes and the on-chip memory for M68HC11 E-series MCUs. Except for a few minor differences, operation is identical for all devices in the E series. Differences are noted where necessary.

2.2 Operating Modes

The values of the mode select inputs MODB and MODA during reset determine the operating mode. Single-chip and expanded multiplexed are the normal modes.

- In single-chip mode only on-chip memory is available.
- Expanded mode, however, allows access to external memory.

Each of the two normal modes is paired with a special mode:

- Bootstrap, a variation of the single-chip mode, is a special mode that executes a bootloader program in an internal bootstrap ROM.
- Test is a special mode that allows privileged access to internal resources.

2.2.1 Single-Chip Mode

In single-chip mode, ports B and C and strobe pins A (STRA) and B (STRB) are available for general-purpose parallel input/output (I/O). In this mode, all software needed to control the MCU is contained in internal resources. If present, read-only memory (ROM) and/or erasable, programmable read-only memory (EPROM) will always be enabled out of reset, ensuring that the reset and interrupt vectors will be available at locations $FFC0–$FFFF.

NOTE

For the MC68HC811E2, the vector locations are the same; however, they are contained in the 2048-byte EEPROM array.

2.2.2 Expanded Mode

In expanded operating mode, the MCU can access the full 64-Kbyte address space. The space includes:

- The same on-chip memory addresses used for single-chip mode
- Addresses for external peripherals and memory devices

The expansion bus is made up of ports B and C, and control signals AS (address strobe) and R/$\overline{W}$ (read/write). R/$\overline{W}$ and AS allow the low-order address and the 8-bit data bus to be multiplexed on the same pins. During the first half of each bus cycle address information is present. During the second half of each bus cycle the pins become the bidirectional data bus. AS is an active-high latch enable signal for an external address latch. Address information is allowed through the transparent latch while AS is high and is latched when AS drives low.

The address, R/$\overline{W}$, and AS signals are active and valid for all bus cycles, including accesses to internal memory locations. The E clock is used to enable external devices to drive data onto the internal data bus during the second half of a read bus cycle (E clock high). R/$\overline{W}$ controls the direction of data transfers. R/$\overline{W}$ drives low when data is being written to the internal data bus. R/$\overline{W}$ will remain low during consecutive data bus write cycles, such as when a double-byte store occurs.

Refer to Figure 2-1.

NOTE

The write enable signal for an external memory is the NAND of the E clock and the inverted R/$\overline{W}$ signal.

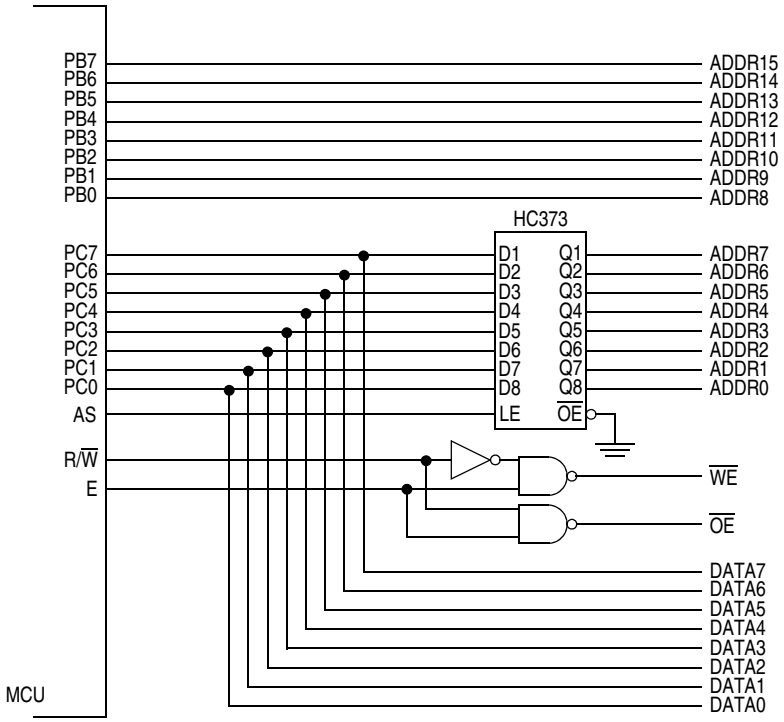


Figure 2-1. Address/Data Demultiplexing

2.2.3 Test Mode

Test mode, a variation of the expanded mode, is primarily used during Freescale's internal production testing; however, it is accessible for programming the configuration (CONFIG) register, programming calibration data into electrically erasable, programmable read-only memory (EEPROM), and supporting emulation and debugging during development.

2.2.4 Bootstrap Mode

When the MCU is reset in special bootstrap mode, a small on-chip read-only memory (ROM) is enabled at address $BF00–$BFFF. The ROM contains a bootloader program and a special set of interrupt and reset vectors. The MCU fetches the reset vector, then executes the bootloader.

Bootstrap mode is a special variation of the single-chip mode. Bootstrap mode allows special-purpose programs to be entered into internal random-access memory (RAM). When bootstrap mode is selected at reset, a small bootstrap ROM becomes present in the memory map. Reset and interrupt vectors are

located in this ROM at $BFC0–$BFFF. The bootstrap ROM contains a small program which initializes the serial communications interface (SCI) and allows the user to download a program into on-chip RAM. The size of the downloaded program can be as large as the size of the on-chip RAM. After a 4-character delay, or after receiving the character for the highest address in RAM, control passes to the loaded program at $0000. Refer to Figure 2-2, Figure 2-3, Figure 2-4, Figure 2-5, and Figure 2-6.

Use of an external pullup resistor is required when using the SCI transmitter pin because port D pins are configured for wired-OR operation by the bootloader. In bootstrap mode, the interrupt vectors are directed to RAM. This allows the use of interrupts through a jump table. Refer to the application note AN1060 entitled *M68HC11 Bootstrap Mode, that is included in this data book*.

2.3 Memory Map

The operating mode determines memory mapping and whether external addresses can be accessed. Refer to Figure 2-2, Figure 2-3, Figure 2-4, Figure 2-5, and Figure 2-6, which illustrate the memory maps for each of the three families comprising the M68HC11 E series of MCUs.

Memory locations for on-chip resources are the same for both expanded and single-chip modes. Control bits in the configuration (CONFIG) register allow EPROM and EEPROM (if present) to be disabled from the memory map. The RAM is mapped to $0000 after reset. It can be placed at any 4-Kbyte boundary ($x000) by writing an appropriate value to the RAM and I/O map register (INIT). The 64-byte register block is mapped to $1000 after reset and also can be placed at any 4-Kbyte boundary ($x000) by writing an appropriate value to the INIT register. If RAM and registers are mapped to the same boundary, the first 64 bytes of RAM will be inaccessible.

Refer to Figure 2-7, which details the MCU register and control bit assignments. Reset states shown are for single-chip mode only.

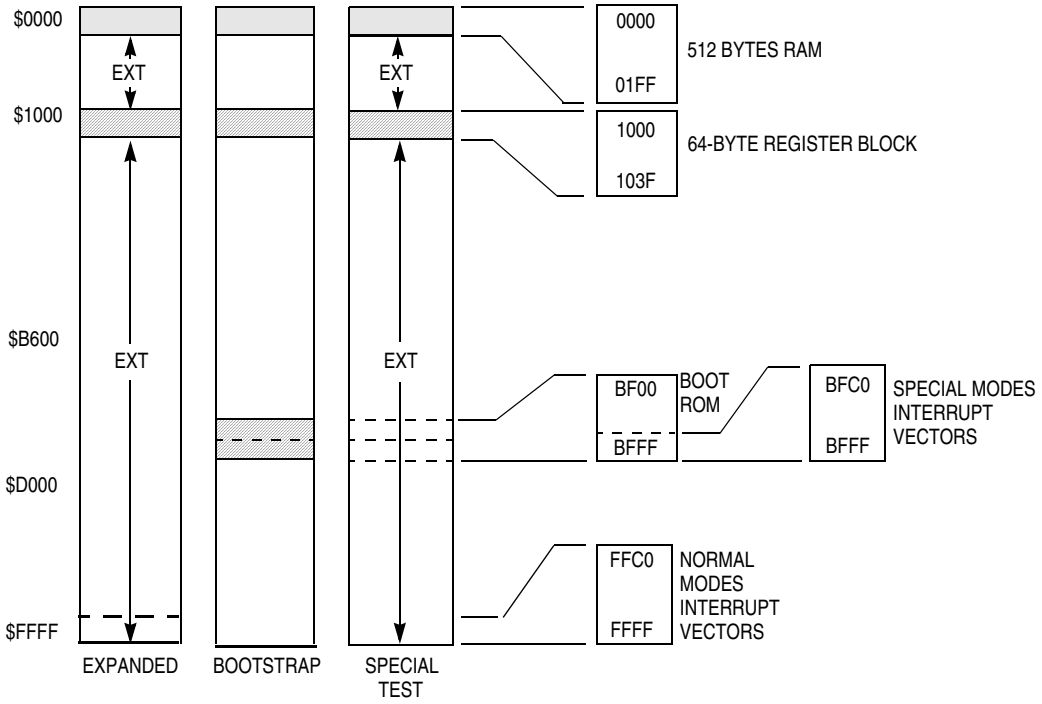


Figure 2-2. Memory Map for MC68HC11E0

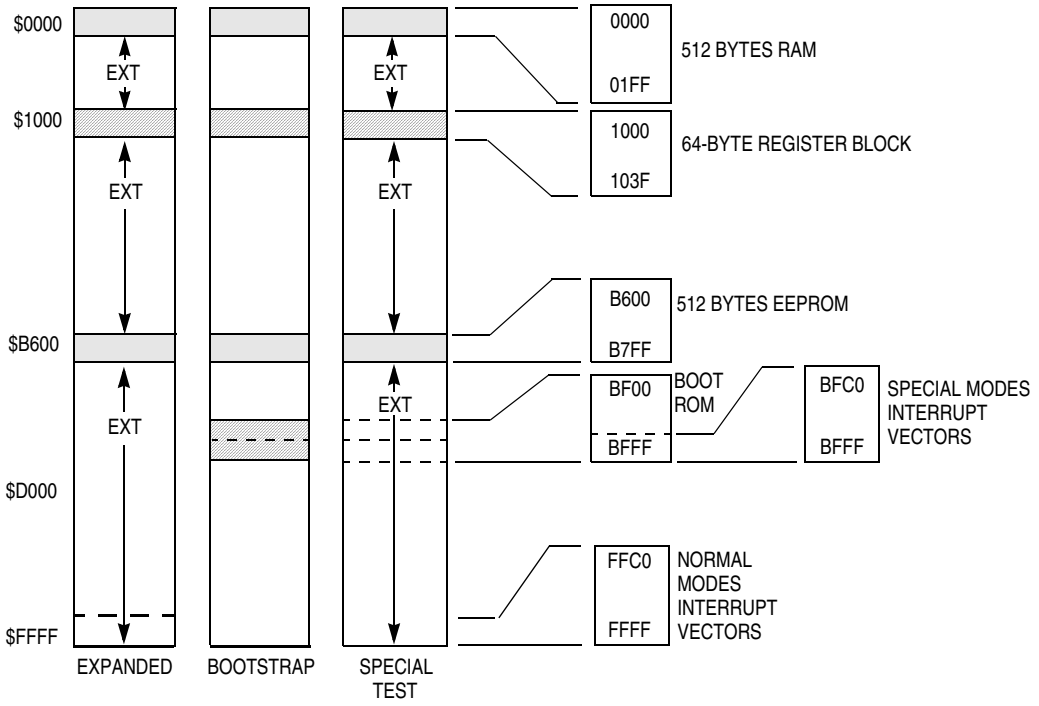


Figure 2-3. Memory Map for MC68HC11E1

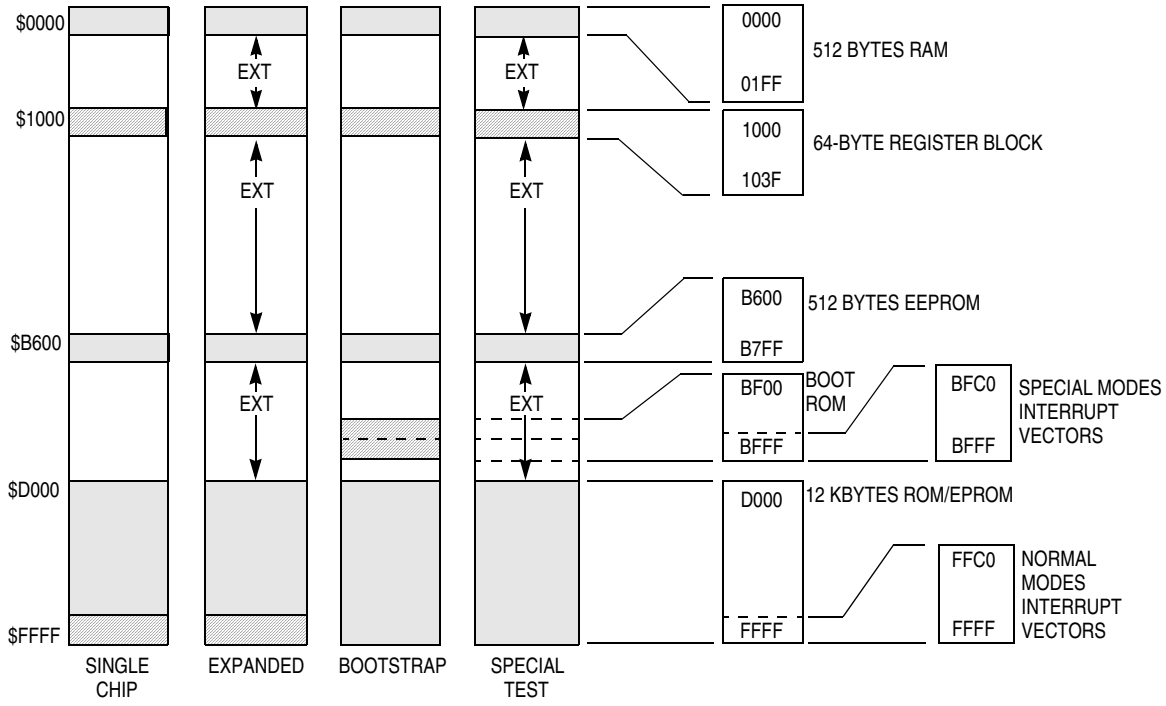
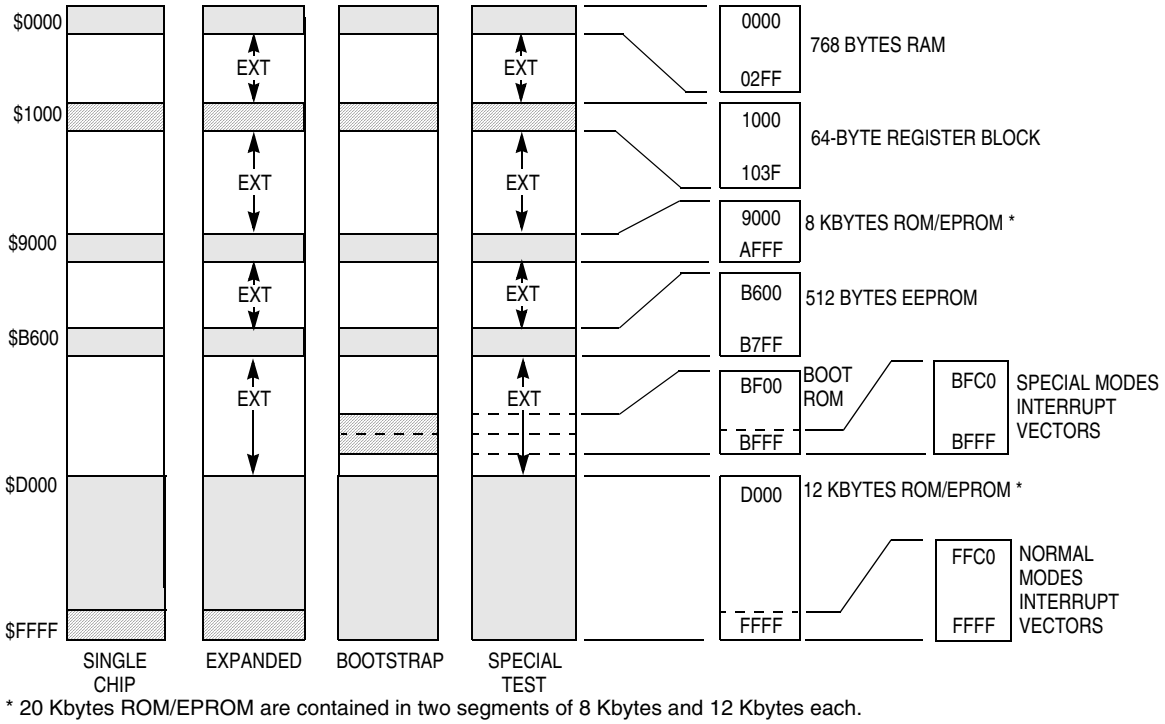


Figure 2-4. Memory Map for MC68HC(7)11E9



* 20 Kbytes ROM/EPROM are contained in two segments of 8 Kbytes and 12 Kbytes each.

Figure 2-5. Memory Map for MC68HC(7)11E20

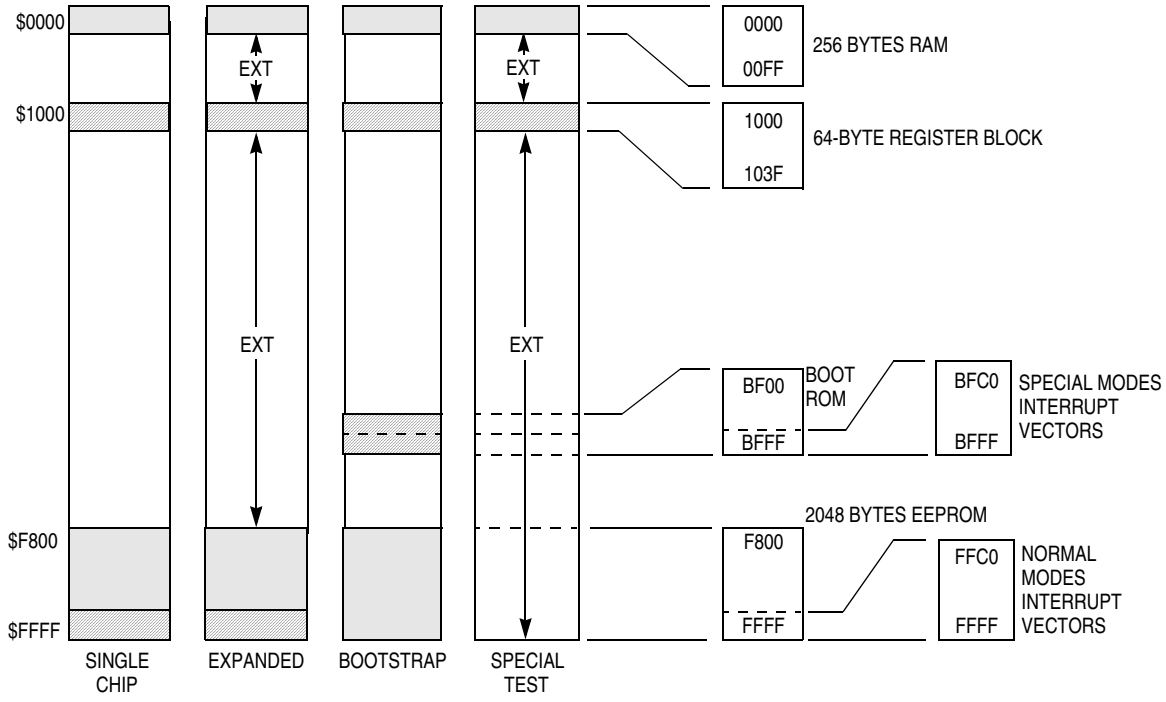


Figure 2-6. Memory Map for MC68HC811E2

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$1000	Port A Data Register (PORTA) See page 98.	Read: Write:	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
		Reset:	I	0	0	0	I	I	I	I
$1001	Reserved		R	R	R	R	R	R	R	R
$1002	Parallel I/O Control Register (PIOC) See page 102.	Read: Write:	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB
		Reset:	0	0	0	0	0	U	1	1
$1003	Port C Data Register (PORTC) See page 99.	Read: Write:	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
		Reset:	Indeterminate after reset							
$1004	Port B Data Register (PORTB) See page 99.	Read: Write:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
		Reset:	0	0	0	0	0	0	0	0
$1005	Port C Latched Register (PORTCL) See page 99.	Read: Write:	PCL7	PCL6	PCL5	PCL4	PCL3	PCL2	PCL1	PCL0
		Reset:	Indeterminate after reset							
$1006	Reserved		R	R	R	R	R	R	R	R
$1007	Port C Data Direction Register (DDRC) See page 100.	Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Reset:	0	0	0	0	0	0	0	0
$1008	Port D Data Register (PORTD) See page 100.	Read: Write:	0	0	PD5	PD4	PD3	PD2	PD1	PD0
		Reset:	U	U	I	I	I	I	I	I
$1009	Port D Data Direction Register (DDRD) See page 100.	Read: Write:			DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0
$100A	Port E Data Register (PORTE) See page 101.	Read: Write:	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
		Reset:	Indeterminate after reset							
$100B	Timer Compare Force Register (CFORC) See page 135.	Read: Write:	FOC1	FOC2	FOC3	FOC4	FOC5			
		Reset:	0	0	0	0	0	0	0	0
$100C	Output Compare 1 Mask Register (OC1M) See page 136.	Read: Write:	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3			
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

I = Indeterminate after reset

Figure 2-7. Register and Control Bit Assignments (Sheet 1 of 6)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$100D	Output Compare 1 Data Register (OC1D) See page 136.	Read:	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3			
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$100E	Timer Counter Register High (TCNTH) See page 137.	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
$100F	Timer Counter Register Low (TCNTL) See page 137.	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
$1010	Timer Input Capture 1 Register High (TIC1H) See page 132.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$1011	Timer Input Capture 1 Register Low (TIC1L) See page 132.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1012	Timer Input Capture 2 Register High (TIC2H) See page 132.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$1013	TImer Input Capture 2 Register Low (TIC2L) See page 132.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1014	Timer Input Capture 3 Register High (TIC3H) See page 132.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$1015	Timer Input Capture 3 Register Low (TIC3L) See page 132.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1016	Timer Output Compare 1 Register High (TOC1H) See page 134.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$1017	Timer Output Compare 1 Register Low (TOC1L) See page 134.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$1018	Timer Output Compare 2 Register High (TOC2H) See page 134.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1

= Unimplemented R = Reserved U = Unaffected

I = Indeterminate after reset

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$1019	Timer Output Compare 2 Register Low (TOC2L) See page 134.	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
$101A	Timer Output Compare 3 Register High (TOC3H) See page 135.	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
$101B	Timer Output Compare 3 Register Low (TOC3L) See page 135.	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
$101C	Timer Output Compare 4 Register High (TOC4H) See page 135.	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
$101D	Timer Output Compare 4 Register Low (TOC4L) See page 135.	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
$101E	Timer Input Capture 4/Output Compare 5 Register High (TI4/O5) See page 133.	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
$101F	Timer Input Capture 4/Output Compare 5 Register Low (TI4/O5) See page 133.	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
$1020	Timer Control Register 1 (TCTL1) See page 137.	Read: Write:	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5
		Reset:	0	0	0	0	0	0	0	0
$1021	Timer Control Register 2 (TCTL2) See page 131.	Read: Write:	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A
		Reset:	0	0	0	0	0	0	0	0
$1022	Timer Interrupt Mask 1 Register (TMSK1) See page 138.	Read: Write:	OC1I	OC2I	OC3I	OC4I	I4/O5I	IC1I	IC2I	IC3I
		Reset:	0	0	0	0	0	0	0	0
$1023	Timer Interrupt Flag 1 (TFLG1) See page 138.	Read: Write:	OC1F	OC2F	OC3F	OC4F	I4/O5F	IC1F	IC2F	IC3F
		Reset:	0	0	0	0	0	0	0	0
$1024	Timer Interrupt Mask 2 Register (TMSK2) See page 139.	Read: Write:	TOI	RTII	PAOVI	PAII			PR1	PR0
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

I = Indeterminate after reset

Figure 2-7. Register and Control Bit Assignments (Sheet 3 of 6)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$1025	Timer Interrupt Flag 2 (TFLG2) See page 142.	Read:	TOF	RTIF	PAOVF	PAIF				
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$1026	Pulse Accumulator Control Register (PACTL) See page 142.	Read:	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$1027	Pulse Accumulator Count Register (PACNT) See page 146.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1028	Serial Peripheral Control Register (SPCR) See page 123.	Read:	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	0	1	U	U
$1029	Serial Peripheral Status Register (SPSR) See page 124.	Read:	SPIF	WCOL		MODF				
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$102A	Serial Peripheral Data I/O Register (SPDR) See page 125.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$102B	Baud Rate Register (BAUD) See page 113.	Read:	TCLR	SCP2(1)	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	U	U	U
$102C	Serial Communications Control Register 1 (SCCR1) See page 110.	Read:	R8	T8		M	WAKE			
		Write:								
		Reset:	I	I	0	0	0	0	0	0
$102D	Serial Communications Control Register 2 (SCCR2) See page 111.	Read:	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$102E	Serial Communications Status Register (SCSR) See page 112.	Read:	TDRE	TC	RDRF	IDLE	OR	NF	FE	
		Write:								
		Reset:	1	1	0	0	0	0	0	0

1. SCP2 adds ÷39 to SCI prescaler and is present only in MC68HC(7)11E20.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$102F	Serial Communications Data Register (SCDR) See page 110.	Read:	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
		Write:								
		Reset:	Indeterminate after reset							
$1030	Analog-to-Digital Control Status Register (ADCTL) See page 62.	Read:	CCF		SCAN	MULT	CD	CC	CB	CA
		Write:								
		Reset:	0	0	Indeterminate after reset					

= Unimplemented R = Reserved U = Unaffected

I = Indeterminate after reset

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$1031	Analog-to-Digital Results Register 1 (ADR1) See page 64.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1032	Analog-to-Digital Results Register 2 (ADR2) See page 64.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1033	Analog-to-Digital Results Register 3 (ADR3) See page 64.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1034	Analog-to-Digital Results Register 4 (ADR4) See page 64.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$1035	Block Protect Register (BPROT) See page 52.	Read:				PTCON	BPRT3	BPRT2	BPRT1	BPRT0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$1036	EPROM Programming Control Register (EPROG)(1) See page 53.	Read:	MBE		ELAT	EXCOL	EXROW	T1	T0	PGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$1037	Reserved		R	R	R	R	R	R	R	R

1. MC68HC711E20 only

$1038	Reserved		R	R	R	R	R	R	R	R
$1039	System Configuration Options Register (OPTION) See page 46.	Read:	ADPU	CSEL	IRQE(1)	DLY(1)	CME		CR1(1)	CR0(1)
		Write:								
		Reset:	0	0	0	1	0	0	0	0
$103A	Arm/Reset COP Timer Circuitry Register (COPRST) See page 81.	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
$103B	EPROM and EEPROM Programming Control Register (PPROG) See page 49.	Read:	ODD	EVEN	ELAT(2)	BYTE	ROW	ERASE	EELAT	EPGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$103C	Highest Priority I Bit Interrupt and Miscellaneous Register (HPRIO) See page 41.	Read:	RBOOT	SMOD	MDA	IRV(NE)	PSEL3	PSEL2	PSEL1	PSEL0
		Write:								
		Reset:	0	0	0	0	0	1	1	0
$103D	RAM and I/O Mapping Register (INIT) See page 45.	Read:	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0
		Write:								
		Reset:	0	0	0	0	0	0	0	1

= Unimplemented R = Reserved U = Unaffected

I = Indeterminate after reset

Figure 2-7. Register and Control Bit Assignments (Sheet 5 of 6)

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

1. Can be written only once in first 64 cycles out of reset in normal modes or at any time during special modes.
2. MC68HC711E9 only
3. MC68HC811E2 only

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

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

2.3.1 RAM and Input/Output Mapping

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

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

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

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

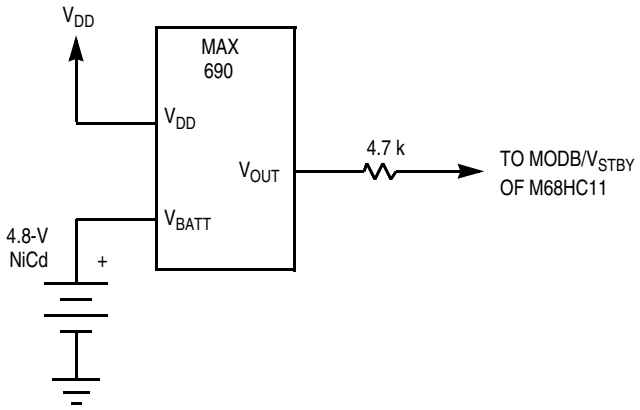


Figure 2-8. RAM Standby MODB/V_{STBY} Connections

The bootloader program is contained in the internal bootstrap ROM. This ROM, which appears as internal memory space at locations $BF00–$BFFF, is enabled only if the MCU is reset in special bootstrap mode.

In expanded modes, the ROM/EPROM/OTPROM (if present) is enabled out of reset and located at the top of the memory map if the ROMON bit in the CONFIG register is set. ROM or EPROM is enabled out of reset in single-chip and bootstrap modes, regardless of the state of ROMON.

For devices with 512 bytes of EEPROM, the EEPROM is located at $B600–$B7FF and has the same read cycle time as the internal ROM. The 512 bytes of EEPROM cannot be remapped to other locations.

For the MC68HC811E2, EEPROM is located at $F800–$FFFF and can be remapped to any 4-Kbyte boundary. EEPROM mapping control bits (EE[3:0] in CONFIG) determine the location of the 2048 bytes of EEPROM and are present only on the MC68HC811E2. Refer to 2.3.3.1 System Configuration Register for a description of the MC68HC811E2 CONFIG register.

EEPROM can be programmed or erased by software and an on-chip charge pump, allowing EEPROM changes using the single V_{DD} supply.

2.3.2 Mode Selection

The four mode variations are selected by the logic states of the MODA and MODB pins during reset. The MODA and MODB logic levels determine the logic state of SMOD and the MDA control bits in the highest priority I-bit interrupt and miscellaneous (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 0. In expanded mode, MODA is normally connected to V_{DD} through a pullup resistor of 4.7 kΩ. The MODA pin also functions as the load instruction register $\overline{LIR}$ pin when the MCU is not in reset. The open-drain active low $\overline{LIR}$ output pin drives low during the first E cycle of each instruction. The MODB pin also functions as standby power input (V_{STBY}), which allows RAM contents to be maintained in absence of V_{DD}.

Refer to Table 2-1, which is a summary of mode pin operation, the mode control bits, and the four operating modes.

Table 2-1. 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	Bootstrap	1	1	0
0	1	Special test	0	1	1

A normal mode is selected when MODB is logic 1 during reset. One of three reset vectors is fetched from address $FFFA–$FFFF, and program execution begins from the address indicated by this vector. If MODB is logic 0 during reset, the special mode reset vector is fetched from addresses $BFFA–$BFFF, and software has access to special test features. Refer to Chapter 5 Resets and Interrupts.

Address: $103C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RBOOT(1)	SMOD(1)	MDA(1)	IRV(NE)(1)	PSEL3	PSEL2	PSEL1	PSEL0
Resets:								
Single chip:	0	0	0	0	0	1	1	0
Expanded:	0	0	1	0	0	1	1	0
Bootstrap:	1	1	0	0	0	1	1	0
Test:	0	1	1	1	0	1	1	0

1. The reset values depend on the mode selected at the RESET pin rising edge.

Figure 2-9. Highest Priority I-Bit Interrupt and Miscellaneous Register (HPRIO)

RBOOT — Read Bootstrap ROM Bit

Valid only when SMOD is set (bootstrap or special test mode); can be written only in special modes

0 = Bootloader ROM disabled and not in map

1 = Bootloader ROM enabled and in map at $BE00–$BFFF

SMOD and MDA — Special Mode Select and Mode Select A Bits

The initial value of SMOD is the inverse of the logic level present on the MODB pin at the rising edge of reset. The initial value of MDA equals the logic level present on the MODA pin at the rising edge of reset. These two bits can be read at any time. They can be written anytime in special modes. MDA can be written only once in normal modes. SMOD cannot be set once it has been cleared.

Input		Mode	Latched at Reset	
MODB	MODA		SMOD	MDA
1	0	Single chip	0	0
1	1	Expanded	0	1

0	0	Bootstrap	1	0
0	1	Special test	1	1

IRV(NE) — Internal Read Visibility (Not E) Bit

IRVNE can be written once in any mode. In expanded modes, IRVNE determines whether IRV is on or off. In special test mode, IRVNE is reset to 1. In all other modes, IRVNE is reset to 0. For the MC68HC811E2, this bit is IRV and only controls the internal read visibility function.

0 = No internal read visibility on external bus

1 = Data from internal reads is driven out the external data bus.

In single-chip modes this bit determines whether the E clock drives out from the chip. For the MC68HC811E2, this bit has no meaning or effect in single-chip and bootstrap modes.

0 = E is driven out from the chip.

1 = E pin is driven low. Refer to the following table.

Mode	IRVNE Out of Reset	E Clock Out of Reset	IRV Out of Reset	IRVNE Affects Only	IRVNE Can Be Written
Single chip	0	On	Off	E	Once
Expanded	0	On	Off	IRV	Once
Bootstrap	0	On	Off	E	Once
Special test	1	On	On	IRV	Once

PSEL[3:0] — Priority Select Bits

Refer to Chapter 5 Resets and Interrupts.

2.3.3 System Initialization

Registers and bits that control initialization and the basic operation of the MCU are protected against writes except under special circumstances. Table 2-2 lists registers that can be written only once after reset or that must be written within the first 64 cycles after reset.

Table 2-2. Write Access Limited Registers

Operating Mode	Register Address	Register Name	Must be Written in First 64 Cycles	Write Anytime
SMOD = 0	$x024	Timer interrupt mask 2 (TMSK2)	Bits [1:0], once only	Bits [7:2]
	$x035	Block protect register (BPROT)	Clear bits, once only	Set bits only
	$x039	System configuration options (OPTION)	Bits [5:4], bits [2:0], once only	Bits [7:6], bit 3
	$x03C	Highest priority I-bit interrupt and miscellaneous (HPRIO)	See HPRIO description	See HPRIO description
	$x03D	RAM and I/O map register (INIT)	Yes, once only	—
SMOD = 1	$x024	Timer interrupt mask 2 (TMSK2)	—	All, set or clear
	$x035	Block protect register (BPROT)	—	All, set or clear
	$x039	System configuration options (OPTION)	—	All, set or clear
	$x03C	Highest priority I-bit interrupt and miscellaneous (HPRIO)	See HPRIO description	See HPRIO description
	$x03D	RAM and I/O map register (INIT)	—	All, set or clear

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)

Address: $103F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EE3	EE2	EE1	EE0	NOSEC	NOCOP		EEON
Write:								
Resets:								
Single chip:	1	1	1	1	U	U	1	1
Bootstrap:	1	1	1	1	U	U(L)	1	1
Expanded:	U	U	U	U	1	U	1	U
Test:	U	U	U	U	1	U(L)	1	0

= 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 2-11. MC68HC811E2 System Configuration Register (CONFIG)

EE[3:0] — EEPROM Mapping Bits

EE[3:0] apply only to MC68HC811E2 and allow the 2048 bytes of EEPROM to be remapped to any 4-Kbyte boundary. See Table 2-3.

Table 2-3. EEPROM Mapping

EE[3:0]	EEPROM Location
0 0 0 0	$0800–$0FFF
0 0 0 1	$1800–$1FFF
0 0 1 0	$2800–$2FFF
0 0 1 1	$3800–$3FFF
0 1 0 0	$4800–$4FFF
0 1 0 1	$5800–$5FFF
0 1 1 0	$6800–$6FFF
0 1 1 1	$7800–$7FFF
1 0 0 0	$8800–$8FFF
1 0 0 1	$9800–$9FFF
1 0 1 0	$A800–$AFFF
1 0 1 1	$B800–$BFFF
1 1 0 0	$C800–$CFFF
1 1 0 1	$D800–$DFFF
1 1 1 0	$E800–$EFFF
1 1 1 1	$F800–$FFFF

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.

Table 2-4. RAM Mapping

RAM[3:0]	Address
0000	$0000–$0xFF
0001	$1000–$1xFF
0010	$2000–$2xFF
0011	$3000–$3xFF
0100	$4000–$4xFF
0101	$5000–$5xFF
0110	$6000–$6xFF
0111	$7000–$7xFF
1000	$8000–$8xFF
1001	$9000–$9xFF
1010	$A000–$AxFF
1011	$B000–$BxFF
1100	$C000–$CxFF
1101	$D000–$DxFF
1110	$E000–$ExFF
1111	$F000–$FxFF

Table 2-5. Register Mapping

REG[3:0]	Address
0000	$0000–$003F
0001	$1000–$103F
0010	$2000–$203F
0011	$3000–$303F
0100	$4000–$403F
0101	$5000–$503F
0110	$6000–$603F
0111	$7000–$703F
1000	$8000–$803F
1001	$9000–$903F
1010	$A000–$A03F
1011	$B000–$B03F
1100	$C000–$C03F
1101	$D000–$D03F
1110	$E000–$E03F
1111	$F000–$F03F

2.3.3.3 System Configuration Options Register

The 8-bit, special-purpose system configuration options register (OPTION) sets internal system configuration options during initialization. The time protected control bits, IRQE, DLY, and CR[1:0], can be written only once after a reset and then they become read-only. This minimizes the possibility of any accidental changes to the system configuration.

Address: $1039

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ADPU	CSEL	IRQE(1)	DLY(1)	CME		CR1(1)	CR0(1)
Reset:	0	0	0	1	0	0	0	0

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

= Unimplemented

Figure 2-13. System Configuration Options Register (OPTION)

ADPU — Analog-to-Digital Converter Power-Up Bit

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

CSEL — Clock Select Bit

Selects alternate clock source for on-chip EEPROM charge pump. Refer to 2.5.1 EEPROM and CONFIG Programming and Erasure for more information on EEPROM use.

CSEL also selects the clock source for the A/D converter, a function discussed in Chapter 3 Analog-to-Digital (A/D) Converter.

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

2.4.1 Programming an Individual EPROM Address

- In this method, the MCU programs its own EPROM by controlling the PPROG register (EPROG in MC68HC711E20). Use these procedures to program the EPROM through the MCU with:
- The ROMON bit set in the CONFIG register
- The 12-volt nominal programming voltage present on the $\overline{\text{XIRQ}}/V_{PPE}$ pin
- The $\overline{\text{IRQ}}$ pin must be pulled high.

NOTE

Any operating mode can be used.

This example applies to all devices with EPROM/OTPROM except for the MC68HC711E20.

```
EPROG   LDAB    #$20
        STAB    $103B   Set ELAT bit in (EPGM = 0) to enable
                        EPROM latches.
        STAA    $0,X    Store data to EPROM address
        LDAB    #$21
        STAB    $103B   Set EPGM bit with ELAT = 1 to enable
                        EPROM programming voltage
        JSR     DLYEP   Delay 2-4 ms
        CLR     $103B   Turn off programming voltage and set
                        to READ mode
```

This example applies only to MC68HC711E20.

```
EPROG   LDAB    #$20
        STAB    $1036   Set ELAT bit (EPGM = 0) to enable
                        EPROM latches.
        STAA    $0,X    Store data to EPROM address
        LDAB    #$21
        STAB    $1036   Set EPGM bit with ELAT = 1 to enable
                        EPROM programming voltage
        JSR     DLYEP   Delay 2-4 ms
        CLR     $1036   Turn off programming voltage and set
                        to READ mode
```

2.4.2 Programming the EPROM with Downloaded Data

When using this method, the EPROM is programmed by software while in the special test or bootstrap modes. User-developed software can be uploaded through the SCI or a ROM-resident EPROM programming utility can be used. The 12-volt nominal programming voltage must be present on the $\overline{\text{XIRQ}}/V_{PPE}$ pin. To use the resident utility, bootload a 3-byte program consisting of a single jump instruction to $BF00. $BF00 is the starting address of a resident EPROM programming utility. The utility program sets the X and Y index registers to default values, then receives programming data from an external host, and puts it in EPROM. The value in IX determines programming delay time. The value in IY is a pointer to the first address in EPROM to be programmed (default = $D000).

When the utility program is ready to receive programming data, it sends the host the $FF character. Then it waits. When the host sees the $FF character, the EPROM programming data is sent, starting with the first location in the EPROM array. After the last byte to be programmed is sent and the corresponding verification data is returned, the programming operation is terminated by resetting the MCU.

For more information, Freescale application note AN1060 entitled M68HC11 Bootstrap Mode has been included at the back of this document.

2.4.3 EPROM and EEPROM Programming Control Register

The EPROM and EEPROM programming control register (PPROG) enables the EPROM programming voltage and controls the latching of data to be programmed.

- For MC68HC711E9, PPROG is also the EEPROM programming control register.
- For the MC68HC711E20, EPROM programming is controlled by the EPROG register and EEPROM programming is controlled by the PPROG register.

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 2-14. EPROM and EEPROM Programming Control Register (PPROG)

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

Refer to 2.5 EEPROM.

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

Refer to 2.5 EEPROM.

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:

a. EPGM enables the high voltage necessary for both EEPROM and EPROM/OTPROM programming.

b. 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: Write:	MBE		ELAT	EXCOL	EXROW	T1	T0	PGM
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

Bit 6 — Unimplemented

Always reads 0

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 PGM = 1; then the write to ELAT is disabled.

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

EXCOL — Select Extra Columns Bit

0 = User array selected
1 = User array is disabled and extra columns are accessed at bits [7:0]. Addresses use bits [13:5] and bits [4:0] are don't care. EXCOL can be read and written only in special modes and always returns 0 in normal modes.

EXROW — Select Extra Rows Bit

0 = User array selected
1 = User array is disabled and two extra rows are available. Addresses use bits [7:0] and bits [13:8] are don't care. EXROW can be read and written only in special modes and always returns 0 in normal modes.

T[1:0] — EPROM Test Mode Select Bits

These bits allow selection of either gate stress or drain stress test modes. They can be read and written only in special modes and always read 0 in normal modes.

T1	T0	Function Selected
0	0	Normal mode
0	1	Reserved
1	0	Gate stress
1	1	Drain stress

PGM — EPROM Programming Voltage Enable Bit

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

0 = Programming voltage to EPROM array disconnected

1 = Programming voltage to EPROM array connected

2.5 EEPROM

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

2.5.1 EEPROM and CONFIG Programming and Erasure

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

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

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

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

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

2.5.1.1 Block Protect Register

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

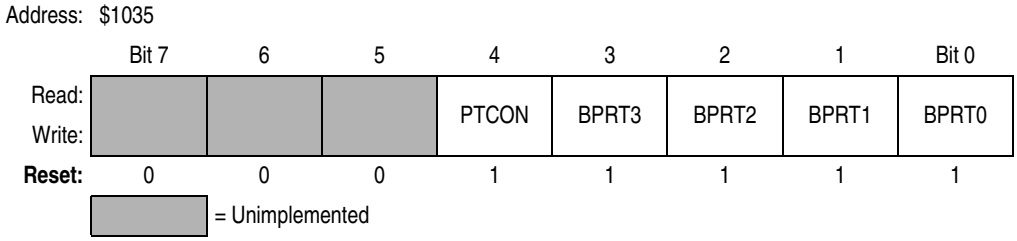


Figure 2-16. Block Protect Register (BPROT)

Bits [7:5] — Unimplemented

Always read 0

PTCON — Protect CONFIG Register Bit

0 = CONFIG register can be programmed or erased normally.
1 = CONFIG register cannot be programmed or erased.

BPRT[3:0] — Block Protect Bits for EEPROM

When set, these bits protect a block of EEPROM from being programmed or electronically erased. Ultraviolet light, however, can erase the entire EEPROM contents regardless of BPRT[3:0] (windowed packages only). Refer to Table 2-6 and Table 2-7.

When cleared, BPRT[3:0] allow programming and erasure of the associated block.

Table 2-6. EEPROM Block Protect

Bit Name	Block Protected	Block Size
BPRT0	$B600–$B61F	32 bytes
BPRT1	$B620–$B65F	64 bytes
BPRT2	$B660–$B6DF	128 bytes
BPRT3	$B6E0–$B7FF	288 bytes

Table 2-7. EEPROM Block Protect in MC68HC811E2 MCUs

Bit Name	Block Protected	Block Size
BPRT0	$x800–$x9FF(1)	512 bytes
BPRT1	$xA00–$xBFF(1)	512 bytes
BPRT2	$xC00–$xDFF(1)	512 bytes
BPRT3	$xE00–$xFFF(1)	512 bytes

1. x is determined by the value of EE[3:0] in CONFIG register. Refer to Figure 2-13.

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

EPGM — EPROM/OTPROM/EEPROM Programming Voltage Enable Bit

0 = Programming voltage to EEPROM array switched off
1 = Programming voltage to EEPROM array switched on

During EEPROM programming, the ROW and BYTE bits of PPROG are not used. If the frequency of the E clock is 1 MHz or less, set the CSEL bit in the OPTION register. Recall that 0s must be erased by a separate erase operation before programming. The following examples of how to program an EEPROM byte assume that the appropriate bits in BPROT are cleared.

```
PROG    LDAB    #$02    EELAT = 1
        STAB    $103B   Set EELAT bit
        STAA    $XXXX   Store data to EEPROM address
                        (for valid EEPROM address see memory
                        map for each device)
        LDAB    #$03    EELAT = 1, EPGM = 1
        STAB    $103B   Turn on programming voltage
        JSR     DLY10   Delay 10 ms
        CLR     $103B   Turn off high voltage and set
                        to READ mode
```

2.5.1.3 EEPROM Bulk Erase

This is an example of how to bulk erase the entire EEPROM. The CONFIG register is not affected in this example.

```
BULKE   LDAB    #$06    EELAT = 1, ERASE = 1
        STAB    $103B   Set to BULK erase mode
        STAA    $XXXX   Store data to any EEPROM address (for
                        valid EEPROM address see memory map
                        for each device)
        LDAB    #$07    EELAT = 1, EPGM = 1, ERASE = 1
        STAB    $103B   Turn on high voltage
        JSR     DLY10   Delay 10 ms
        CLR     $103B   Turn off high voltage and set
                        to READ mode
```

2.5.1.4 EEPROM Row Erase

This example shows how to perform a fast erase of large sections of EEPROM.

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

2.5.1.5 EEPROM Byte Erase

This is an example of how to erase a single byte of EEPROM.

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

2.5.1.6 CONFIG Register Programming

Because the CONFIG register is implemented with EEPROM cells, use EEPROM procedures to erase and program this register. The procedure for programming is the same as for programming a byte in the EEPROM array, except that the CONFIG register address is used. CONFIG can be programmed or erased (including byte erase) while the MCU is operating in any mode, provided that PTCON in BPROT is clear.

To change the value in the CONFIG register, complete this procedure.

1. Erase the CONFIG register.
2. Program the new value to the CONFIG address.
3. Initiate reset.

NOTE

Do not initiate a reset until the procedure is complete.

2.5.2 EEPROM Security

The optional security feature, available only on ROM-based MCUs, protects the EEPROM and RAM contents from unauthorized access. A program, or a key portion of a program, can be protected against unauthorized duplication. To accomplish this, the protection mechanism restricts operation of protected devices to the single-chip modes. This prevents the memory locations from being monitored externally because single-chip modes do not allow visibility of the internal address and data buses. Resident programs, however, have unlimited access to the internal EEPROM and RAM and can read, write, or transfer the contents of these memories.

An enhanced security feature which protects EPROM contents, RAM, and EEPROM from unauthorized accesses is available in MC68S711E9. Refer to Chapter 11 Ordering Information and Mechanical Specifications for the exact part number.

For further information, these engineering bulletins have been included at the back of this data book:

- EB183 — Enabling the Security Feature on the MC68HC711E9 Devices with PCbug11 on the M68HC711E9PGMR
- EB188 — Enabling the Security Feature on M68HC811E2 Devices with PCbug11 on the M68HC711E9PGMR

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.

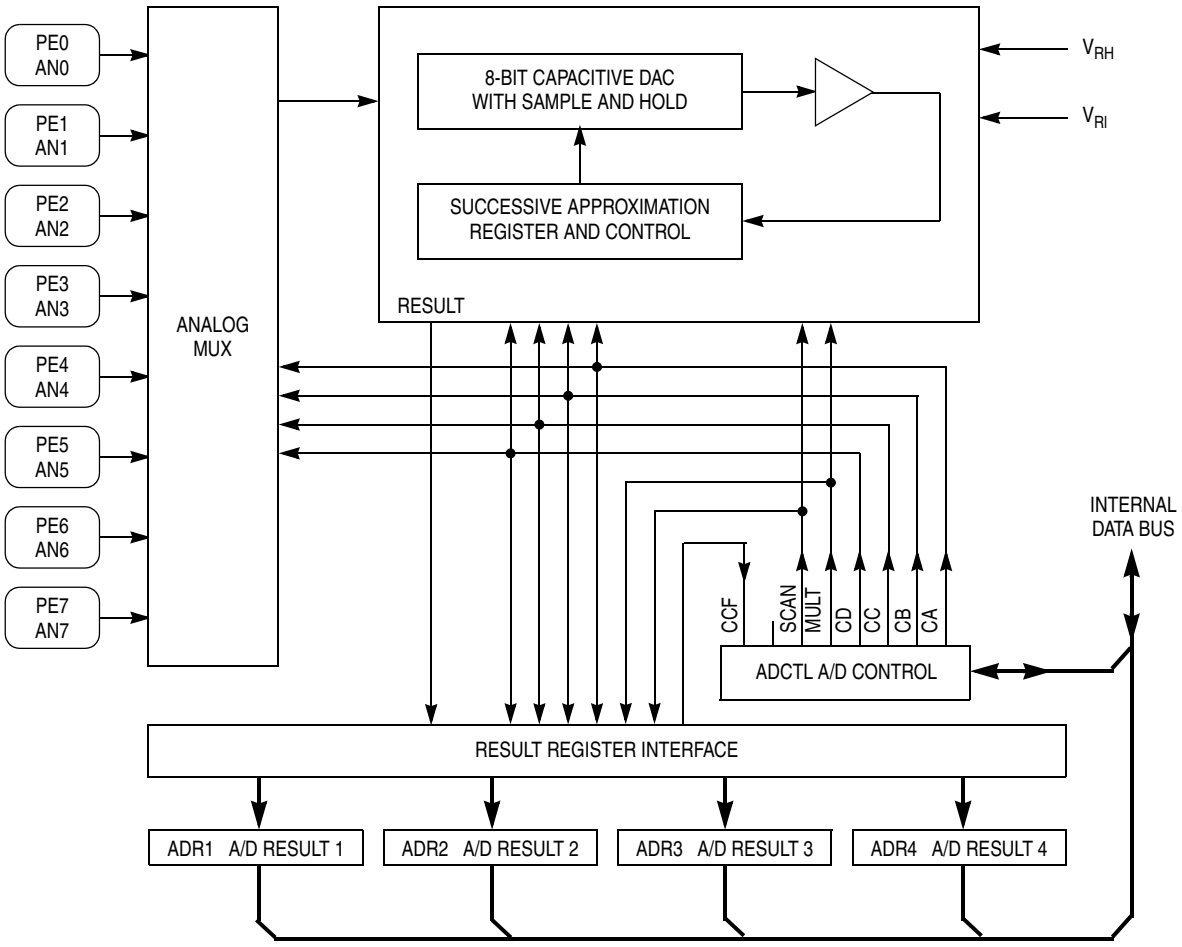
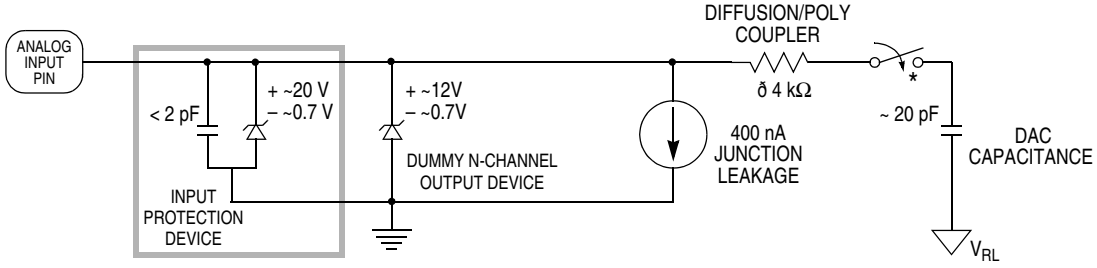


Figure 3-1. A/D Converter Block Diagram



* THIS ANALOG SWITCH IS CLOSED ONLY DURING THE 12-CYCLE SAMPLE TIME.

Figure 3-2. Electrical Model of an A/D Input Pin (Sample Mode)

3.2.3 Digital Control

All A/D converter operations are controlled by bits in register ADCTL. In addition to selecting the analog input to be converted, ADCTL bits indicate conversion status and control whether single or continuous conversions are performed. Finally, the ADCTL bits determine whether conversions are performed on single or multiple channels.

3.2.4 Result Registers

Four 8-bit registers ADR[4:1] store conversion results. Each of these registers can be accessed by the processor in the CPU. The conversion complete flag (CCF) indicates when valid data is present in the result registers. The result registers are written during a portion of the system clock cycle when reads do not occur, so there is no conflict.

3.2.5 A/D Converter Clocks

The CSEL bit in the OPTION register selects whether the A/D converter uses the system E clock or an internal RC oscillator for synchronization. When E-clock frequency is below 750 kHz, charge leakage in the capacitor array can cause errors, and the internal oscillator should be used. When the RC clock is used, additional errors can occur because the comparator is sensitive to the additional system clock noise.

3.2.6 Conversion Sequence

A/D converter operations are performed in sequences of four conversions each. A conversion sequence can repeat continuously or stop after one iteration. The conversion complete flag (CCF) is set after the fourth conversion in a sequence to show the availability of data in the result registers. Figure 3-3 shows the timing of a typical sequence. Synchronization is referenced to the system E clock.

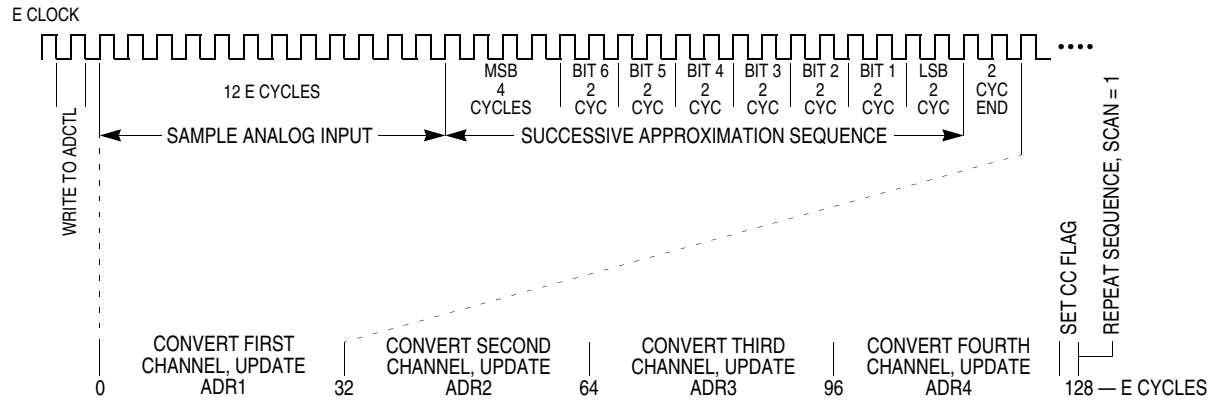


Figure 3-3. A/D Conversion Sequence

3.3 A/D Converter Power-Up and Clock Select

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

Address: $1039

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ADPU	CSEL	IRQE(1)	DLY(1)	CME		CR1(1)	CR0(1)
Reset:	0	0	0	1	0	0	0	0

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

= Unimplemented

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

ADPU — A/D Power-Up Bit

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

CSEL — Clock Select Bit

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

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

Refer to Chapter 5 Resets and Interrupts.

DLY — Enable Oscillator Startup Delay Bit

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

CME — Clock Monitor Enable Bit

Refer to Chapter 5 Resets and Interrupts.

Bit 2 — Not implemented

Always reads 0

CR[1:0] — COP Timer Rate Select Bits

Refer to Chapter 5 Resets and Interrupts and Chapter 9 Timing Systems.

3.4 Conversion Process

The A/D conversion sequence begins one E-clock cycle after a write to the A/D control/status register, ADCTL. The bits in ADCTL select the channel and the mode of conversion.

An input voltage equal to V_{RL} converts to $00 and an input voltage equal to V_{RH} converts to $FF (full scale), with no overflow indication. For ratiometric conversions of this type, the source of each analog input should use V_{RH} as the supply voltage and be referenced to V_{RL}.

3.5 Channel Assignments

The multiplexer allows the A/D converter to select one of 16 analog signals. Eight of these channels correspond to port E input lines to the MCU, four of the channels are internal reference points or test functions, and four channels are reserved. Refer to Table 3-1.

Table 3-1. Converter Channel Assignments

Channel Number	Channel Signal	Result in ADRx if MULT = 1
1	AN0	ADR1
2	AN1	ADR2
3	AN2	ADR3
4	AN3	ADR4
5	AN4	ADR1
6	AN5	ADR2
7	AN6	ADR3
8	AN7	ADR4
9 – 12	Reserved	—
13	V_{RH}(1)	ADR1
14	V_{RL}(1)	ADR2
15	$(V_{RH})/2$(1)	ADR3
16	Reserved(1)	ADR4

1. Used for factory testing

3.6 Single-Channel Operation

The two types of single-channel operation are:

1. When SCAN = 0, the single selected channel is converted four consecutive times. The first result is stored in A/D result register 1 (ADR1), and the fourth result is stored in ADR4. After the fourth conversion is complete, all conversion activity is halted until a new conversion command is written to the ADCTL register.
2. When SCAN = 1, conversions continue to be performed on the selected channel with the fifth conversion being stored in register ADR1 (overwriting the first conversion result), the sixth conversion overwriting ADR2, and so on.

3.7 Multiple-Channel Operation

The two types of multiple-channel operation are:

1. When SCAN = 0, a selected group of four channels is converted one time each. The first result is stored in A/D result register 1 (ADR1), and the fourth result is stored in ADR4. After the fourth conversion is complete, all conversion activity is halted until a new conversion command is written to the ADCTL register.
2. When SCAN = 1, conversions continue to be performed on the selected group of channels with the fifth conversion being stored in register ADR1 (replacing the earlier conversion result for the first channel in the group), the sixth conversion overwriting ADR2, and so on.

3.8 Operation in Stop and Wait Modes

If a conversion sequence is in progress when either the stop or wait mode is entered, the conversion of the current channel is suspended. When the MCU resumes normal operation, that channel is resampled and the conversion sequence is resumed. As the MCU exits wait mode, the A/D circuits are stable and valid results can be obtained on the first conversion. However, in stop mode, all analog bias currents are disabled and it is necessary to allow a stabilization period when leaving stop mode. If stop mode is exited with a delay (DLY = 1), there is enough time for these circuits to stabilize before the first conversion. If stop mode is exited with no delay (DLY bit in OPTION register = 0), allow 10 ms for the A/D circuitry to stabilize to avoid invalid results.

3.9 A/D Control/Status Register

All bits in this register can be read or written, except bit 7, which is a read-only status indicator, and bit 6, which always reads as 0. Write to ADCTL to initiate a conversion. To quit a conversion in progress, write to this register and a new conversion sequence begins immediately.

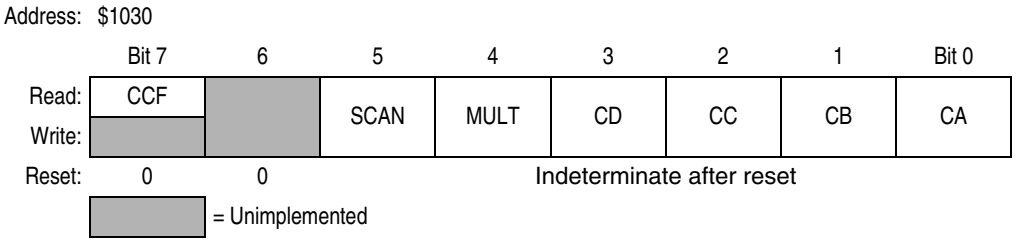


Figure 3-5. A/D Control/Status Register (ADCTL)

CCF — Conversion Complete Flag

A read-only status indicator, this bit is set when all four A/D result registers contain valid conversion results. Each time the ADCTL register is overwritten, this bit is automatically cleared to 0 and a conversion sequence is started. In the continuous mode, CCF is set at the end of the first conversion sequence.

Bit 6 — Unimplemented

Always reads 0

SCAN — Continuous Scan Control Bit

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

MULT — Multiple Channel/Single Channel Control Bit

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

NOTE

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

CD:CA — Channel Selects D:A Bits

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

Table 3-2. A/D Converter Channel Selection

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

1. Used for factory testing

3.10 A/D Converter Result Registers

These read-only registers hold an 8-bit conversion result. Writes to these registers have no effect. Data in the A/D converter result registers is valid when the CCF flag in the ADCTL register is set, indicating a conversion sequence is complete. If conversion results are needed sooner, refer to Figure 3-3, which shows the A/D conversion sequence diagram.

Register name: Analog-to-Digital Converter Result Register 1 Address: $1031

	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:	Indeterminate after reset							

Register name: Analog-to-Digital Converter Result Register 2 Address: $1032

	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:	Indeterminate after reset							

Register name: Analog-to-Digital Converter Result Register 3 Address: $1033

	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:	Indeterminate after reset							

Register name: Analog-to-Digital Converter Result Register 4 Address: $1034

	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:	Indeterminate after reset							

= Unimplemented

Figure 3-6. Analog-to-Digital Converter Result Registers (ADR1–ADR4)

Chapter 4
Central Processor Unit (CPU)

4.1 Introduction

Features of the M68HC11 Family include:

- Central processor unit (CPU) architecture
- Data types
- Addressing modes
- Instruction set
- Special operations such as subroutine calls and interrupts

The CPU is designed to treat all peripheral, input/output (I/O), and memory locations identically as addresses in the 64-Kbyte memory map. This is referred to as memory-mapped I/O. There are no special instructions for I/O that are separate from those used for memory. This architecture also allows accessing an operand from an external memory location with no execution time penalty.

4.2 CPU Registers

M68HC11 CPU registers are an integral part of the CPU and are not addressed as if they were memory locations. The seven registers, discussed in the following paragraphs, are shown in Figure 4-1.

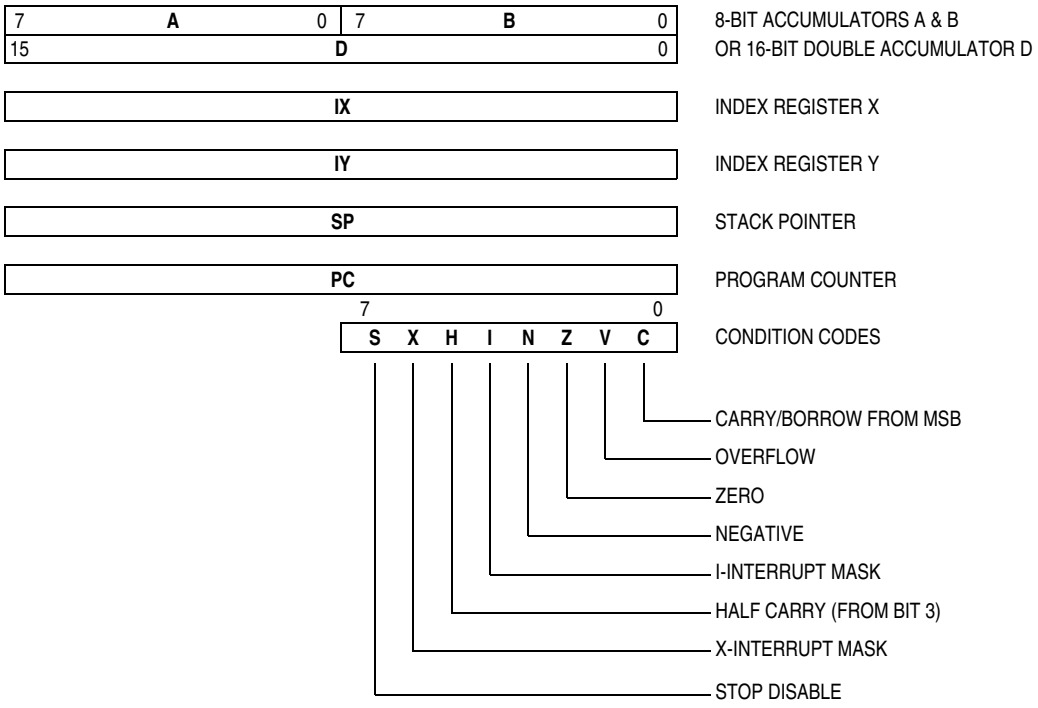


Figure 4-1. Programming Model

4.2.1 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, these 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.
- The add, subtract, and compare instructions associated with both A and B (ABA, SBA, and CBA) only operate in one direction, making it important to plan ahead to ensure that the correct operand is in the correct accumulator.

4.2.2 Index Register X (IX)

The IX register provides a 16-bit indexing value that can be added to the 8-bit offset provided in an instruction to create an effective address. The IX register can also be used as a counter or as a temporary storage register.

4.2.3 Index Register Y (IY)

The 16-bit IY register performs an indexed mode function similar to that of the IX register. However, most instructions using the IY register require an extra byte of machine code and an extra cycle of execution time because of the way the opcode map is implemented. Refer to 4.4 Opcodes and Operands for further information.

4.2.4 Stack Pointer (SP)

The M68HC11 CPU has an automatic program stack. This stack can be located anywhere in the address space and can be any size up to the amount of memory available in the system. Normally, the SP is initialized by one of the first instructions in an application program. The stack is configured as a data structure that grows downward from high memory to low memory. Each time a new byte is pushed onto the stack, the SP is decremented. Each time a byte is pulled from the stack, the SP is incremented. At any given time, the SP holds the 16-bit address of the next free location in the stack. Figure 4-2 is a summary of SP operations.

When a subroutine is called by a jump-to-subroutine (JSR) or branch-to- subroutine (BSR) instruction, the address of the instruction after the JSR or BSR is automatically pushed onto the stack, least significant byte first. When the subroutine is finished, a return-from-subroutine (RTS) instruction is executed. The RTS pulls the previously stacked return address from the stack and loads it into the program counter. Execution then continues at this recovered return address.

When an interrupt is recognized, the current instruction finishes normally, the return address (the current value in the program counter) is pushed onto the stack, all of the CPU registers are pushed onto the stack, and execution continues at the address specified by the vector for the interrupt.

At the end of the interrupt service routine, an return-from interrupt (RTI) instruction is executed. The RTI instruction causes the saved registers to be pulled off the stack in reverse order. Program execution resumes at the return address.

Certain instructions push and pull the A and B accumulators and the X and Y index registers and are often used to preserve program context. For example, pushing accumulator A onto the stack when entering a subroutine that uses accumulator A and then pulling accumulator A off the stack just before leaving the subroutine ensures that the contents of a register will be the same after returning from the subroutine as it was before starting the subroutine.

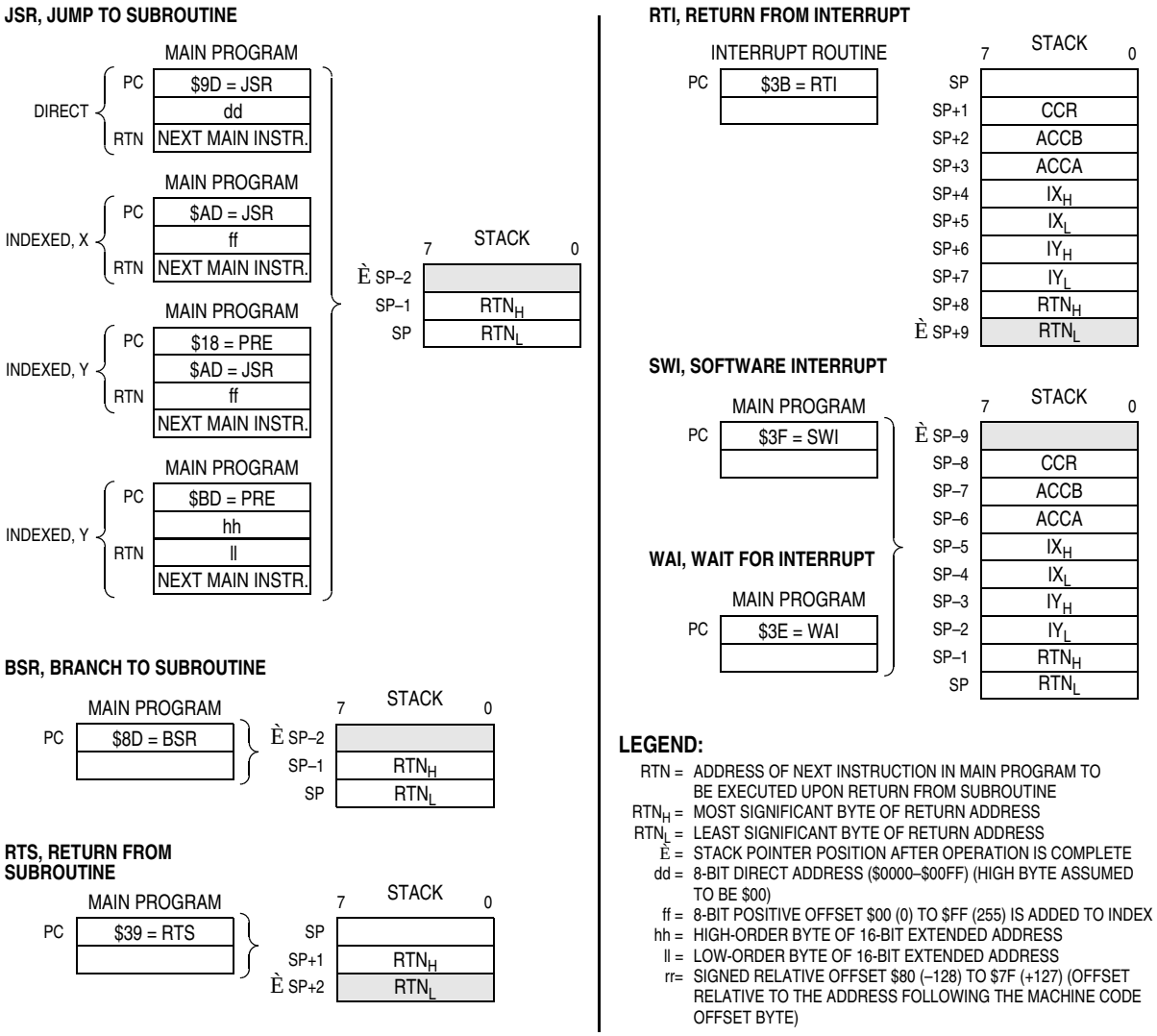


Figure 4-2. Stacking Operations

4.2.5 Program Counter (PC)

The program counter, a 16-bit register, contains the address of the next instruction to be executed. After reset, the program counter is initialized from one of six possible vectors, depending on operating mode and the cause of reset. See Table 4-1.

Table 4-1. Reset Vector Comparison

Mode	POR or $\overline{\text{RESET}}$ Pin	Clock Monitor	COP Watchdog
Normal	$FFFE, F	$FFFC, D	$FFFA, B
Test or Boot	$BFFE, F	$BFFC, D	$BFFA, B

4.2.6 Condition Code Register (CCR)

This 8-bit register contains:

- Five condition code indicators (C, V, Z, N, and H),
- Two interrupt masking bits ($\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$)
- A stop disable bit (S)

In the M68HC11 CPU, condition codes are updated automatically by most instructions. For example, load accumulator A (LDAA) and store accumulator A (STAA) instructions automatically set or clear the N, Z, and V condition code flags. Pushes, pulls, add B to X (ABX), add B to Y (ABY), and transfer/exchange instructions do not affect the condition codes. Refer to Table 4-2, which shows what condition codes are affected by a particular instruction.

4.2.6.1 Carry/Borrow (C)

The C bit is set if the arithmetic logic unit (ALU) performs a carry or borrow during an arithmetic operation. The C bit also acts as an error flag for multiply and divide operations. Shift and rotate instructions operate with and through the carry bit to facilitate multiple-word shift operations.

4.2.6.2 Overflow (V)

The overflow bit is set if an operation causes an arithmetic overflow. Otherwise, the V bit is cleared.

4.2.6.3 Zero (Z)

The Z bit is set if the result of an arithmetic, logic, or data manipulation operation is 0. Otherwise, the Z bit is cleared. Compare instructions do an internal implied subtraction and the condition codes, including Z, reflect the results of that subtraction. A few operations (INX, DEX, INY, and DEY) affect the Z bit and no other condition flags. For these operations, only = and ≠ conditions can be determined.

4.2.6.4 Negative (N)

The N bit is set if the result of an arithmetic, logic, or data manipulation operation is negative (MSB = 1). Otherwise, the N bit is cleared. A result is said to be negative if its most significant bit (MSB) is a 1. A quick way to test whether the contents of a memory location has the MSB set is to load it into an accumulator and then check the status of the N bit.

4.2.6.5 Interrupt Mask (I)

The interrupt request (IRQ) mask (I bit) is a global mask that disables all maskable interrupt sources. While the I bit is set, interrupts can become pending, but the operation of the CPU continues uninterrupted until the I bit is cleared. After any reset, the I bit is set by default and can only be cleared by a software instruction. When an interrupt is recognized, the I bit is set after the registers are stacked, but before the interrupt vector is fetched. After the interrupt has been serviced, a return-from-interrupt instruction is normally executed, restoring the registers to the values that were present before the interrupt occurred. Normally, the I bit is 0 after a return from interrupt is executed. Although the I bit can be cleared within an interrupt service routine, "nesting" interrupts in this way should only be done when there is a clear understanding of latency and of the arbitration mechanism. Refer to Chapter 5 Resets and Interrupts.

4.2.6.6 Half Carry (H)

The H bit is set when a carry occurs between bits 3 and 4 of the arithmetic logic unit during an ADD, ABA, or ADC instruction. Otherwise, the H bit is cleared. Half carry is used during BCD operations.

4.2.6.7 X Interrupt Mask (X)

The XIRQ mask (X) bit disables interrupts from the $\overline{\text{XIRQ}}$ pin. After any reset, X is set by default and must be cleared by a software instruction. When an $\overline{\text{XIRQ}}$ interrupt is recognized, the X and I bits are set after the registers are stacked, but before the interrupt vector is fetched. After the interrupt has been serviced, an RTI instruction is normally executed, causing the registers to be restored to the values that were present before the interrupt occurred. The X interrupt mask bit is set only by hardware ($\overline{\text{RESET}}$ or $\overline{\text{XIRQ}}$ acknowledge). X is cleared only by program instruction (TAP, where the associated bit of A is 0; or RTI, where bit 6 of the value loaded into the CCR from the stack has been cleared). There is no hardware action for clearing X.

4.2.6.8 STOP Disable (S)

Setting the STOP disable (S) bit prevents the STOP instruction from putting the M68HC11 into a low-power stop condition. If the STOP instruction is encountered by the CPU while the S bit is set, it is treated as a no-operation (NOP) instruction, and processing continues to the next instruction. S is set by reset; STOP is disabled by default.

4.3 Data Types

The M68HC11 CPU supports four data types:

1. Bit data
2. 8-bit and 16-bit signed and unsigned integers
3. 16-bit unsigned fractions
4. 16-bit addresses

A byte is eight bits wide and can be accessed at any byte location. A word is composed of two consecutive bytes with the most significant byte at the lower value address. Because the M68HC11 is an 8-bit CPU, there are no special requirements for alignment of instructions or operands.

4.4 Opcodes and Operands

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

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

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

4.5 Addressing Modes

Six addressing modes can be used to access memory:

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

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

4.5.1 Immediate

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

4.5.2 Direct

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

4.5.3 Extended

In the extended addressing mode, the effective address of the argument is contained in two bytes following the opcode byte. These are 3-byte instructions (or 4-byte instructions if a prebyte is required). One or two bytes are needed for the opcode and two for the effective address.

4.5.4 Indexed

In the indexed addressing mode, an 8-bit unsigned offset contained in the instruction is added to the value contained in an index register (IX or IY). The sum is the effective address. This addressing mode allows referencing any memory location in the 64-Kbyte address space. These are 2- to 5-byte instructions, depending on whether or not a prebyte is required.

4.5.5 Inherent

In the inherent addressing mode, all the information necessary to execute the instruction is contained in the opcode. Operations that use only the index registers or accumulators, as well as control instructions with no arguments, are included in this addressing mode. These are
1- or 2-byte instructions.

4.5.6 Relative

The relative addressing mode is used only for branch instructions. If the branch condition is true, an 8-bit signed offset included in the instruction is added to the contents of the program counter to form the effective branch address. Otherwise, control proceeds to the next instruction. These are usually 2-byte instructions.

4.6 Instruction Set

Refer to Table 4-2, which shows all the M68HC11 instructions in all possible addressing modes. For each instruction, the table shows the operand construction, the number of machine code bytes, and execution time in CPU E-clock cycles.

Table 4-2. Instruction Set (Sheet 1 of 7)

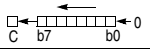
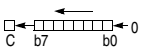
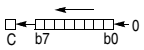
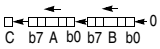
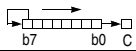
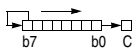
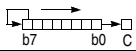
Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
ABA	Add Accumulators	A + B ⇒ A	INH	1B	—	2	—	—	Δ	—	Δ	Δ	Δ	Δ
ABX	Add B to X	IX + (00 : B) ⇒ IX	INH	3A	—	3	—	—	—	—	—	—	—	—
ABY	Add B to Y	IY + (00 : B) ⇒ IY	INH	18 3A	—	4	—	—	—	—	—	—	—	—
ADCA (opr)	Add with Carry to A	A + M + C ⇒ A	A IMM	89	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			A DIR	99	dd	3								
			A EXT	B9	hh ll	4								
			A IND,X	A9	ff	4								
			A IND,Y	18 A9	ff	5								
ADCB (opr)	Add with Carry to B	B + M + C ⇒ B	B IMM	C9	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			B DIR	D9	dd	3								
			B EXT	F9	hh ll	4								
			B IND,X	E9	ff	4								
			B IND,Y	18 E9	ff	5								
ADDA (opr)	Add Memory to A	A + M ⇒ A	A IMM	8B	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			A DIR	9B	dd	3								
			A EXT	BB	hh ll	4								
			A IND,X	AB	ff	4								
			A IND,Y	18 AB	ff	5								
ADDB (opr)	Add Memory to B	B + M ⇒ B	B IMM	CB	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			B DIR	DB	dd	3								
			B EXT	FB	hh ll	4								
			B IND,X	EB	ff	4								
			B IND,Y	18 EB	ff	5								
ADDD (opr)	Add 16-Bit to D	D + (M : M + 1) ⇒ D	IMM	C3	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ
			DIR	D3	dd	5								
			EXT	F3	hh ll	6								
			IND,X	E3	ff	6								
			IND,Y	18 E3	ff	7								
ANDA (opr)	AND A with Memory	A • M ⇒ A	A IMM	84	ii	2	—	—	—	—	Δ	Δ	0	—
			A DIR	94	dd	3								
			A EXT	B4	hh ll	4								
			A IND,X	A4	ff	4								
			A IND,Y	18 A4	ff	5								
ANDB (opr)	AND B with Memory	B • M ⇒ B	B IMM	C4	ii	2	—	—	—	—	Δ	Δ	0	—
			B DIR	D4	dd	3								
			B EXT	F4	hh ll	4								
			B IND,X	E4	ff	4								
			B IND,Y	18 E4	ff	5								
ASL (opr)	Arithmetic Shift Left		EXT	78	hh ll	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND,X	68	ff	6								
			IND,Y	18 68	ff	7								
ASLA	Arithmetic Shift Left A		A INH	48	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLB	Arithmetic Shift Left B		B INH	58	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLD	Arithmetic Shift Left D		INH	05	—	3	—	—	—	—	Δ	Δ	Δ	Δ
ASR	Arithmetic Shift Right		EXT	77	hh ll	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND,X	67	ff	6								
			IND,Y	18 67	ff	7								
ASRA	Arithmetic Shift Right A		A INH	47	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRB	Arithmetic Shift Right B		B INH	57	—	2	—	—	—	—	Δ	Δ	Δ	Δ
BCC (rel)	Branch if Carry Clear	? C = 0	REL	24	rr	3	—	—	—	—	—	—	—	—
BCLR (opr) (msk)	Clear Bit(s)	M • ($\overline{mm}$) ⇒ M	DIR	15	dd mm	6	—	—	—	—	Δ	Δ	0	—
			IND,X	1D	ff mm	7								
			IND,Y	18 1D	ff mm	8								
BCS (rel)	Branch if Carry Set	? C = 1	REL	25	rr	3	—	—	—	—	—	—	—	—
BEQ (rel)	Branch if = Zero	? Z = 1	REL	27	rr	3	—	—	—	—	—	—	—	—
BGE (rel)	Branch if Δ Zero	? N ⊕ V = 0	REL	2C	rr	3	—	—	—	—	—	—	—	—

Table 4-2. Instruction Set (Sheet 2 of 7)

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	? ($\overline{M}$) • 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

Table 4-2. Instruction Set (Sheet 3 of 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
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	2 3 4 4 5	—	—	—	—	Δ	Δ	Δ	Δ
COM (opr)	Ones Complement Memory Byte	$FF – M ⇒ M	EXT IND,X IND,Y	73 63 18 63	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	0	1
COMA	Ones Complement A	$FF – A ⇒ A	A INH	43	—	2	—	—	—	—	Δ	Δ	0	1
COMB	Ones Complement B	$FF – B ⇒ B	B INH	53	—	2	—	—	—	—	Δ	Δ	0	1
CPD (opr)	Compare D to Memory 16-Bit	D – M : M + 1	IMM DIR EXT IND,X IND,Y	1A 83 1A 93 1A B3 1A A3 CD A3	jj kk dd hh ll ff ff	5 6 7 7 7	—	—	—	—	Δ	Δ	Δ	Δ
CPX (opr)	Compare X to Memory 16-Bit	IX – M : M + 1	IMM DIR EXT IND,X IND,Y	8C 9C BC AC CD AC	jj kk dd hh ll ff ff	4 5 6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
CPY (opr)	Compare Y to Memory 16-Bit	IY – M : M + 1	IMM DIR EXT IND,X IND,Y	18 8C 18 9C 18 BC 1A AC 18 AC	jj kk dd hh ll ff ff	5 6 7 7 7	—	—	—	—	Δ	Δ	Δ	Δ
DAA	Decimal Adjust A	Adjust Sum to BCD	INH	19	—	2	—	—	—	—	Δ	Δ	Δ	Δ
DEC (opr)	Decrement Memory Byte	M – 1 ⇒ M	EXT IND,X IND,Y	7A 6A 18 6A	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	—
DECA	Decrement Accumulator A	A – 1 ⇒ A	A INH	4A	—	2	—	—	—	—	Δ	Δ	Δ	—
DECB	Decrement Accumulator B	B – 1 ⇒ B	B INH	5A	—	2	—	—	—	—	Δ	Δ	Δ	—
DES	Decrement Stack Pointer	SP – 1 ⇒ SP	INH	34	—	3	—	—	—	—	—	—	—	—
DEX	Decrement Index Register X	IX – 1 ⇒ IX	INH	09	—	3	—	—	—	—	—	Δ	—	—
DEY	Decrement Index Register Y	IY – 1 ⇒ IY	INH	18 09	—	4	—	—	—	—	—	Δ	—	—
EORA (opr)	Exclusive OR A with Memory	A ⊕ M ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	88 98 B8 A8 18 A8	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
EORB (opr)	Exclusive OR B with Memory	B ⊕ M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	C8 D8 F8 E8 18 E8	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
FDIV	Fractional Divide 16 by 16	D / IX ⇒ IX; r ⇒ D	INH	03	—	41	—	—	—	—	—	Δ	Δ	Δ
IDIV	Integer Divide 16 by 16	D / IX ⇒ IX; r ⇒ D	INH	02	—	41	—	—	—	—	—	Δ	0	Δ
INC (opr)	Increment Memory Byte	M + 1 ⇒ M	EXT IND,X IND,Y	7C 6C 18 6C	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	—
INCA	Increment Accumulator A	A + 1 ⇒ A	A INH	4C	—	2	—	—	—	—	Δ	Δ	Δ	—

Table 4-2. Instruction Set (Sheet 4 of 7)

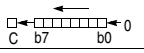
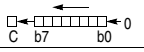
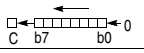
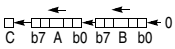
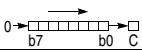
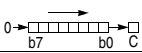
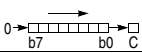
Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
INCB	Increment Accumulator B	B + 1 ⇒ B	B INH	5C	—	2	—	—	—	—	Δ	Δ	Δ	—
INS	Increment Stack Pointer	SP + 1 ⇒ SP	INH	31	—	3	—	—	—	—	—	—	—	—
INX	Increment Index Register X	IX + 1 ⇒ IX	INH	08	—	3	—	—	—	—	—	Δ	—	—
INY	Increment Index Register Y	IY + 1 ⇒ IY	INH	18 08	—	4	—	—	—	—	—	Δ	—	—
JMP (opr)	Jump	See Figure 3–2	EXT IND,X IND,Y	7E 6E 18 6E	hh ll ff ff	3 3 4	—	—	—	—	—	—	—	—
JSR (opr)	Jump to Subroutine	See Figure 3–2	DIR EXT IND,X IND,Y	9D BD AD 18 AD	dd hh ll ff ff	5 6 6 7	—	—	—	—	—	—	—	—
LDAA (opr)	Load Accumulator A	M ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	86 96 B6 A6 18 A6	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
LDAB (opr)	Load Accumulator B	M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	C6 D6 F6 E6 18 E6	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
LDD (opr)	Load Double Accumulator D	M ⇒ A,M + 1 ⇒ B	IMM DIR EXT IND,X IND,Y	CC DC FC EC 18 EC	jj kk dd hh ll ff ff	3 4 5 5 6	—	—	—	—	Δ	Δ	0	—
LDS (opr)	Load Stack Pointer	M : M + 1 ⇒ SP	IMM DIR EXT IND,X IND,Y	8E 9E BE AE 18 AE	jj kk dd hh ll ff ff	3 4 5 5 6	—	—	—	—	Δ	Δ	0	—
LDX (opr)	Load Index Register X	M : M + 1 ⇒ IX	IMM DIR EXT IND,X IND,Y	CE DE FE EE CD EE	jj kk dd hh ll ff ff	3 4 5 5 6	—	—	—	—	Δ	Δ	0	—
LDY (opr)	Load Index Register Y	M : M + 1 ⇒ IY	IMM DIR EXT IND,X IND,Y	18 CE 18 DE 18 FE 1A EE 18 EE	jj kk dd hh ll ff ff	4 5 6 6 6	—	—	—	—	Δ	Δ	0	—
LSL (opr)	Logical Shift Left		EXT IND,X IND,Y	78 68 18 68	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
LSLA	Logical Shift Left A		A INH	48	—	2	—	—	—	—	Δ	Δ	Δ	Δ
LSLB	Logical Shift Left B		B INH	58	—	2	—	—	—	—	Δ	Δ	Δ	Δ
LSLD	Logical Shift Left Double		INH	05	—	3	—	—	—	—	Δ	Δ	Δ	Δ
LSR (opr)	Logical Shift Right		EXT IND,X IND,Y	74 64 18 64	hh ll ff ff	6 6 7	—	—	—	—	0	Δ	Δ	Δ
LSRA	Logical Shift Right A		A INH	44	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRB	Logical Shift Right B		B INH	54	—	2	—	—	—	—	0	Δ	Δ	Δ

Table 4-2. Instruction Set (Sheet 5 of 7)

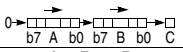
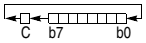
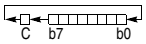
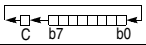
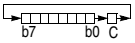
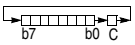
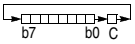
Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
LSRD	Logical Shift Right Double		INH	04	—	3	—	—	—	—	0	Δ	Δ	Δ
MUL	Multiply 8 by 8	A * B ⇒ D	INH	3D	—	10	—	—	—	—	—	—	—	Δ
NEG (opr)	Two's Complement Memory Byte	0 – M ⇒ M	EXT IND,X IND,Y	70 60 18 60	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
NEGA	Two's Complement A	0 – A ⇒ A	A INH	40	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGB	Two's Complement B	0 – B ⇒ B	B INH	50	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NOP	No operation	No Operation	INH	01	—	2	—	—	—	—	—	—	—	—
ORAA (opr)	OR Accumulator A (Inclusive)	A + M ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	8A 9A BA AA 18 AA	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
ORAB (opr)	OR Accumulator B (Inclusive)	B + M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	CA DA FA EA 18 EA	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
PSHA	Push A onto Stack	A ⇒ Stk,SP = SP – 1	A INH	36	—	3	—	—	—	—	—	—	—	—
PSHB	Push B onto Stack	B ⇒ Stk,SP = SP – 1	B INH	37	—	3	—	—	—	—	—	—	—	—
PSHX	Push X onto Stack (Lo First)	IX ⇒ Stk,SP = SP – 2	INH	3C	—	4	—	—	—	—	—	—	—	—
PSHY	Push Y onto Stack (Lo First)	IY ⇒ Stk,SP = SP – 2	INH	18 3C	—	5	—	—	—	—	—	—	—	—
PULA	Pull A from Stack	SP = SP + 1, A ⇐ Stk	A INH	32	—	4	—	—	—	—	—	—	—	—
PULB	Pull B from Stack	SP = SP + 1, B ⇐ Stk	B INH	33	—	4	—	—	—	—	—	—	—	—
PULX	Pull X From Stack (Hi First)	SP = SP + 2, IX ⇐ Stk	INH	38	—	5	—	—	—	—	—	—	—	—
PULY	Pull Y from Stack (Hi First)	SP = SP + 2, IY ⇐ Stk	INH	18 38	—	6	—	—	—	—	—	—	—	—
ROL (opr)	Rotate Left		EXT IND,X IND,Y	79 69 18 69	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
ROLA	Rotate Left A		A INH	49	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLB	Rotate Left B		B INH	59	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROR (opr)	Rotate Right		EXT IND,X IND,Y	76 66 18 66	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
RORA	Rotate Right A		A INH	46	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORB	Rotate Right B		B INH	56	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RTI	Return from Interrupt	See Figure 3–2	INH	3B	—	12	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ
RTS	Return from Subroutine	See Figure 3–2	INH	39	—	5	—	—	—	—	—	—	—	—
SBA	Subtract B from A	A – B ⇒ A	INH	10	—	2	—	—	—	—	Δ	Δ	Δ	Δ

Table 4-2. Instruction Set (Sheet 6 of 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	X	H	I	N	Z	V	C
SBCA (opr)	Subtract with Carry from A	A – M – C ⇒ A	A IMM	82	ii	2	—	—	—	—	Δ	Δ	Δ	Δ
			A DIR	92	dd	3								
			A EXT	B2	hh ll	4								
			A IND,X	A2	ff	4								
			A IND,Y	18 A2	ff	5								
SBCB (opr)	Subtract with Carry from B	B – M – C ⇒ B	B IMM	C2	ii	2	—	—	—	—	Δ	Δ	Δ	Δ
			B DIR	D2	dd	3								
			B EXT	F2	hh ll	4								
			B IND,X	E2	ff	4								
			B IND,Y	18 E2	ff	5								
SEC	Set Carry	1 ⇒ C	INH	0D	—	2	—	—	—	—	—	—	—	1
SEI	Set Interrupt Mask	1 ⇒ I	INH	0F	—	2	—	—	—	1	—	—	—	—
SEV	Set Overflow Flag	1 ⇒ V	INH	0B	—	2	—	—	—	—	—	—	1	—
STAA (opr)	Store Accumulator A	A ⇒ M	A DIR	97	dd	3	—	—	—	—	Δ	Δ	0	—
			A EXT	B7	hh ll	4								
			A IND,X	A7	ff	4								
			A IND,Y	18 A7	ff	5								
STAB (opr)	Store Accumulator B	B ⇒ M	B DIR	D7	dd	3	—	—	—	—	Δ	Δ	0	—
			B EXT	F7	hh ll	4								
			B IND,X	E7	ff	4								
			B IND,Y	18 E7	ff	5								
STD (opr)	Store Accumulator D	A ⇒ M, B ⇒ M + 1	DIR	DD	dd	4	—	—	—	—	Δ	Δ	0	—
			EXT	FD	hh ll	5								
			IND,X	ED	ff	5								
			IND,Y	18 ED	ff	6								
STOP	Stop Internal Clocks	—	INH	CF	—	2	—	—	—	—	—	—	—	—
STS (opr)	Store Stack Pointer	SP ⇒ M : M + 1	DIR	9F	dd	4	—	—	—	—	Δ	Δ	0	—
			EXT	BF	hh ll	5								
			IND,X	AF	ff	5								
			IND,Y	18 AF	ff	6								
STX (opr)	Store Index Register X	IX ⇒ M : M + 1	DIR	DF	dd	4	—	—	—	—	Δ	Δ	0	—
			EXT	FF	hh ll	5								
			IND,X	EF	ff	5								
			IND,Y	CD EF	ff	6								
STY (opr)	Store Index Register Y	IY ⇒ M : M + 1	DIR	18 DF	dd	5	—	—	—	—	Δ	Δ	0	—
			EXT	18 FF	hh ll	6								
			IND,X	1A EF	ff	6								
			IND,Y	18 EF	ff	6								
SUBA (opr)	Subtract Memory from A	A – M ⇒ A	A IMM	80	ii	2	—	—	—	—	Δ	Δ	Δ	Δ
			A DIR	90	dd	3								
			A EXT	B0	hh ll	4								
			A IND,X	A0	ff	4								
			A IND,Y	18 A0	ff	5								
SUBB (opr)	Subtract Memory from B	B – M ⇒ B	A IMM	C0	ii	2	—	—	—	—	Δ	Δ	Δ	Δ
			A DIR	D0	dd	3								
			A EXT	F0	hh ll	4								
			A IND,X	E0	ff	4								
			A IND,Y	18 E0	ff	5								
SUBD (opr)	Subtract Memory from D	D – M : M + 1 ⇒ D	IMM	83	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ
			DIR	93	dd	5								
			EXT	B3	hh ll	6								
			IND,X	A3	ff	6								
			IND,Y	18 A3	ff	7								
SWI	Software Interrupt	See Figure 3–2	INH	3F	—	14	—	—	—	1	—	—	—	—
TAB	Transfer A to B	A ⇒ B	INH	16	—	2	—	—	—	—	Δ	Δ	0	—
TAP	Transfer A to CC Register	A ⇒ CCR	INH	06	—	2	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ
TBA	Transfer B to A	B ⇒ A	INH	17	—	2	—	—	—	—	Δ	Δ	0	—
TEST	TEST (Only in Test Modes)	Address Bus Counts	INH	00	—	*	—	—	—	—	—	—	—	—
TPA	Transfer CC Register to A	CCR ⇒ A	INH	07	—	2	—	—	—	—	—	—	—	—
TST (opr)	Test for Zero or Minus	M – 0	EXT	7D	hh ll	6	—	—	—	—	Δ	Δ	0	0
			IND,X	6D	ff	6								
			IND,Y	18 6D	ff	7								

Table 4-2. Instruction Set (Sheet 7 of 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
TSTA	Test A for Zero or Minus	A – 0	A INH	4D	—	2	—	—	—	—	Δ	Δ	0	0
TSTB	Test B for Zero or Minus	B – 0	B INH	5D	—	2	—	—	—	—	Δ	Δ	0	0
TSX	Transfer Stack Pointer to X	SP + 1 ⇒ IX	INH	30	—	3	—	—	—	—	—	—	—	—
TSY	Transfer Stack Pointer to Y	SP + 1 ⇒ IY	INH	18 30	—	4	—	—	—	—	—	—	—	—
TXS	Transfer X to Stack Pointer	IX – 1 ⇒ SP	INH	35	—	3	—	—	—	—	—	—	—	—
TYS	Transfer Y to Stack Pointer	IY – 1 ⇒ SP	INH	18 35	—	4	—	—	—	—	—	—	—	—
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E	—	**	—	—	—	—	—	—	—	—
XGDX	Exchange D with X	IX ⇒ D, D ⇒ IX	INH	8F	—	3	—	—	—	—	—	—	—	—
XGDY	Exchange D with Y	IY ⇒ D, D ⇒ IY	INH	18 8F	—	4	—	—	—	—	—	—	—	—

Cycle

* Infinity or until reset occurs

** 12 cycles are used beginning with the opcode fetch. A wait state is entered which remains in effect for an integer number of MPU E-clock cycles (n) until an interrupt is recognized. Finally, two additional cycles are used to fetch the appropriate interrupt vector (14 + n total).

Operands

`dd` = 8-bit direct address ($0000–$00FF) (high byte assumed to be $00)
`ff` = 8-bit positive offset $00 (0) to $FF (255) (is added to index)
`hh` = High-order byte of 16-bit extended address
`ii` = One byte of immediate data
`jj` = High-order byte of 16-bit immediate data
`kk` = Low-order byte of 16-bit immediate data
`ll` = Low-order byte of 16-bit extended address
`mm` = 8-bit mask (set bits to be affected)
`rr` = Signed relative offset $80 (–128) to $7F (+127)
(offset relative to address following machine code offset byte))

Operators

( ) Contents of register shown inside parentheses
⇐ Is transferred to
⇑ Is pulled from stack
⇓ Is pushed onto stack
• Boolean AND
\+ Arithmetic addition symbol except where used as inclusive-OR symbol in Boolean formula
⊕ Exclusive-OR
* Multiply
: Concatenation
– Arithmetic subtraction symbol or negation symbol (two's complement)

Condition Codes

— Bit not changed
0 Bit always cleared
1 Bit always set
Δ Bit cleared or set, depending on operation
↓ Bit can be cleared, cannot become set

Chapter 5
Resets and Interrupts

5.1 Introduction

Resets and interrupt operations load the program counter with a vector that points to a new location from which instructions are to be fetched. A reset immediately stops execution of the current instruction and forces the program counter to a known starting address. Internal registers and control bits are initialized so the MCU can resume executing instructions. An interrupt temporarily suspends normal program execution while an interrupt service routine is being executed. After an interrupt has been serviced, the main program resumes as if there had been no interruption.

5.2 Resets

The four possible sources of reset are:

- Power-on reset (POR)
- External reset ($\overline{\text{RESET}}$)
- Computer operating properly (COP) reset
- Clock monitor reset

POR and $\overline{\text{RESET}}$ share the normal reset vector. COP reset and the clock monitor reset each has its own vector.

5.2.1 Power-On Reset (POR)

A positive transition on V_{DD} generates a power-on reset (POR), which is used only for power-up conditions. POR cannot be used to detect drops in power supply voltages. A 4064 t_{CYC} (internal clock cycle) delay after the oscillator becomes active allows the clock generator to stabilize. If $\overline{\text{RESET}}$ is at logical 0 at the end of 4064 t_{CYC}, the CPU remains in the reset condition until $\overline{\text{RESET}}$ goes to logical 1.

The POR circuit only initializes internal circuitry during cold starts. Refer to Figure 1-7. External Reset Circuit.

NOTE

It is important to protect the MCU during power transitions. Most M68HC11 systems need an external circuit that holds the $\overline{\text{RESET}}$ pin low whenever V_{DD} is below the minimum operating level. This external voltage level detector, or other external reset circuits, are the usual source of reset in a system.

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 E/2^{15} By	XTAL = 4.0 MHz Timeout – 0 ms, + 32.8 ms	XTAL = 8.0 MHz Timeout – 0 ms, + 16.4 ms	XTAL = 12.0 MHz Timeout – 0 ms, + 10.9 ms	XTAL = 16.0 MHz Timeout – 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

Address	$103A							
	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:	0	0	0	0	0	0	0	0

Figure 5-1. Arm/Reset COP Timer Circuitry Register (COPRST)

Complete this 2-step reset sequence to service the COP timer:

1. Write $55 to COPRST to arm the COP timer clearing mechanism.
2. Write $AA to COPRST to clear the COP timer.

Performing instructions between these two steps is possible as long as both steps are completed in the correct sequence before the timer times out.

5.2.4 Clock Monitor Reset

The clock monitor circuit is based on an internal resistor capacitor (RC) time delay. If no MCU clock edges are detected within this RC time delay, the clock monitor can optionally generate a system reset. The clock monitor function is enabled or disabled by the CME control bit in the OPTION register. The presence of a timeout is determined by the RC delay, which allows the clock monitor to operate without any MCU clocks.

Clock monitor is used as a backup for the COP system. Because the COP needs a clock to function, it is disabled when the clock stops. Therefore, the clock monitor system can detect clock failures not detected by the COP system.

Semiconductor wafer processing causes variations of the RC timeout values between individual devices. An E-clock frequency below 10 kHz is detected as a clock monitor error. An E-clock frequency of 200 kHz or more prevents clock monitor errors. Using the clock monitor function when the E-clock is below 200 kHz is not recommended.

Special considerations are needed when a STOP instruction is executed and the clock monitor is enabled. Because the STOP function causes the clocks to be halted, the clock monitor function generates a reset sequence if it is enabled at the time the stop mode was initiated. Before executing a STOP instruction, clear the CME bit in the OPTION register to 0 to disable the clock monitor. After recovery from STOP, set the CME bit to logic 1 to enable the clock monitor. Alternatively, executing a STOP instruction with the CME bit set to logic 1 can be used as a software initiated reset.

5.2.5 System Configuration Options Register

Address:	$1039							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ADPU	CSEL	IRQE(1)	DLY(1)	CME		CR1(1)	CR0(1)
Reset:	0	0	0	1	0	0	0	0

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

= Unimplemented

Figure 5-2. System Configuration Options Register (OPTION)

ADPU — Analog-to-Digital Converter Power-Up Bit

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

CSEL — Clock Select Bit

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

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

0 = $\overline{\text{IRQ}}$ is configured for level-sensitive operation.
1 = $\overline{\text{IRQ}}$ is configured for edge-sensitive-only operation.

DLY — Enable Oscillator Startup Delay Bit

Refer to Chapter 2 Operating Modes and On-Chip Memory and Chapter 3 Analog-to-Digital (A/D) Converter.

CME — Clock Monitor Enable Bit

This control bit can be read or written at any time and controls whether or not the internal clock monitor circuit triggers a reset sequence when the system clock is slow or absent. When it is clear, the clock monitor circuit is disabled, and when it is set, the clock monitor circuit is enabled. Reset clears the CME bit.

0 = Clock monitor circuit disabled
1 = Slow or stopped clocks cause reset

Bit 2 — Unimplemented

Always reads 0

CR[1:0] — COP Timer Rate Select Bit

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. See Table 5-1 for specific timeout settings.

5.2.6 Configuration Control Register

Address:	$103F							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EE3	EE2	EE1	EE0	NOSEC	NOCOP	ROMON	EEON
Reset:	0	0	0	0	1	1	1	1

Figure 5-3. Configuration Control Register (CONFIG)

EE[3:0] — EEPROM Mapping Bits

EE[3:0] apply only to MC68HC811E2. Refer to Chapter 2 Operating Modes and On-Chip Memory.

NOSEC — Security Mode Disable Bit

Refer to Chapter 2 Operating Modes and On-Chip Memory.

NOCOP — COP System Disable Bit

0 = COP enabled (forces reset on timeout)
1 = COP disabled (does not force reset on timeout)

ROMON — ROM (EPROM) Enable Bit

Refer to Chapter 2 Operating Modes and On-Chip Memory.

EEON — EEPROM Enable Bit

Refer to Chapter 2 Operating Modes and On-Chip Memory.

5.3 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, Reset Vector, and Operating Mode

Cause of Reset	Normal Mode Vector	Special Test or Bootstrap
POR or $\overline{\text{RESET}}$ pin	$FFFE, FFFF	$BFFE, $BFFF
Clock monitor failure	$FFFC, FFFD	$BFFC, $BFFD
COP Watchdog Timeout	$FFFA, FFFB	$BFFA, $BFFB

These initial states then control on-chip peripheral systems to force them to known startup states, as described in the following subsections.

5.3.1 Central Processor Unit (CPU)

After reset, the central processor unit (CPU) fetches the restart vector from the appropriate address during the first three cycles and begins executing instructions. The stack pointer and other CPU registers are indeterminate immediately after reset; however, the X and I interrupt mask bits in the condition code register (CCR) are set to mask any interrupt requests. Also, the S bit in the CCR is set to inhibit stop mode.

5.3.2 Memory Map

After reset, the INIT register is initialized to $01, mapping the RAM at $00 and the control registers at $1000.

For the MC68HC811E2, the CONFIG register resets to $FF. EEPROM mapping bits (EE[3:0]) place the EEPROM at $F800. Refer to the memory map diagram for MC68HC811E2 in Chapter 2 Operating Modes and On-Chip Memory.

5.3.3 Timer

During reset, the timer system is initialized to a count of $0000. The prescaler bits are cleared, and all output compare registers are initialized to $FFFF. All input capture registers are indeterminate after reset. The output compare 1 mask (OC1M) register is cleared so that successful OC1 compares do not affect any I/O pins. The other four output compares are configured so that they do not affect any I/O pins on successful compares. All input capture edge-detector circuits are configured for capture disabled operation. The timer overflow interrupt flag and all eight timer function interrupt flags are cleared. All nine timer interrupts are disabled because their mask bits have been cleared.

The I4/O5 bit in the PACTL register is cleared to configure the I4/O5 function as OC5; however, the OM5:OL5 control bits in the TCTL1 register are clear so OC5 does not control the PA3 pin.

5.3.4 Real-Time Interrupt (RTI)

The real-time interrupt flag (RTIF) is cleared and automatic hardware interrupts are masked. The rate control bits are cleared after reset and can be initialized by software before the real-time interrupt (RTI) system is used.

5.3.5 Pulse Accumulator

The pulse accumulator system is disabled at reset so that the pulse accumulator input (PAI) pin defaults to being a general-purpose input pin.

5.3.6 Computer Operating Properly (COP)

The COP watchdog system is enabled if the NOCOP control bit in the CONFIG register is cleared and disabled if NOCOP is set. The COP rate is set for the shortest duration timeout.

5.3.7 Serial Communications Interface (SCI)

The reset condition of the SCI system is independent of the operating mode. At reset, the SCI baud rate control register (BAUD) is initialized to $04. All transmit and receive interrupts are masked and both the transmitter and receiver are disabled so the port pins default to being general-purpose I/O lines. The SCI frame format is initialized to an 8-bit character size. The send break and receiver wakeup functions are disabled. The TDRE and TC status bits in the SCI status register (SCSR) are both 1s, indicating that there is no transmit data in either the transmit data register or the transmit serial shift register. The RDRF, IDLE, OR, NF, FE, PF, and RAF receive-related status bits in the SCI control register 2 (SCCR2) are cleared.

5.3.8 Serial Peripheral Interface (SPI)

The SPI system is disabled by reset. The port pins associated with this function default to being general-purpose I/O lines.

5.3.9 Analog-to-Digital (A/D) Converter

The analog-to-digital (A/D) converter configuration is indeterminate after reset. The ADPU bit is cleared by reset, which disables the A/D system. The conversion complete flag is indeterminate.

5.3.10 System

The EEPROM programming controls are disabled, so the memory system is configured for normal read operation. PSEL[3:0] are initialized with the value %0110, causing the external $\overline{\text{IRQ}}$ pin to have the highest I-bit interrupt priority. The $\overline{\text{IRQ}}$ pin is configured for level-sensitive operation (for wired-OR systems). The RBOOT, SMOD, and MDA bits in the HPRIO register reflect the status of the MODB and MODA inputs at the rising edge of reset. MODA and MODB inputs select one of the four operating modes. After reset, writing SMOD and MDA in special modes causes the MCU to change operating modes. Refer to the description of HPRIO register in Chapter 2 Operating Modes and On-Chip Memory for a detailed description of SMOD and MDA. The DLY control bit is set to specify that an oscillator startup delay is imposed upon recovery from stop mode. The clock monitor system is disabled because CME is cleared.

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.

5.4.1 Highest Priority Interrupt and Miscellaneous Register

Address: $103C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RBOOT(1)	SMOD(1)	MDA(1)	IRVNE	PSEL2	PSEL2	PSEL1	PSEL0
Reset:								
Single chip:	0	0	0	0	0	1	1	0
Expanded:	0	0	1	0	0	1	1	0
Bootstrap:	1	1	0	0	0	1	1	0
Special test:	0	1	1	1	0	1	1	0

1. The values of the RBOOT, SMOD, and MDA reset bits depend on the mode selected at the RESET pin rising edge. Refer to Table 2-1. Hardware Mode Select Summary.

Figure 5-4. Highest Priority I-Bit Interrupt and Miscellaneous Register (HPRIO)

RBOOT — Read Bootstrap ROM Bit

Has meaning only when the SMOD bit is a 1 (bootstrap mode or special test mode). At all other times this bit is clear and cannot be written. Refer to Chapter 2 Operating Modes and On-Chip Memory for more information.

SMOD — Special Mode Select Bit

This bit reflects the inverse of the MODB input pin at the rising edge of reset. Refer to Chapter 2 Operating Modes and On-Chip Memory for more information.

MDA — Mode Select A Bit

The mode select A bit reflects the status of the MODA input pin at the rising edge of reset. Refer to Chapter 2 Operating Modes and On-Chip Memory for more information.

IRVNE — Internal Read Visibility/Not E Bit

The IRVNE control bit allows internal read accesses to be available on the external data bus during operation in expanded modes. In single-chip and bootstrap modes, IRVNE determines whether the E clock is driven out an external pin. For the MC68HC811E2, this bit is IRV and only controls internal read visibility. Refer to Chapter 2 Operating Modes and On-Chip Memory for more information.

PSEL[3:0] — Priority Select Bits

These bits select one interrupt source to be elevated above all other I-bit-related sources and can be written only while the I bit in the CCR is set (interrupts disabled).

Table 5-3. Highest Priority Interrupt Selection

PSEL[3:0]	Interrupt Source Promoted
0 0 0 0	Timer overflow
0 0 0 1	Pulse accumulator overflow
0 0 1 0	Pulse accumulator input edge
0 0 1 1	SPI serial transfer complete
0 1 0 0	SCI serial system
0 1 0 1	Reserved (default to $\overline{\text{IRQ}}$)
0 1 1 0	$\overline{\text{IRQ}}$ (external pin or parallel I/O)
0 1 1 1	Real-time interrupt
1 0 0 0	Timer input capture 1
1 0 0 1	Timer input capture 2
1 0 1 0	Timer input capture 3
1 0 1 1	Timer output compare 1
1 1 0 0	Timer output compare 2
1 1 0 1	Timer output compare 3
1 1 1 0	Timer output compare 4
1 1 1 1	Timer input capture 4/output compare 5

5.5 Interrupts

The MCU has 18 interrupt vectors that support 22 interrupt sources. The 15 maskable interrupts are generated by on-chip peripheral systems. These interrupts are recognized when the global interrupt mask bit (I) in the condition code register (CCR) is clear. The three non-maskable interrupt sources are illegal opcode trap, software interrupt, and $\overline{\text{XIRQ}}$ pin. Refer to Table 5-4, which shows the interrupt sources and vector assignments for each source.

For some interrupt sources, such as the SCI interrupts, the flags are automatically cleared during the normal course of responding to the interrupt requests. For example, the RDRF flag in the SCI system is cleared by the automatic clearing mechanism consisting of a read of the SCI status register while RDRF is set, followed by a read of the SCI data register. The normal response to an RDRF interrupt request would be to read the SCI status register to check for receive errors, then to read the received data from the SCI data register. These steps satisfy the automatic clearing mechanism without requiring special instructions.

Table 5-4. Interrupt and Reset Vector Assignments

Vector Address	Interrupt Source	CCR Mask Bit	Local Mask
FFC0, C1 – FFD4, D5	Reserved	—	—
FFD6, D7	SCI serial system • SCI receive data register full • SCI receiver overrun • SCI transmit data register empty • SCI transmit complete • SCI idle line detect	I	RIE RIE TIE TCIE ILIE
FFD8, D9	SPI serial transfer complete	I	SPIE
FFDA, DB	Pulse accumulator input edge	I	PAII
FFDC, DD	Pulse accumulator overflow	I	PAOVI
FFDE, DF	Timer overflow	I	TOI
FFE0, E1	Timer input capture 4/output compare 5	I	I4/O5I
FFE2, E3	Timer output compare 4	I	OC4I
FFE4, E5	Timer output compare 3	I	OC3I
FFE6, E7	Timer output compare 2	I	OC2I
FFE8, E9	Timer output compare 1	I	OC1I
FFEA, EB	Timer input capture 3	I	IC3I
FFEC, ED	Timer input capture 2	I	IC2I
FFEE, EF	Timer input capture 1	I	IC1I
FFF0, F1	Real-time interrupt	I	RTII
FFF2, F3	$\overline{\text{IRQ}}$ (external pin)	I	None
FFF4, F5	$\overline{\text{XIRQ}}$ pin	X	None
FFF6, F7	Software interrupt	None	None
FFF8, F9	Illegal opcode trap	None	None
FFFA, FB	COP failure	None	NOCOP
FFFC, FD	Clock monitor fail	None	CME
FFFE, FF	$\overline{\text{RESET}}$	None	None

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 Chapter 4 Central Processor Unit (CPU).

Table 5-5. Stacking Order on Entry to Interrupts

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

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

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

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

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

5.5.3 Illegal Opcode Trap

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

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

5.5.4 Software Interrupt (SWI)

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.

5.5.5 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.

5.5.6 Reset and Interrupt Processing

Figure 5-5 and Figure 5-6 illustrate the reset and interrupt process. Figure 5-5 illustrates how the CPU begins from a reset and how interrupt detection relates to normal opcode fetches. Figure 5-6 is an expansion of a block in Figure 5-5 and illustrates interrupt priorities. Figure 5-7 shows the resolution of interrupt sources within the SCI subsystem.

5.6 Low-Power Operation

Both stop mode and wait mode suspend CPU operation until a reset or interrupt occurs. Wait mode suspends processing and reduces power consumption to an intermediate level. Stop mode turns off all on-chip clocks and reduces power consumption to an absolute minimum while retaining the contents of the entire RAM array.

5.6.1 Wait Mode

The WAI opcode places the MCU in wait mode, during which the CPU registers are stacked and CPU processing is suspended until a qualified interrupt is detected. The interrupt can be an external $\overline{\text{IRQ}}$, an $\overline{\text{XIRQ}}$, or any of the internally generated interrupts, such as the timer or serial interrupts. The on-chip crystal oscillator remains active throughout the wait standby period.

The reduction of power in the wait condition depends on how many internal clock signals driving on-chip peripheral functions can be shut down. The CPU is always shut down during wait. While in the wait state, the address/data bus repeatedly runs read cycles to the address where the CCR contents were stacked. The MCU leaves the wait state when it senses any interrupt that has not been masked.

The free-running timer system is shut down only if the I bit is set to 1 and the COP system is disabled by NOCOP being set to 1. Several other systems also can be in a reduced power-consumption state depending on the state of software-controlled configuration control bits. Power consumption by the analog-to-digital (A/D) converter is not affected significantly by the wait condition. However, the A/D converter current can be eliminated by writing the ADPU bit to 0. The SPI system is enabled or disabled by the SPE control bit. The SCI transmitter is enabled or disabled by the TE bit, and the SCI receiver is enabled or disabled by the RE bit. Therefore, the power consumption in wait is dependent on the particular application.

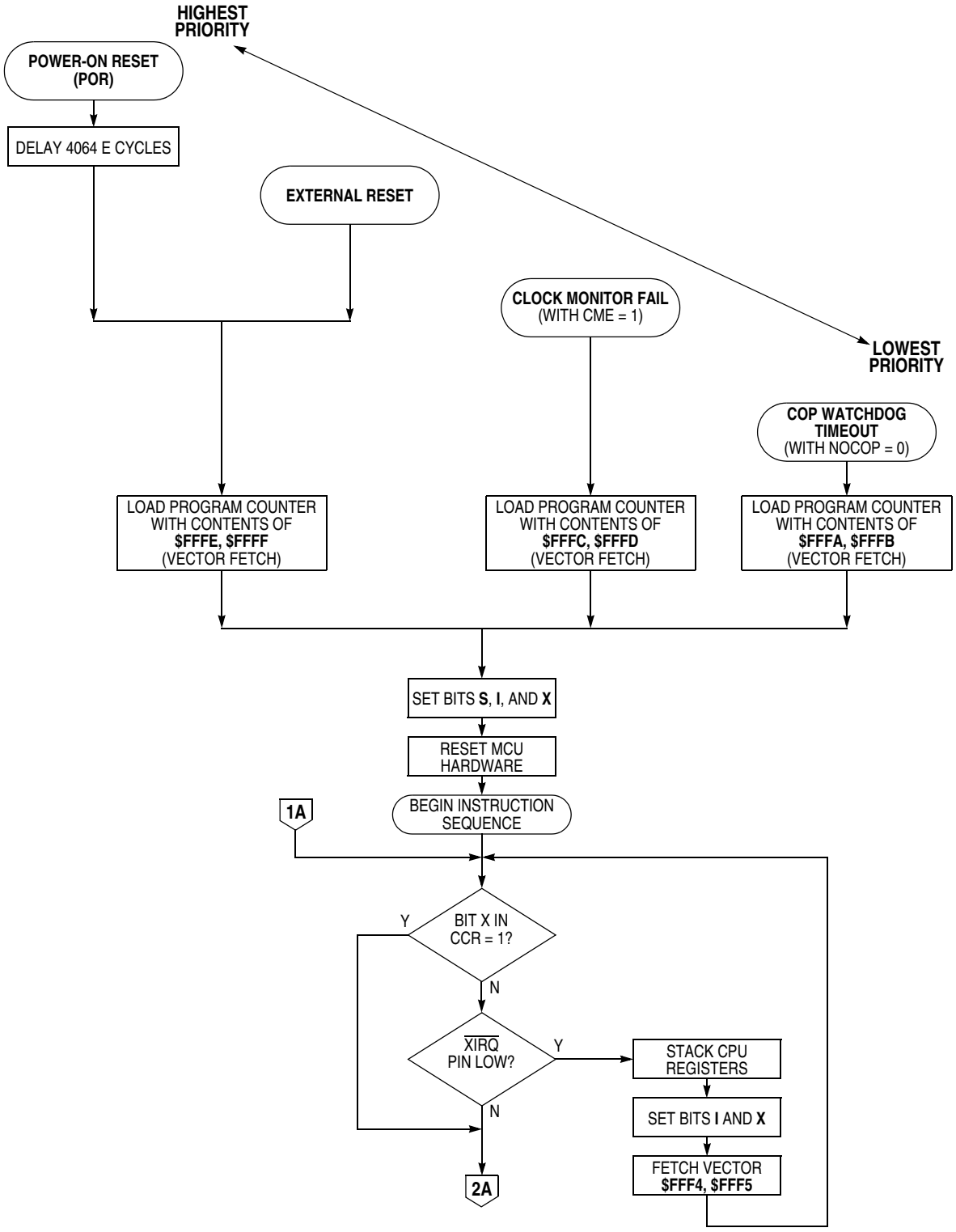


Figure 5-5. Processing Flow Out of Reset (Sheet 1 of 2)

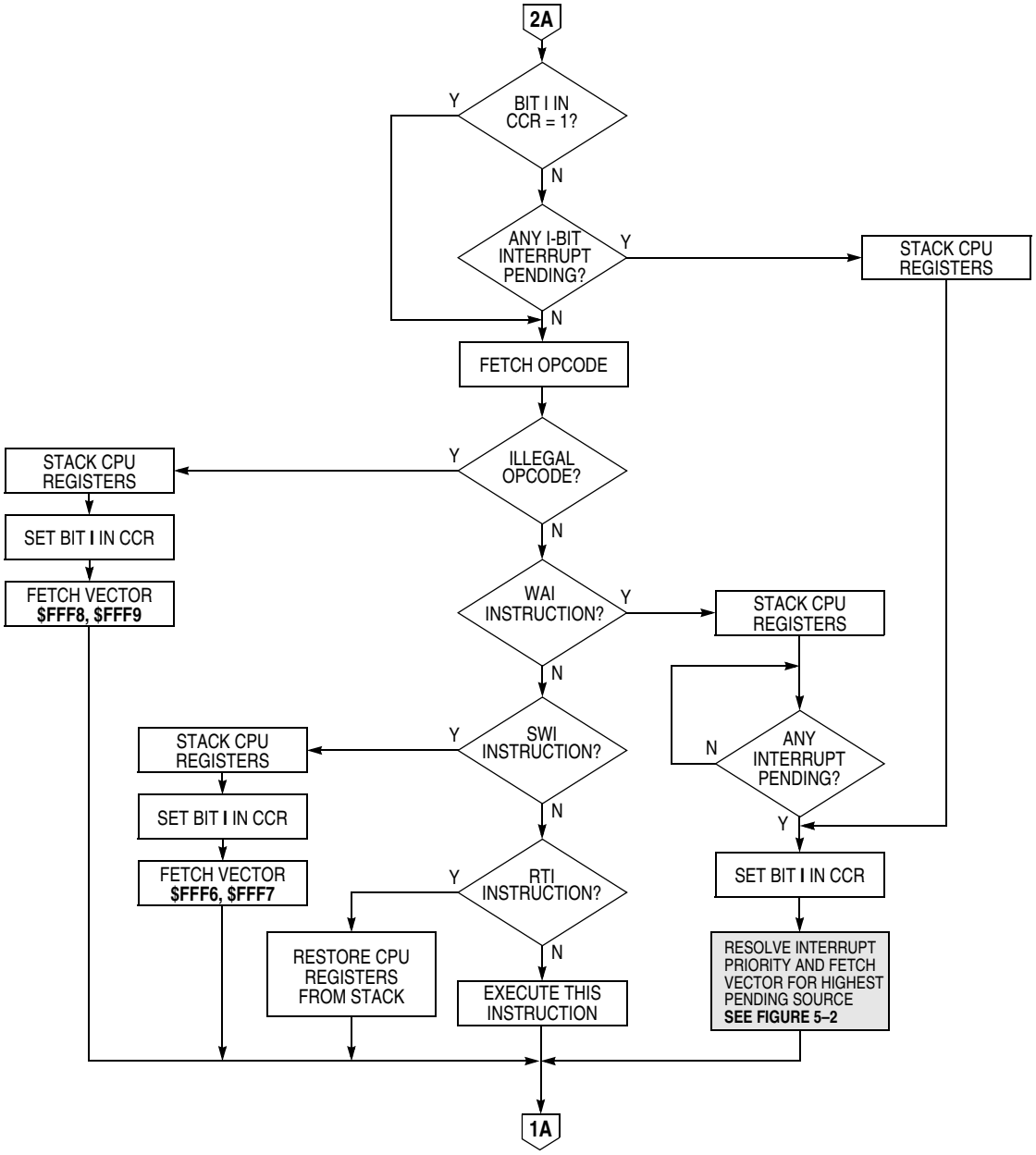


Figure 5-5. Processing Flow Out of Reset (Sheet 2 of 2)

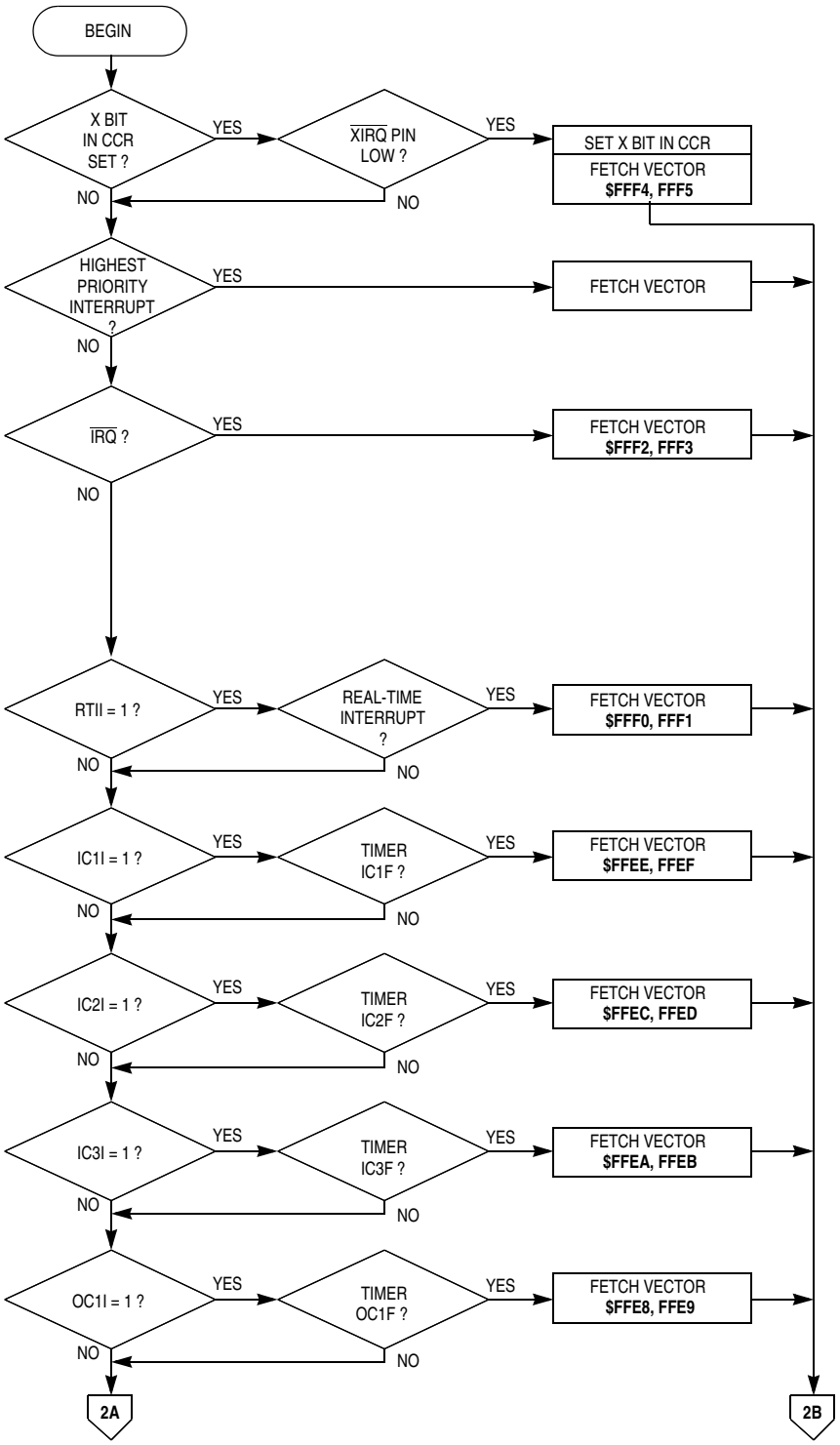


Figure 5-6. Interrupt Priority Resolution (Sheet 1 of 2)

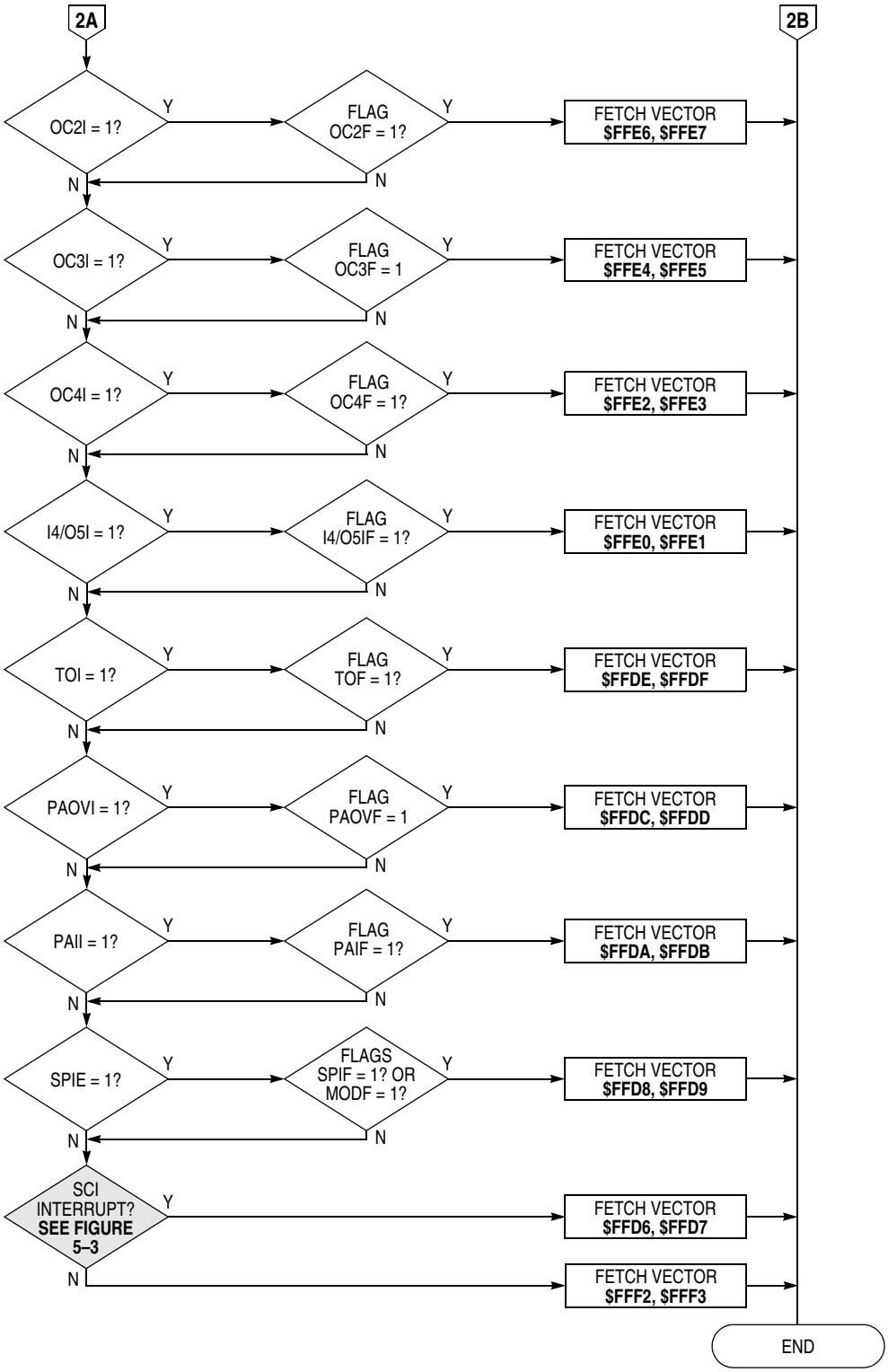


Figure 5-6. Interrupt Priority Resolution (Sheet 2 of 2)

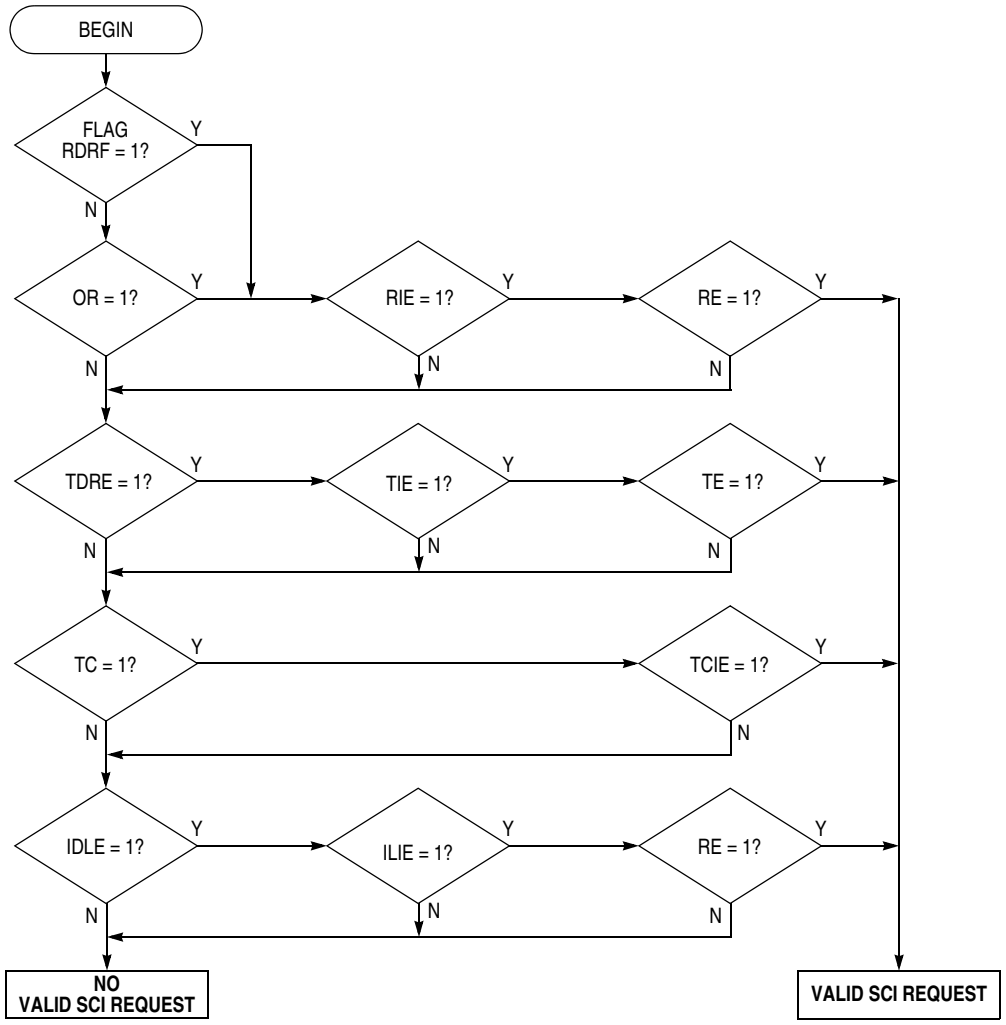


Figure 5-7. Interrupt Source Resolution Within SCI

5.6.2 Stop Mode

Executing the STOP instruction while the S bit in the CCR is equal to 0 places the MCU in stop mode. If the S bit is not 0, the stop opcode is treated as a no-op (NOP). Stop mode offers minimum power consumption because all clocks, including the crystal oscillator, are stopped while in this mode. To exit stop and resume normal processing, a logic low level must be applied to one of the external interrupts ($\overline{\text{IRQ}}$ or $\overline{\text{XIRQ}}$) or to the $\overline{\text{RESET}}$ pin. A pending edge-triggered $\overline{\text{IRQ}}$ can also bring the CPU out of stop.

Because all clocks are stopped in this mode, all internal peripheral functions also stop. The data in the internal RAM is retained as long as V_{DD} power is maintained. The CPU state and I/O pin levels are static and are unchanged by stop. Therefore, when an interrupt comes to restart the system, the MCU resumes processing as if there were no interruption. If reset is used to restart the system, a normal reset sequence results in which all I/O pins and functions are also restored to their initial states.

To use the $\overline{\text{IRQ}}$ pin as a means of recovering from stop, the I bit in the CCR must be clear ($\overline{\text{IRQ}}$ not masked). The $\overline{\text{XIRQ}}$ pin can be used to wake up the MCU from stop regardless of the state of the X bit in the CCR, although the recovery sequence depends on the state of the X bit. If X is set to 0 ($\overline{\text{XIRQ}}$ not

masked), the MCU starts up, beginning with the stacking sequence leading to normal service of the $\overline{\text{XIRQ}}$ request. If X is set to 1 ($\overline{\text{XIRQ}}$ masked or inhibited), then processing continues with the instruction that immediately follows the STOP instruction, and no $\overline{\text{XIRQ}}$ interrupt service is requested or pending.

Because the oscillator is stopped in stop mode, a restart delay may be imposed to allow oscillator stabilization upon leaving stop. If the internal oscillator is being used, this delay is required; however, if a stable external oscillator is being used, the DLY control bit can be used to bypass this startup delay. The DLY control bit is set by reset and can be optionally cleared during initialization. If the DLY equal to 0 option is used to avoid startup delay on recovery from stop, then reset should not be used as the means of recovering from stop, as this causes DLY to be set again by reset, imposing the restart delay. This same delay also applies to power-on reset, regardless of the state of the DLY control bit, but does not apply to a reset while the clocks are running.

Chapter 6
Parallel Input/Output (I/O) Ports

6.1 Introduction

All M68HC11 E-series MCUs have five input/output (I/O) ports and up to 38 I/O lines, depending on the operating mode. Refer to Table 6-1 for a summary of the ports and their shared functions.

Table 6-1. Input/Output Ports

Port	Input Pins	Output Pins	Bidirectional Pins	Shared Functions
Port A	3	3	2	Timer
Port B	—	8	—	High-order address
Port C	—	—	8	Low-order address and data bus
Port D	—	—	6	Serial communications interface (SCI) and serial peripheral interface (SPI)
Port E	8	—	—	Analog-to-digital (A/D) converter

Port pin function is mode dependent. Do not confuse pin function with the electrical state of the pin at reset. Port pins are either driven to a specified logic level or are configured as high-impedance inputs. I/O pins configured as high-impedance inputs have port data that is indeterminate.

In port descriptions, an I indicates this condition. Port pins that are driven to a known logic level during reset are shown with a value of either 1 or 0. Some control bits are unaffected by reset. Reset states for these bits are indicated with a U.

6.2 Port A

Port A shares functions with the timer system and has:

- Three input-only pins
- Three output-only pins
- Two bidirectional I/O pins

Address:	$1000							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
Reset:	I	0	0	0	I	I	I	I
Alternate function:	PAI	OC2	OC3	OC4	IC4/OC5	IC1	IC2	IC3
And/or:	OC1	OC1	OC1	OC1	OC1	—	—	—

I = Indeterminate after reset

Figure 6-1. Port A Data Register (PORTA)

Address:	$1026							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRA7	PAEWN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0
Reset:	0	0	0	0	0	0	0	0

Figure 6-2. Pulse Accumulator Control Register (PACTL)

DDRA7 — Data Direction for Port A Bit 7

Overridden if an output compare function is configured to control the PA7 pin

0 = Input
1 = Output

The pulse accumulator uses port A bit 7 as the PAI input, but the pin can also be used as general-purpose I/O or as an output compare.

NOTE

Even when port A bit 7 is configured as an output, the pin still drives the input to the pulse accumulator.

PAEN — Pulse Accumulator System Enable Bit

Refer to Chapter 9 Timing Systems.

PAMOD — Pulse Accumulator Mode Bit

Refer to Chapter 9 Timing Systems.

PEDGE — Pulse Accumulator Edge Control Bit

Refer to Chapter 9 Timing Systems.

DDRA3 — Data Direction for Port A Bit 3

This bit is overridden if an output compare function is configured to control the PA3 pin.

0 = Input
1 = Output

I4/O5 — Input Capture 4/Output Compare 5 Bit

Refer to Chapter 9 Timing Systems.

RTR[1:0] — RTI Interrupt Rate Select Bits

Refer to Chapter 9 Timing Systems.

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)

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

Address: $1007

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

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

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

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

0 = Input
1 = Output

6.5 Port D

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

Address: $1008

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

I = Indeterminate after reset

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

Address: $1009

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

= Unimplemented

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

Bits [7:6] — Unimplemented

Always read 0

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

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

0 = Input
1 = Output

6.6 Port E

Port E is used for general-purpose static inputs or pins that share functions with the analog-to-digital (A/D) converter system. When some port E pins are being used for general-purpose input and others are being used as A/D inputs, PORTE should not be read during the sample portion of an A/D conversion.

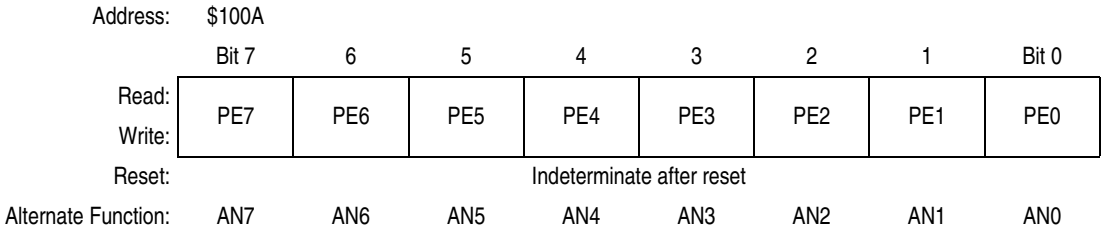


Figure 6-9. Port E Data Register (PORTE)

6.7 Handshake Protocol

Simple and full handshake input and output functions are available on ports B and C pins in single-chip mode. In simple strobed mode, port B is a strobed output port and port C is a latching input port. The two activities are available simultaneously.

The STRB output is pulsed for two E-clock periods each time there is a write to the PORTB register. The INVB bit in the PIOC register controls the polarity of STRB pulses. Port C levels are latched into the alternate port C latch (PORTCL) register on each assertion of the STRA input. STRA edge select, flag, and interrupt enable bits are located in the PIOC register. Any or all of the port C lines can still be used as general-purpose I/O while in strobed input mode.

Full handshake modes use port C pins and the STRA and STRB lines. Input and output handshake modes are supported, and output handshake mode has a 3-stated variation. STRA is an edge-detecting input and STRB is a handshake output. Control and enable bits are located in the PIOC register.

In full input handshake mode, the MCU asserts STRB to signal an external system that it is ready to latch data. Port C logic levels are latched into PORTCL when the STRA line is asserted by the external system. The MCU then negates STRB. The MCU reasserts STRB after the PORTCL register is read. In this mode, a mix of latched inputs, static inputs, and static outputs is allowed on port C, differentiated by the data direction bits and use of the PORTC and PORTCL registers.

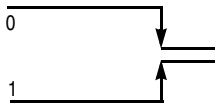
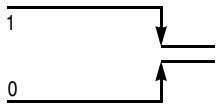
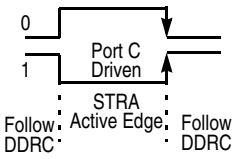
In full output handshake mode, the MCU writes data to PORTCL which, in turn, asserts the STRB output to indicate that data is ready. The external system reads port C data and asserts the STRA input to acknowledge that data has been received.

In the 3-state variation of output handshake mode, lines intended as 3-state handshake outputs are configured as inputs by clearing the corresponding DDRC bits. The MCU writes data to PORTCL and asserts STRB. The external system responds by activating the STRA input, which forces the MCU to drive the data in PORTC out on all of the port C lines. After the trailing edge of the active signal on STRA, the MCU negates the STRB signal. The 3-state mode variation does not allow part of port C to be used for static inputs while other port C pins are being used for handshake outputs. Refer to the 6.8 Parallel I/O Control Register for further information.

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

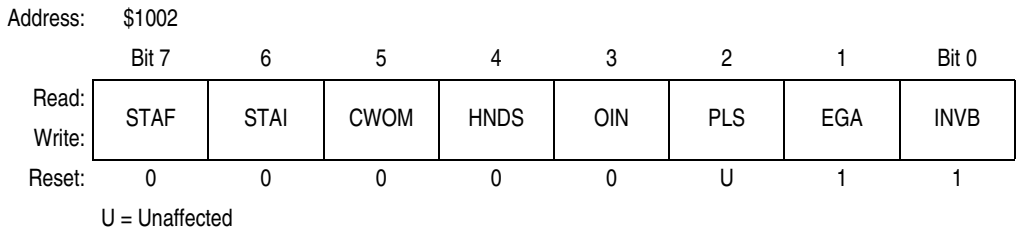


Figure 6-10. Parallel I/O Control Register (PIOC)

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

CWOM — Port C Wired-OR Mode Bit (affects all eight port C pins)

It is customary to have an external pullup resistor on lines that are driven by open-drain devices.

0 = Port C outputs are normal CMOS outputs.
1 = Port C outputs are open-drain outputs.

HNDS — Handshake Mode Bit

0 = Simple strobe mode
1 = Full input or output handshake mode

OIN — Output or Input Handshake Select Bit

HNDS must be set to 1 for this bit to have meaning.

0 = Input handshake
1 = Output handshake

PLS — Pulsed/Interlocked Handshake Operation Bit

HNDS must be set to 1 for this bit to have meaning. When interlocked handshake is selected, strobe B is active until the selected edge of strobe A is detected.

0 = Interlocked handshake
1 = Pulsed handshake (Strobe B pulses high for two E-clock cycles.)

EGA — Active Edge for Strobe A Bit

0 = STRA falling edge selected, high level activates port C outputs (output handshake)
1 = STRA rising edge selected, low level activates port C outputs (output handshake)

INVB — Invert Strobe B Bit

0 = Active level is logic 0.
1 = Active level is logic 1.

Chapter 7
Serial Communications Interface (SCI)

7.1 Introduction

The serial communications interface (SCI) is a universal asynchronous receiver transmitter (UART), one of two independent serial input/output (I/O) subsystems in the M68HC11 E series of microcontrollers. It has a standard non-return-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.

All members of the E series contain the same SCI, with one exception. The SCI system in the MC68HC11E20 and MC68HC711E20 MCUs have an enhanced SCI baud rate generator. A divide-by-39 stage has been added that is enabled by an extra bit in the BAUD register. This increases the available SCI baud rate selections. Refer to Figure 7-8 and 7.7.5 Baud Rate Register.

7.2 Data Format

The serial data format requires these conditions:

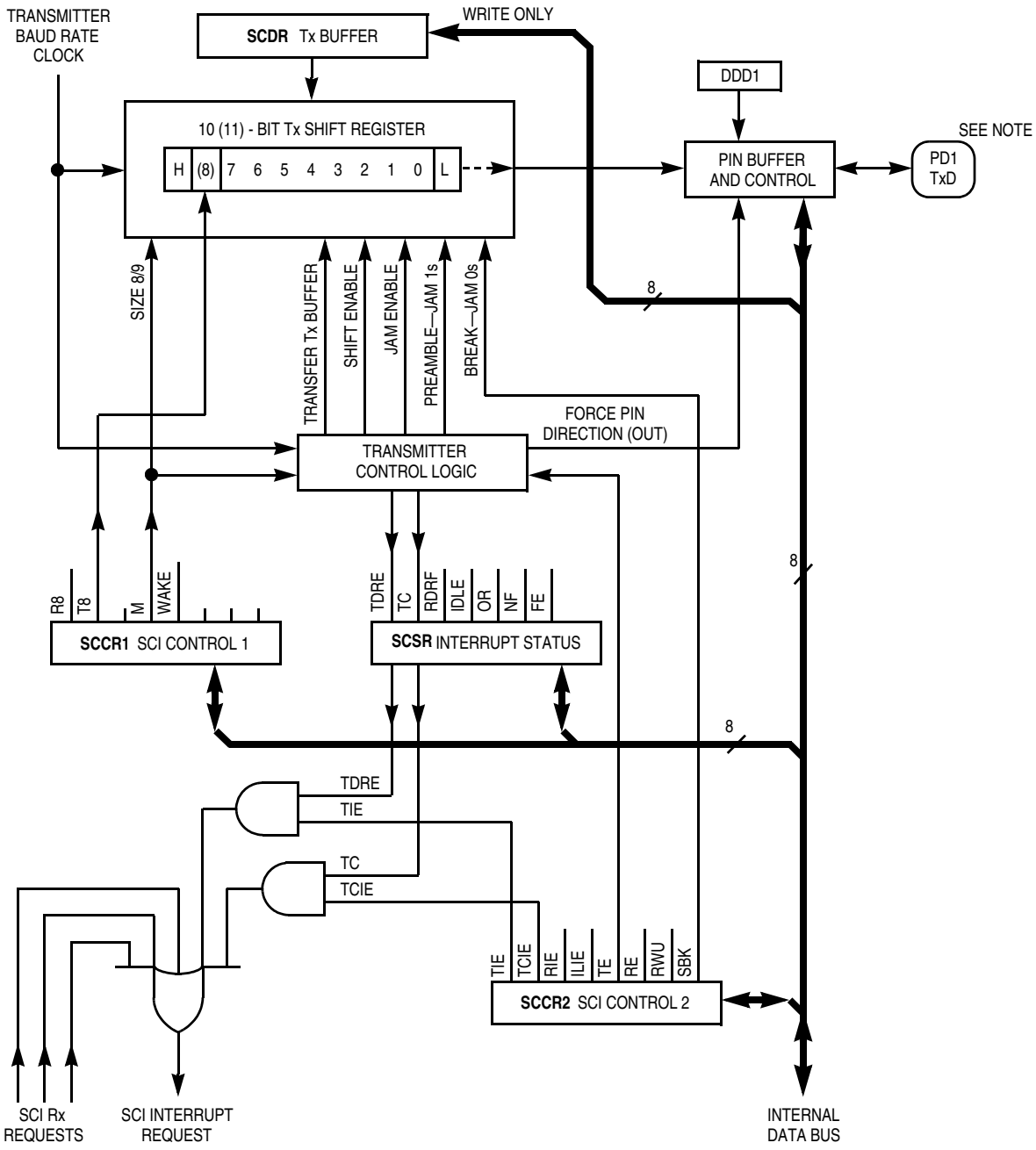
1. An idle line in the high state before transmission or reception of a message
2. A start bit, logic 0, transmitted or received, that indicates the start of each character
3. Data that is transmitted and received least significant bit (LSB) first
4. A stop bit, logic 1, used to indicate the end of a 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 0 for some multiple number of frames

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

7.3 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 be written only 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.



Note: Refer to Figure B-1. EVBU Schematic Diagram for an example of connecting TxD to a PC.

Figure 7-1. SCI Transmitter Block Diagram

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.

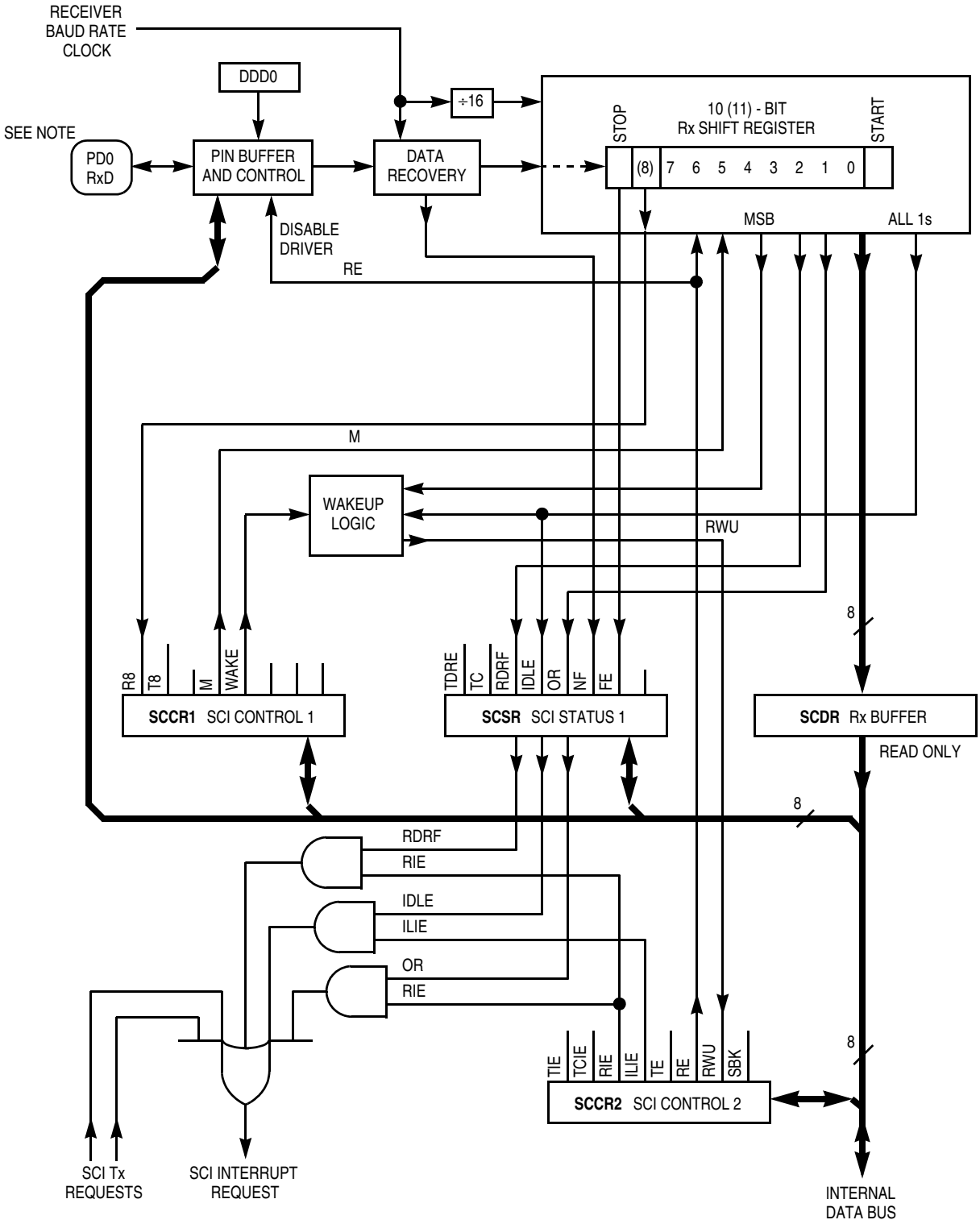
Two methods of wakeup are available:

- 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.



Note: Refer to Figure B-1. EVBU Schematic Diagram for an example of connecting RxD to a PC.

Figure 7-2. SCI Receiver Block Diagram

7.5.2 Address-Mark Wakeup

The serial characters in this type of wakeup consist of seven (eight if M = 1) information bits and an MSB, which indicates an address character (when set to 1, or mark). The first character of each message is an addressing character (MSB = 1). All receivers in the system evaluate this character to determine if the remainder of the message is directed toward this particular receiver. As soon as a receiver determines that a message is not intended for it, the receiver activates the RWU function by using a software write to set the RWU bit. Because setting RWU inhibits receiver-related flags, there is no further software overhead for the rest of this message.

When the next message begins, its first character has its MSB set, which automatically clears the RWU bit and enables normal character reception. The first character whose MSB is set is also the first character to be received after wakeup because RWU gets cleared before the stop bit for that frame is serially received. This type of wakeup allows messages to include gaps of idle time, unlike the idle-line method, but there is a loss of efficiency because of the extra bit time for each character (address bit) required for all characters.

7.6 SCI Error Detection

Three error conditions – SCDR overrun, received bit noise, and framing – can occur during generation of SCI system interrupts. Three bits (OR, NF, and FE) in the serial communications status register (SCSR) indicate if one of these error conditions exists.

The overrun error (OR) bit is set when the next byte is ready to be transferred from the receive shift register to the SCDR and the SCDR is already full (RDRF bit is set). When an overrun error occurs, the data that caused the overrun is lost and the data that was already in SCDR is not disturbed. The OR is cleared when the SCSR is read (with OR set), followed by a read of the SCDR.

The noise flag (NF) bit is set if there is noise on any of the received bits, including the start and stop bits. The NF bit is not set until the RDRF flag is set. The NF bit is cleared when the SCSR is read (with FE equal to 1) followed by a read of the SCDR.

When no stop bit is detected in the received data character, the framing error (FE) bit is set. FE is set at the same time as the RDRF. If the byte received causes both framing and overrun errors, the processor only recognizes the overrun error. The framing error flag inhibits further transfer of data into the SCDR until it is cleared. The FE bit is cleared when the SCSR is read (with FE equal to 1) followed by a read of the SCDR.

7.7 SCI Registers

Five addressable registers are associated with the SCI:

- Four control and status registers:
  - Serial communications control register 1 (SCCR1)
  - Serial communications control register 2 (SCCR2)
  - Baud rate register (BAUD)
  - Serial communications status register (SCSR)
- One data register:
  - Serial communications data register (SCDR)

The SCI registers are the same for all M68HC11 E-series devices with one exception. The SCI system for MC68HC(7)11E20 contains an extra bit in the BAUD register that provides a greater selection of baud prescaler rates. Refer to 7.7.5 Baud Rate Register, Figure 7-8, and Figure 7-9.

7.7.1 Serial Communications Data Register

SCDR is a parallel register that performs two functions:

- The receive data register when it is read
- The transmit data register when it is written

Reads access the receive data buffer and writes access the transmit data buffer. Receive and transmit are double buffered.

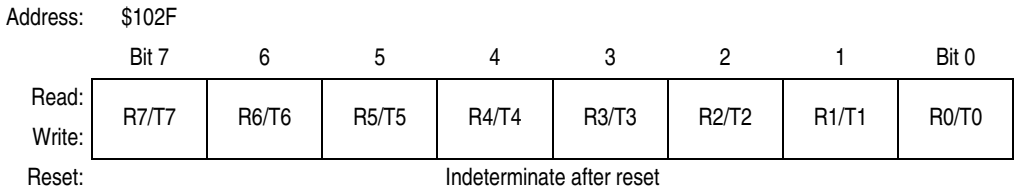


Figure 7-3. Serial Communications Data Register (SCDR)

7.7.2 Serial Communications Control Register 1

The SCCR1 register provides the control bits that determine word length and select the method used for the wakeup feature.

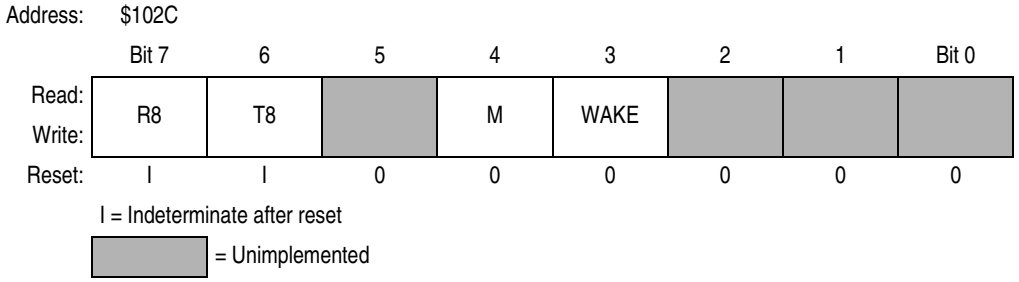


Figure 7-4. Serial Communications Control Register 1 (SCCR1)

R8 — Receive Data Bit 8

If M bit is set, R8 stores the ninth bit in the receive data character.

T8 — Transmit Data Bit 8

If M bit is set, T8 stores the ninth bit in the transmit data character.

Bit 5 — Unimplemented

Always reads 0

M — Mode Bit (select character format)

0 = Start bit, 8 data bits, 1 stop bit
1 = Start bit, 9 data bits, 1 stop bit

WAKE — Wakeup by Address Mark/Idle Bit

0 = Wakeup by IDLE line recognition
1 = Wakeup by address mark (most significant data bit set)

Bits [2:0] — Unimplemented

Always read 0

7.7.3 Serial Communications Control Register 2

The SCCR2 register provides the control bits that enable or disable individual SCI functions.

Address:	$102D							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset:	0	0	0	0	0	0	0	0

Figure 7-5. Serial Communications Control Register 2 (SCCR2)

TIE — Transmit Interrupt Enable Bit

0 = TDRE interrupts disabled
1 = SCI interrupt requested when TDRE status flag is set

TCIE — Transmit Complete Interrupt Enable Bit

0 = TC interrupts disabled
1 = SCI interrupt requested when TC status flag is set

RIE — Receiver Interrupt Enable Bit

0 = RDRF and OR interrupts disabled
1 = SCI interrupt requested when RDRF flag or the OR status flag is set

ILIE — Idle-Line Interrupt Enable Bit

0 = IDLE interrupts disabled
1 = SCI interrupt requested when IDLE status flag is set

TE — Transmitter Enable Bit

When TE goes from 0 to 1, one unit of idle character time (logic 1) is queued as a preamble.

0 = Transmitter disabled
1 = Transmitter enabled

RE — Receiver Enable Bit

0 = Receiver disabled
1 = Receiver enabled

RWU — Receiver Wakeup Control Bit

0 = Normal SCI receiver
1 = Wakeup enabled and receiver interrupts inhibited

SBK — Send Break

At least one character time of break is queued and sent each time SBK is written to 1. As long as the SBK bit is set, break characters are queued and sent. More than one break may be sent if the transmitter is idle at the time the SBK bit is toggled on and off, as the baud rate clock edge could occur between writing the 1 and writing the 0 to SBK.

0 = Break generator off
1 = Break codes generated

7.7.4 Serial Communication Status Register

The SCSR provides inputs to the interrupt logic circuits for generation of the SCI system interrupt.

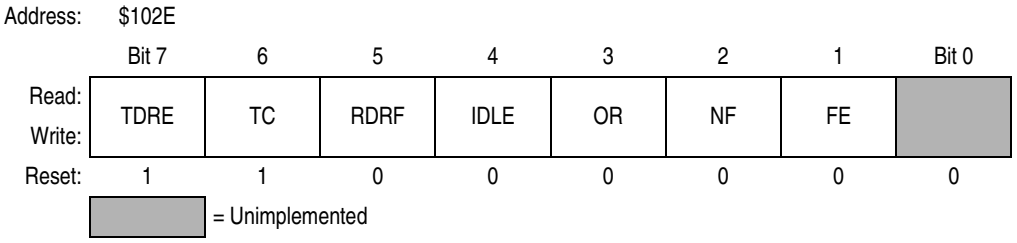


Figure 7-6. Serial Communications Status Register (SCSR)

TDRE — Transmit Data Register Empty Flag

This flag is set when SCDR is empty. Clear the TDRE flag by reading SCSR with TDRE set and then writing to SCDR.

0 = SCDR busy
0 = SCDR empty

TC — Transmit Complete Flag

This flag is set when the transmitter is idle (no data, preamble, or break transmission in progress). Clear the TC flag by reading SCSR with TC set and then writing to SCDR.

0 = Transmitter busy
1 = Transmitter idle

RDRF — Receive Data Register Full Flag

This flag is set if a received character is ready to be read from SCDR. Clear the RDRF flag by reading SCSR with RDRF set and then reading SCDR.

0 = SCDR empty
1 = SCDR full

IDLE — Idle Line Detected Flag

This flag is set if the RxD line is idle. Once cleared, IDLE is not set again until the RxD line has been active and becomes idle again. The IDLE flag is inhibited when RWU = 1. Clear IDLE by reading SCSR with IDLE set and then reading SCDR.

0 = RxD line active
1 = RxD line idle

OR — Overrun Error Flag

OR is set if a new character is received before a previously received character is read from SCDR. Clear the OR flag by reading SCSR with OR set and then reading SCDR.

0 = No overrun
1 = Overrun detected

NF — Noise Error Flag

NF is set if majority sample logic detects anything other than a unanimous decision. Clear NF by reading SCSR with NF set and then reading SCDR.

0 = Unanimous decision
1 = Noise detected

FE — Framing Error Flag

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

0 = Stop bit detected
1 = Zero detected

Bit 0 — Unimplemented

Always reads 0

7.7.5 Baud Rate Register

Use this register to select different baud rates for the SCI system. The SCP[1:0] (SCP[2:0] in MC68HC(7)11E20) bits function as a prescaler for the SCR[2:0] bits. Together, these five bits provide multiple baud rate combinations for a given crystal frequency. Normally, this register is written once during initialization. The prescaler is set to its fastest rate by default out of reset and can be changed at any time. Refer to Table 7-1 for normal baud rate selections.

Address: $102B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TCLR	SCP2	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0
Reset:	0	0	0	0	0	U	U	U

U = Unaffected

Figure 7-7. Baud Rate Register (BAUD)

TCLR — Clear Baud Rate Counter Bit (Test)

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

NOTE

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

RCKB — SCI Baud Rate Clock Check Bit (Test)

See Table 7-1.

Table 7-1. Baud Rate Values

Prescaler Selects						Prescale Divide	Baud Set Divide	Crystal Frequency (MHz)					
								4.00	4.9152	8.00	10.00	12.00	16.00
								Bus Frequency (MHz)					
SCP2	SCP1	SCP0	SCR2	SCR1	SCR0			1.00	1.23	2.00	2.50	3.00	4.00
0	0	0	0	0	0	1	1	62500	76800	125000	156250	187500	250000
0	0	0	0	0	1	1	2	31250	38400	62500	78125	93750	125000
0	0	0	0	1	0	1	4	15625	19200	31250	39063	46875	62500
0	0	0	0	1	1	1	8	7813	9600	15625	19531	23438	31250
0	0	0	1	0	0	1	16	3906	4800	7813	9766	11719	15625
0	0	0	1	0	1	1	32	1953	2400	3906	4883	5859	7813
0	0	0	1	1	0	1	64	977	1200	1953	2441	2930	3906
0	0	0	1	1	1	1	128	488	600	977	1221	1465	1953
0	0	1	0	0	0	3	1	20833	25600	41667	52083	62500	83333
0	0	1	0	0	1	3	2	10417	12800	20833	26042	31250	41667
0	0	1	0	1	0	3	4	5208	6400	10417	13021	15625	20833
0	0	1	0	1	1	3	8	2604	3200	5208	6510	7813	10417
0	0	1	1	0	0	3	16	1302	1600	2604	3255	3906	5208
0	0	1	1	0	1	3	32	651	800	1302	1628	1953	2604
0	0	1	1	1	0	3	64	326	400	651	814	977	1302
0	0	1	1	1	1	3	128	163	200	326	407	488	651
0	1	0	0	0	0	4	1	15625	19200	31250	39063	46875	62500
0	1	0	0	0	1	4	2	7813	9600	15625	19531	23438	31250
0	1	0	0	1	0	4	4	3906	4800	7813	9766	11719	15625
0	1	0	0	1	1	4	8	1953	2400	3906	4883	5859	7813
0	1	0	1	0	0	4	16	977	1200	1953	2441	2930	3906
0	1	0	1	0	1	4	32	488	600	977	1221	1465	1953
0	1	0	1	1	0	4	64	244	300	488	610	732	977
0	1	0	1	1	1	4	128	122	150	244	305	366	488
0	1	1	0	0	0	13	1	4808	5908	9615	12019	14423	19231
0	1	1	0	0	1	13	2	2404	2954	4808	6010	7212	9615
0	1	1	0	1	0	13	4	1202	1477	2404	3005	3606	4808
0	1	1	0	1	1	13	8	601	738	1202	1502	1803	2404
0	1	1	1	0	0	13	16	300	369	601	751	901	1202
0	1	1	1	0	1	13	32	150	185	300	376	451	601
0	1	1	1	1	0	13	64	75	92	150	188	225	300
0	1	1	1	1	1	13	128	38	46	75	94	113	150
1	0	0	0	0	0	39	1	1603	1969	3205	4006	4808	6410
1	0	0	0	0	1	39	2	801	985	1603	2003	2404	3205
1	0	0	0	1	0	39	4	401	492	801	1002	1202	1603
1	0	0	0	1	1	39	8	200	246	401	501	601	801
1	0	0	1	0	0	39	16	100	123	200	250	300	401
1	0	0	1	0	1	39	32	50	62	100	125	150	200
1	0	0	1	1	0	39	64	25	31	50	63	75	100
1	0	0	1	1	1	39	128	13	15	25	31	38	50

Shaded areas reflect standard baud rates.
On MC68HC(7)11E20 do not set SCP1 or SCP0 when SCP2 is 1.

SCR[2:0] — SCI Baud Rate Select Bits

Selects receiver and transmitter bit rate based on output from baud rate prescaler stage. Refer to Figure 7-8 and Figure 7-9.

The prescaler bits, SCP[2:0], determine the highest baud rate, and the SCR[2:0] bits select an additional binary submultiple (÷1, ÷2, ÷4, through ÷128) of this highest baud rate. The result of these two dividers in series is the 16X receiver baud rate clock. The SCR[2:0] bits are not affected by reset and can be changed at any time, although they should not be changed when any SCI transfer is in progress.

Figure 7-8 and Figure 7-9 illustrate the SCI baud rate timing chain. The prescaler select bits determine the highest baud rate. The rate select bits determine additional divide by two stages to arrive at the receiver timing (RT) clock rate. The baud rate clock is the result of dividing the RT clock by 16.

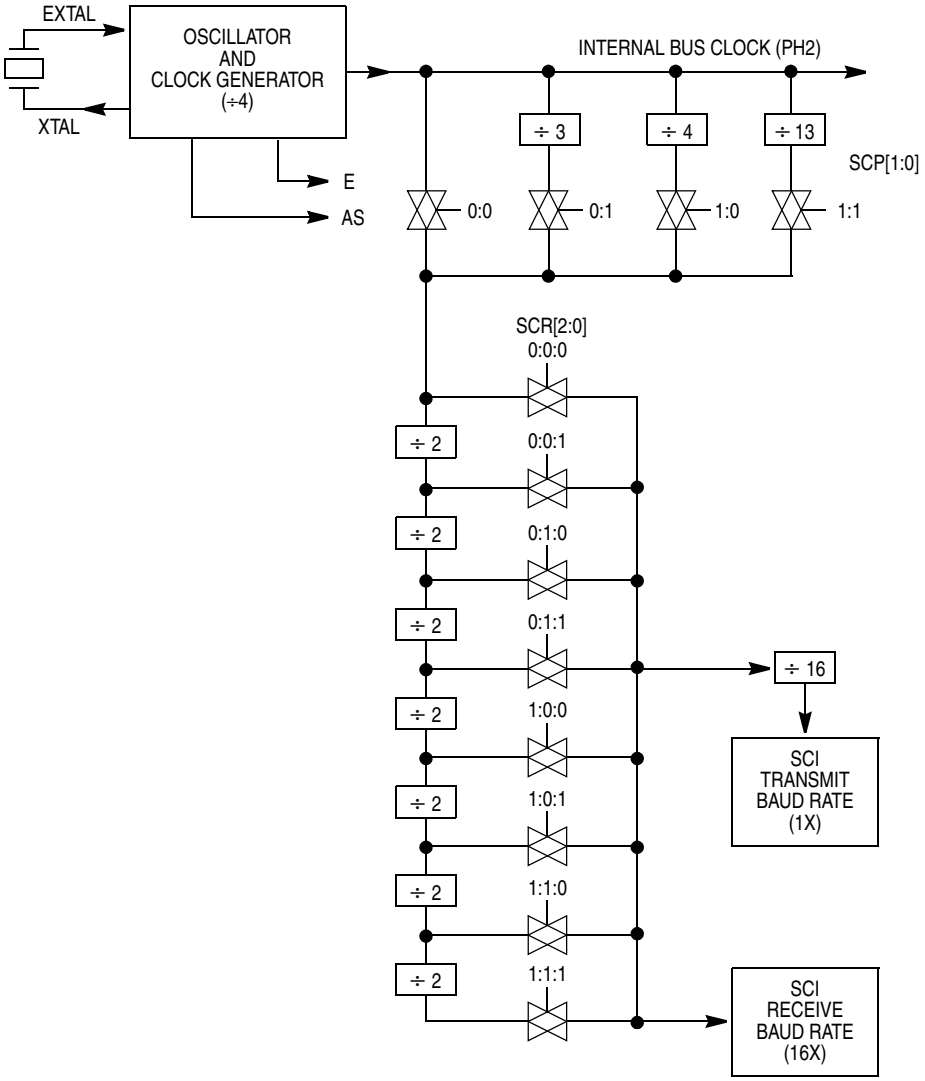
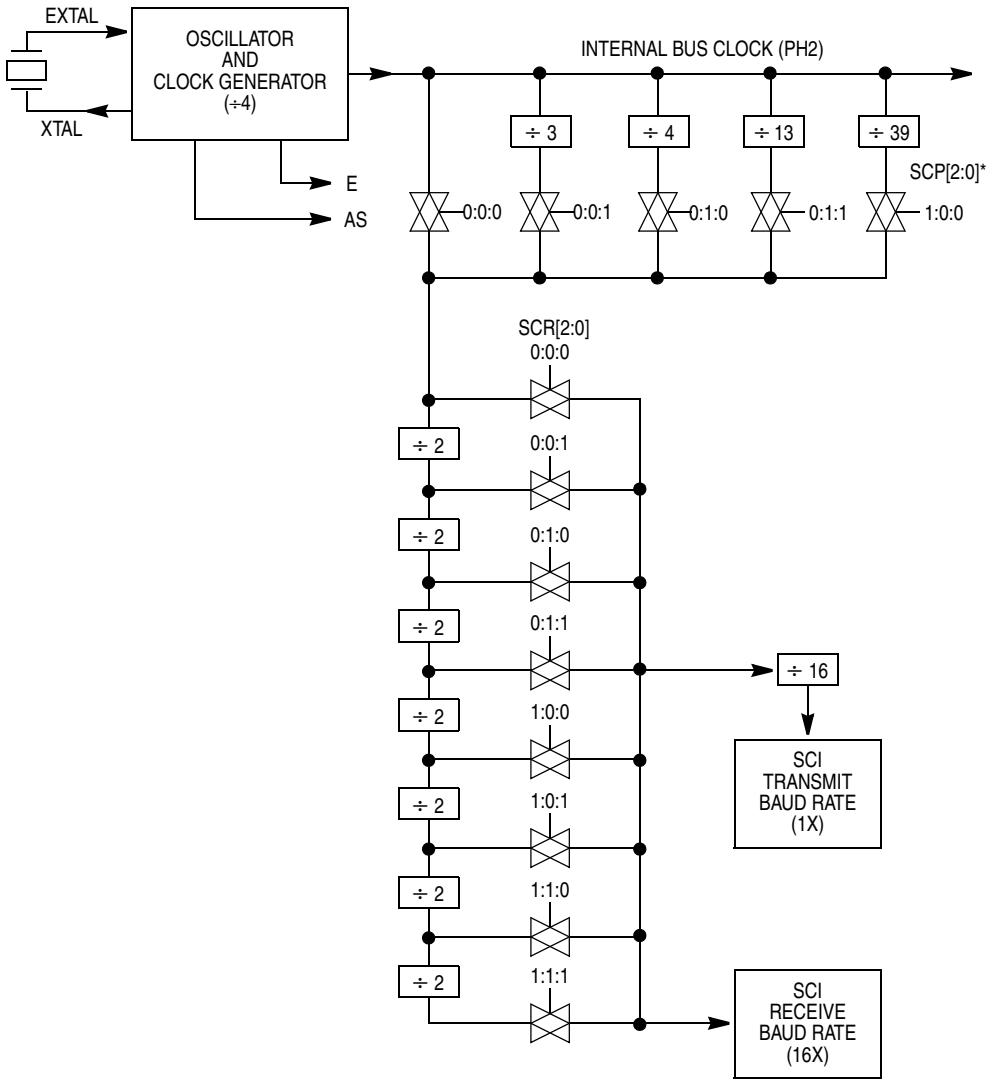


Figure 7-8. SCI Baud Rate Generator Block Diagram



*SCP2 is present only on MC68HC(7)11E20.

Figure 7-9. MC68HC(7)11E20 SCI Baud Rate Generator Block Diagram

7.8 Status Flags and Interrupts

The SCI transmitter has two status flags. These status flags can be read by software (polled) to tell when the corresponding condition exists. Alternatively, a local interrupt enable bit can be set to enable each of these status conditions to generate interrupt requests when the corresponding condition is present. Status flags are automatically set by hardware logic conditions, but must be cleared by software, which provides an interlock mechanism that enables logic to know when software has noticed the status indication. The software clearing sequence for these flags is automatic. Functions that are normally performed in response to the status flags also satisfy the conditions of the clearing sequence.

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

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

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

7.9 Receiver Flags

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

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

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

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

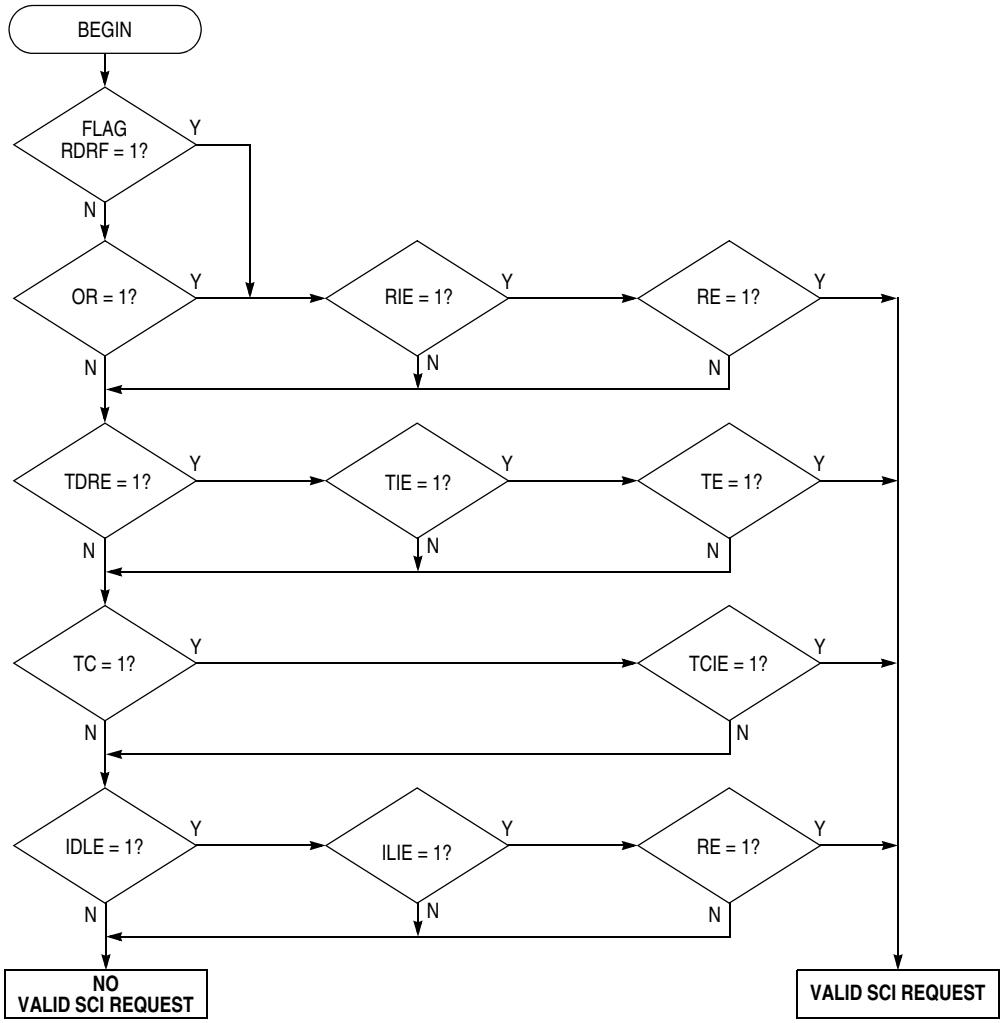


Figure 7-10. Interrupt Source Resolution Within SCI

Chapter 8
Serial Peripheral Interface (SPI)

8.1 Introduction

The serial peripheral interface (SPI), an independent serial communications subsystem, allows the MCU to communicate synchronously with peripheral devices, such as:

- Frequency synthesizers
- Liquid crystal display (LCD) drivers
- Analog-to-digital (A/D) converter subsystems
- Other microprocessors

The SPI is also capable of inter-processor communication in a multiple master system. The SPI system can be configured as either a master or a slave device. When configured as a master, data transfer rates can be as high as one-half the E-clock rate (1.5 Mbits per second for a 3-MHz bus frequency). When configured as a slave, data transfers can be as fast as the E-clock rate (3 Mbits per second for a 3-MHz bus frequency).

8.2 Functional Description

The central element in the SPI system is the block containing the shift register and the read data buffer. The system is single buffered in the transmit direction and double buffered in the receive direction. This means that new data for transmission cannot be written to the shifter until the previous transfer is complete; however, received data is transferred into a parallel read data buffer so the shifter is free to accept a second serial character. As long as the first character is read out of the read data buffer before the next serial character is ready to be transferred, no overrun condition occurs. A single MCU register address is used for reading data from the read data buffer and for writing data to the shifter.

The SPI status block represents the SPI status functions (transfer complete, write collision, and mode fault) performed by the serial peripheral status register (SPSR). The SPI control block represents those functions that control the SPI system through the serial peripheral control register (SPCR).

Refer to Figure 8-1, which shows the SPI block diagram.

8.3 SPI Transfer Formats

During an SPI transfer, data is simultaneously transmitted and received. A serial clock line synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows individual selection of a slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the select line can optionally be used to indicate a multiple master bus contention. Refer to Figure 8-2.

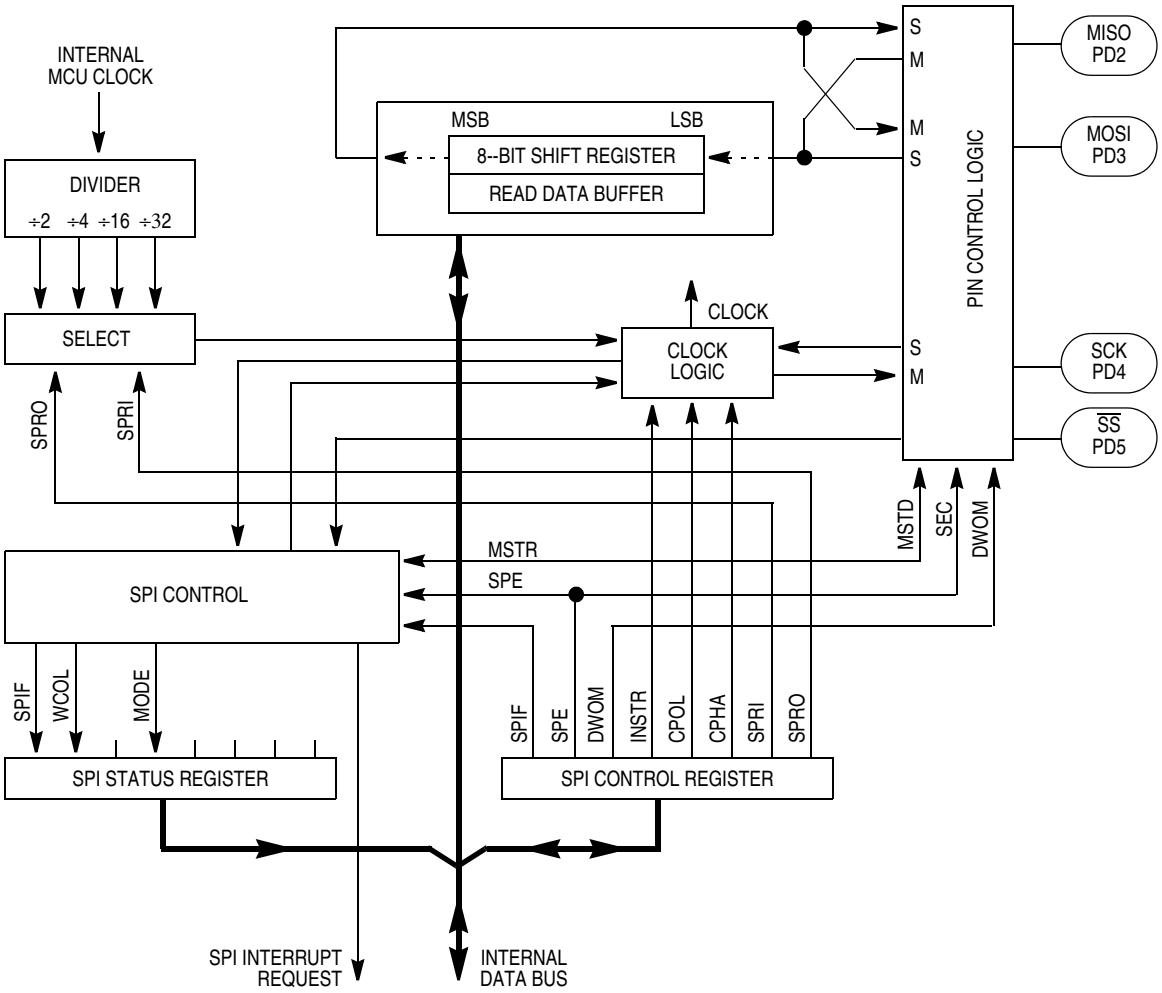


Figure 8-1. SPI Block Diagram

8.4 Clock Phase and Polarity Controls

Software can select one of four combinations of serial clock phase and polarity using two bits in the SPI control register (SPCR). The clock polarity is specified by the CPOL control bit, which selects an active high or active low clock, and has no significant effect on the transfer format. The clock phase (CPHA) control bit selects one of two different transfer formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transfers to allow a master device to communicate with peripheral slaves having different requirements.

When CPHA equals 0, the $\overline{SS}$ line must be negated and reasserted between each successive serial byte. Also, if the slave writes data to the SPI data register (SPDR) while $\overline{SS}$ is low, a write collision error results.

When CPHA equals 1, the $\overline{SS}$ line can remain low between successive transfers.

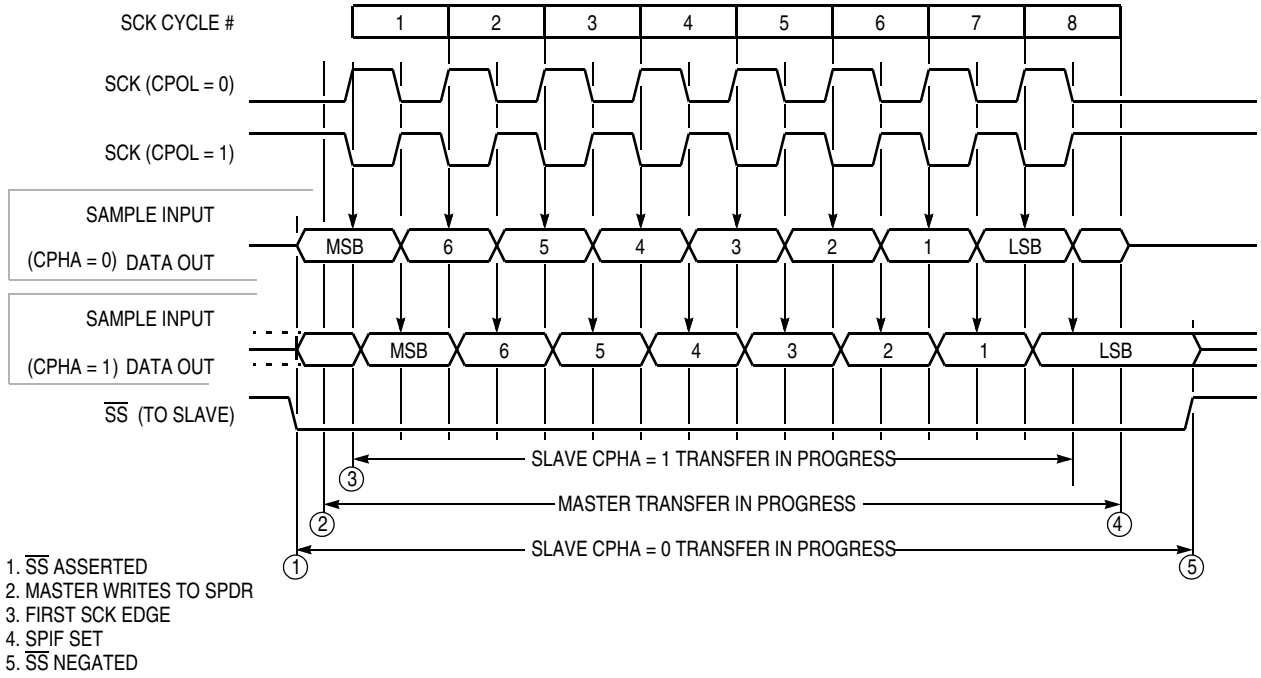


Figure 8-2. SPI Transfer Format

8.5 SPI Signals

This subsection contains descriptions of the four SPI signals:

- Master in/slave out (MISO)
- Master out/slave in (MOSI)
- Serial clock (SCK)
- Slave select ($\overline{SS}$)

Any SPI output line must have its corresponding data direction bit in DDRD register set. If the DDR bit is clear, that line is disconnected from the SPI logic and becomes a general-purpose input. All SPI input lines are forced to act as inputs regardless of the state of the corresponding DDR bits in DDRD register.

8.5.1 Master In/Slave Out

MISO is one of two unidirectional serial data signals. It is an input to a master device and an output from a slave device. The MISO line of a slave device is placed in the high-impedance state if the slave device is not selected.

8.5.2 Master Out/Slave In

The MOSI line is the second of the two unidirectional serial data signals. It is an output from a master device and an input to a slave device. The master device places data on the MOSI line a half-cycle before the clock edge that the slave device uses to latch the data.

8.5.3 Serial Clock

SCK, an input to a slave device, is generated by the master device and synchronizes data movement in and out of the device through the MOSI and MISO lines. Master and slave devices are capable of exchanging a byte of information during a sequence of eight clock cycles.

Four possible timing relationships can be chosen by using control bits CPOL and CPHA in the serial peripheral control register (SPCR). Both master and slave devices must operate with the same timing. The SPI clock rate select bits, SPR[1:0], in the SPCR of the master device, select the clock rate. In a slave device, SPR[1:0] have no effect on the operation of the SPI.

8.5.4 Slave Select

The slave select ($\overline{SS}$) input of a slave device must be externally asserted before a master device can exchange data with the slave device. $\overline{SS}$ must be low before data transactions and must stay low for the duration of the transaction.

The $\overline{SS}$ line of the master must be held high. If it goes low, a mode fault error flag (MODF) is set in the serial peripheral status register (SPSR). To disable the mode fault circuit, write a 1 in bit 5 of the port D data direction register. This sets the $\overline{SS}$ pin to act as a general-purpose output rather than the dedicated input to the slave select circuit, thus inhibiting the mode fault flag. The other three lines are dedicated to the SPI whenever the serial peripheral interface is on.

The state of the master and slave CPHA bits affects the operation of $\overline{SS}$. CPHA settings should be identical for master and slave. When CPHA = 0, the shift clock is the OR of $\overline{SS}$ with SCK. In this clock phase mode, $\overline{SS}$ must go high between successive characters in an SPI message. When CPHA = 1, $\overline{SS}$ can be left low between successive SPI characters. In cases where there is only one SPI slave MCU, its $\overline{SS}$ line can be tied to V_{SS} as long as only CPHA = 1 clock mode is used.

8.6 SPI System Errors

Two system errors can be detected by the SPI system. The first type of error arises in a multiple-master system when more than one SPI device simultaneously tries to be a master. This error is called a mode fault. The second type of error, write collision, indicates that an attempt was made to write data to the SPDR while a transfer was in progress.

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 0, 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 1, 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 1 when SPIF is set.

8.7 SPI Registers

The three SPI registers are:

- Serial peripheral control register (SPCR)
- Serial peripheral status register (SPSR)
- Serial peripheral data register (SPDR)

These registers provide control, status, and data storage functions.

8.7.1 Serial Peripheral Control Register

Address: $1028

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

U = Unaffected

Figure 8-3. Serial Peripheral Control Register (SPCR)

SPIE — Serial Peripheral Interrupt Enable Bit

Set the SPE bit to 1 to request a hardware interrupt sequence each time the SPIF or MODF status flag is set. SPI interrupts are inhibited if this bit is clear or if the I bit in the condition code register is 1.

0 = SPI system interrupts disabled
1 = SPI system interrupts enabled

SPE — Serial Peripheral System Enable Bit

When the SPE bit is set, the port D bit 2, 3, 4, and 5 pins are dedicated to the SPI function. If the SPI is in the master mode and DDRD bit 5 is set, then the port D bit 5 pin becomes a general-purpose output instead of the $\overline{SS}$ input.

0 = SPI system disabled
1 = SPI system enabled

DWOM — Port D Wired-OR Mode Bit

DWOM affects all port D pins.

0 = Normal CMOS outputs
1 = Open-drain outputs

MSTR — Master Mode Select Bit

It is customary to have an external pullup resistor on lines that are driven by open-drain devices.

0 = Slave mode
1 = Master mode

CPOL — Clock Polarity Bit

When the clock polarity bit is cleared and data is not being transferred, the SCK pin of the master device has a steady state low value. When CPOL is set, SCK idles high. Refer to Figure 8-2 and 8.4 Clock Phase and Polarity Controls.

CPHA — Clock Phase Bit

The clock phase bit, in conjunction with the CPOL bit, controls the clock-data relationship between master and slave. The CPHA bit selects one of two different clocking protocols. Refer to Figure 8-2 and 8.4 Clock Phase and Polarity Controls.

SPR[1:0] — SPI Clock Rate Select Bits

These two bits select the SPI clock (SCK) rate when the device is configured as master. When the device is configured as slave, these bits have no effect. Refer to Table 8-1.

Table 8-1. SPI Clock Rates

SPR[1:0]	Divide E Clock By	Frequency at E = 1 MHz (Baud)	Frequency at E = 2 MHz (Baud)	Frequency at E = 3 MHz (Baud)	Frequency at E = 4 MHz (Baud)
0 0	2	500 kHz	1.0 MHz	1.5 MHz	2 MHz
0 1	4	250 kHz	500 kHz	750 kHz	1 MHz
1 0	16	62.5 kHz	125 kHz	187.5 kHz	250 kHz
1 1	32	31.3 kHz	62.5 kHz	93.8 kHz	125 kHz

8.7.2 Serial Peripheral Status Register

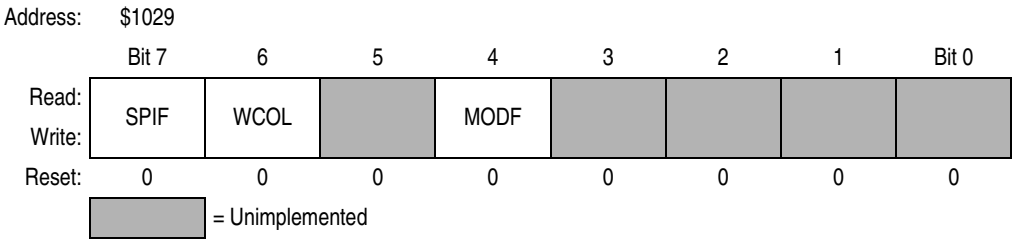


Figure 8-4. Serial Peripheral Status Register (SPSR)

SPIF — SPI Interrupt Complete Flag

SPIF is set upon completion of data transfer between the processor and the external device. If SPIF goes high, and if SPIE is set, a serial peripheral interrupt is generated. To clear the SPIF bit, read the SPSR with SPIF set, then access the SPDR. Unless SPSR is read (with SPIF set) first, attempts to write SPDR are inhibited.

WCOL — Write Collision Bit

Clearing the WCOL bit is accomplished by reading the SPSR (with WCOL set) followed by an access of SPDR. Refer to 8.5.4 Slave Select and 8.6 SPI System Errors.

0 = No write collision
1 = Write collision

Bit 5 — Unimplemented

Always reads 0

MODF — Mode Fault Bit

To clear the MODF bit, read the SPSR (with MODF set), then write to the SPCR. Refer to 8.5.4 Slave Select and 8.6 SPI System Errors.

0 = No mode fault
1 = Mode fault

Bits [3:0] — Unimplemented

Always read 0

8.7.3 Serial Peripheral Data I/O Register

The SPDR is used when transmitting or receiving data on the serial bus. Only a write to this register initiates transmission or reception of a byte, and this only occurs in the master device. At the completion of transferring a byte of data, the SPIF status bit is set in both the master and slave devices.

A read of the SPDR is actually a read of a buffer. To prevent an overrun and the loss of the byte that caused the overrun, the first SPIF must be cleared by the time a second transfer of data from the shift register to the read buffer is initiated.

Address: $102A

	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:	Indeterminate after reset							

Figure 8-5. Serial Peripheral Data I/O Register (SPDR)

SPI is double buffered in and single buffered out.

Chapter 9
Timing Systems

9.1 Introduction

The M68HC11 timing system is composed of five clock divider chains. The main clock divider chain includes a 16-bit free-running counter, which is driven by a programmable prescaler. The main timer's programmable prescaler provides one of the four clocking rates to drive the 16-bit counter. Two prescaler control bits select the prescale rate.

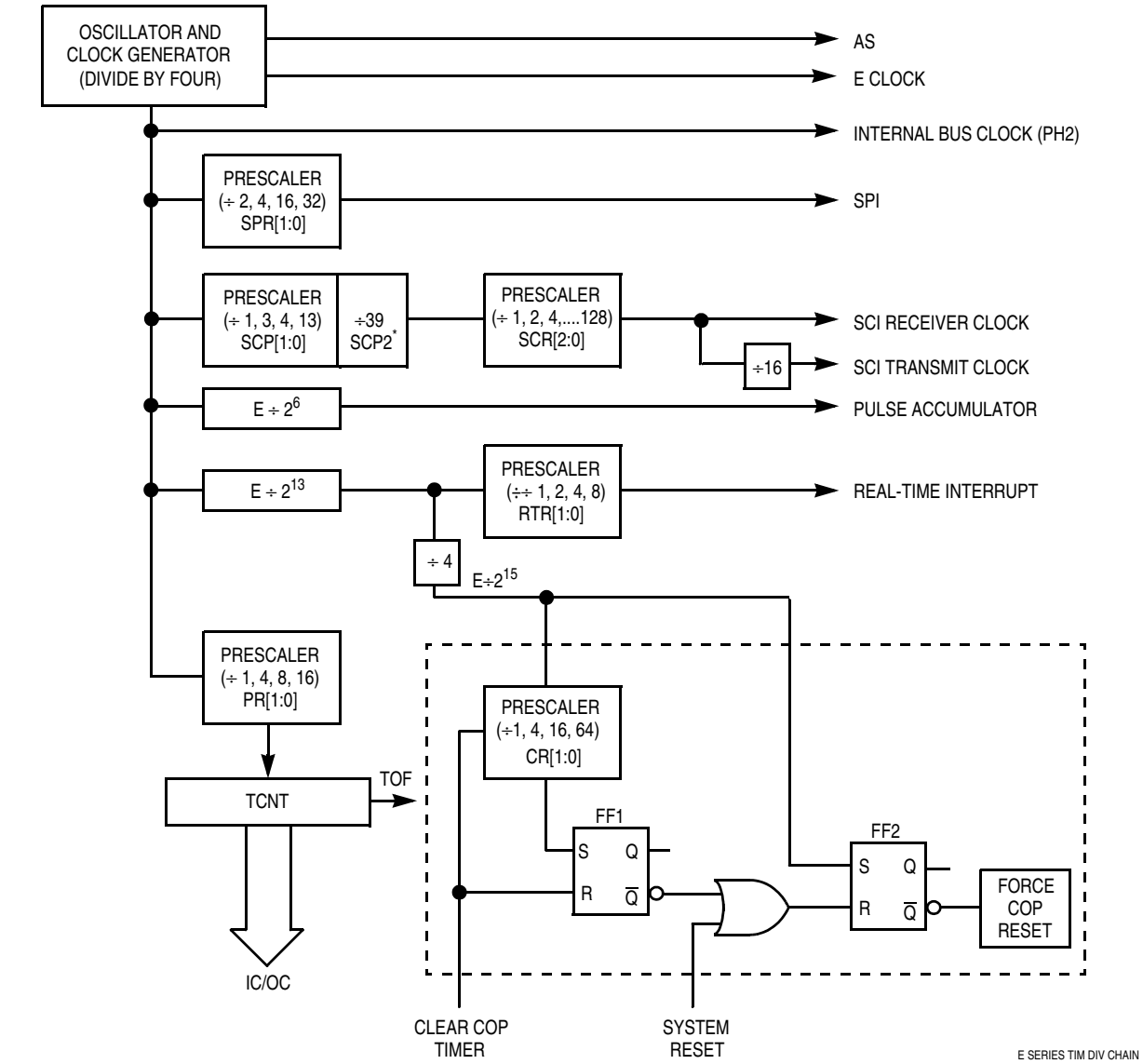
The prescaler output divides the system clock by 1, 4, 8, or 16. Taps off of this main clocking chain drive circuitry that generates the slower clocks used by the pulse accumulator, the real-time interrupt (RTI), and the computer operating properly (COP) watchdog subsystems, also described in this section. Refer to Figure 9-1.

All main timer system activities are referenced to this free-running counter. The counter begins incrementing from $0000 as the MCU comes out of reset and continues to the maximum count, $FFFF. At the maximum count, the counter rolls over to $0000, sets an overflow flag, and continues to increment. As long as the MCU is running in a normal operating mode, there is no way to reset, change, or interrupt the counting. The capture/compare subsystem features three input capture channels, four output compare channels, and one channel that can be selected to perform either input capture or output compare. Each of the three input capture functions has its own 16-bit input capture register (time capture latch) and each of the output compare functions has its own 16-bit compare register. All timer functions, including the timer overflow and RTI, have their own interrupt controls and separate interrupt vectors.

The pulse accumulator contains an 8-bit counter and edge select logic. The pulse accumulator can operate in either event counting mode or gated time accumulation mode. During event counting mode, the pulse accumulator's 8-bit counter increments when a specified edge is detected on an input signal. During gated time accumulation mode, an internal clock source increments the 8-bit counter while an input signal has a predetermined logic level.

The real-time interrupt (RTI) is a programmable periodic interrupt circuit that permits pacing the execution of software routines by selecting one of four interrupt rates.

The COP watchdog clock input ($E \div 2^{15}$) is tapped off of the free-running counter chain. The COP automatically times out unless it is serviced within a specific time by a program reset sequence. If the COP is allowed to time out, a reset is generated, which drives the $\overline{\text{RESET}}$ pin low to reset the MCU and the external system. Refer to Table 9-1 for crystal-related frequencies and periods.



* SCP2 present on MC68HC(7)11E20 only

Figure 9-1. Timer Clock Divider Chains

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.

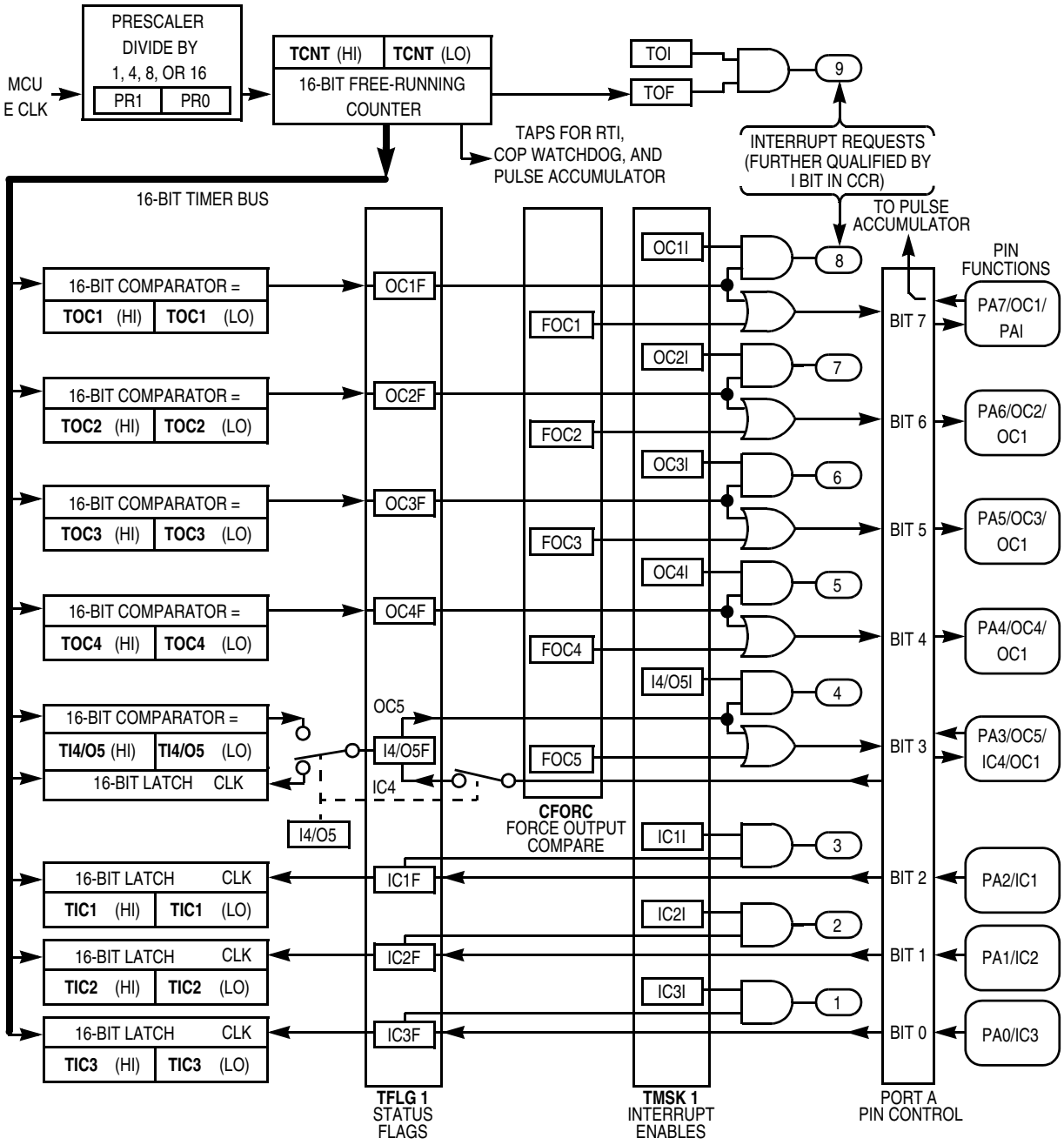


Figure 9-2. Capture/Compare Block Diagram

The control and status bits that implement the input capture functions are contained in:

- Pulse accumulator control register (PACTL)
- Timer control 2 register (TCTL2)
- Timer interrupt mask 1 register (TMSK1)
- Timer interrupt flag 2 register (TFLG1)

To configure port A bit 3 as an input capture, clear the DDRA3 bit of the PACTL register. Note that this bit is cleared out of reset. To enable PA3 as the fourth input capture, set the I4/O5 bit in the PACTL register. Otherwise, PA3 is configured as a fifth output compare out of reset, with bit I4/O5 being cleared. If the DDRA3 bit is set (configuring PA3 as an output), and IC4 is enabled, then writes to PA3 cause edges on the pin to result in input captures. Writing to TI4/O5 has no effect when the TI4/O5 register is acting as IC4.

9.3.1 Timer Control Register 2

Use the control bits of this register to program input capture functions to detect a particular edge polarity on the corresponding timer input pin. Each of the input capture functions can be independently configured to detect rising edges only, falling edges only, any edge (rising or falling), or to disable the input capture function. The input capture functions operate independently of each other and can capture the same TCNT value if the input edges are detected within the same timer count cycle.

Address: $1021

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A
Reset:	0	0	0	0	0	0	0	0

Figure 9-3. Timer Control Register 2 (TCTL2)

EDGxB and EDGxA — Input Capture Edge Control Bits

There are four pairs of these bits. Each pair is cleared to 0 by reset and must be encoded to configure the corresponding input capture edge detector circuit. IC4 functions only if the I4/O5 bit in the PACTL register is set. Refer to Table 9-2 for timer control configuration.

Table 9-2. Timer Control Configuration

EDGxB	EDGxA	Configuration
0	0	Capture disabled
0	1	Capture on rising edges only
1	0	Capture on falling edges only
1	1	Capture on any edge

9.3.2 Timer Input Capture Registers

When an edge has been detected and synchronized, the 16-bit free-running counter value is transferred into the input capture register pair as a single 16-bit parallel transfer. Timer counter value captures and timer counter incrementing occur on opposite half-cycles of the phase 2 clock so that the count value is stable whenever a capture occurs. The timer input capture registers are not affected by reset. Input capture values can be read from a pair of 8-bit read-only registers. A read of the high-order byte of an

input capture register pair inhibits a new capture transfer for one bus cycle. If a double-byte read instruction, such as load double accumulator D (LDD), is used to read the captured value, coherency is assured. When a new input capture occurs immediately after a high-order byte read, transfer is delayed for an additional cycle but the value is not lost.

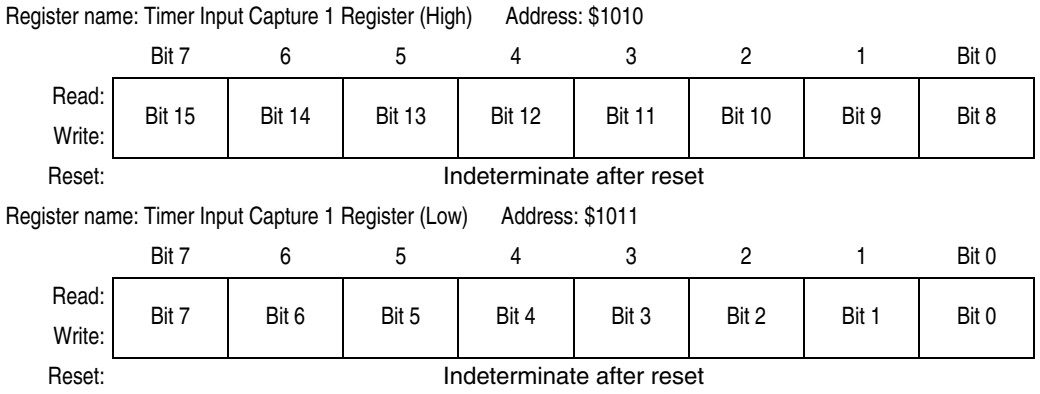


Figure 9-4. Timer Input Capture 1 Register Pair (TIC1)

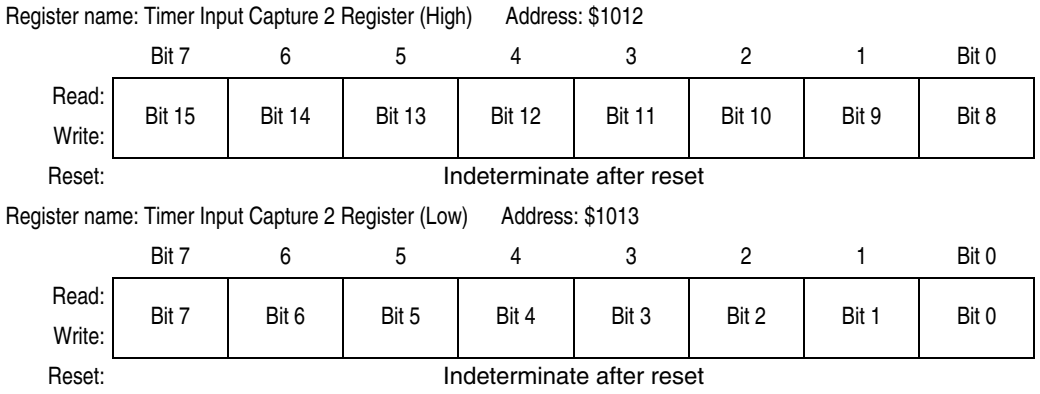


Figure 9-5. Timer Input Capture 2 Register Pair (TIC2)

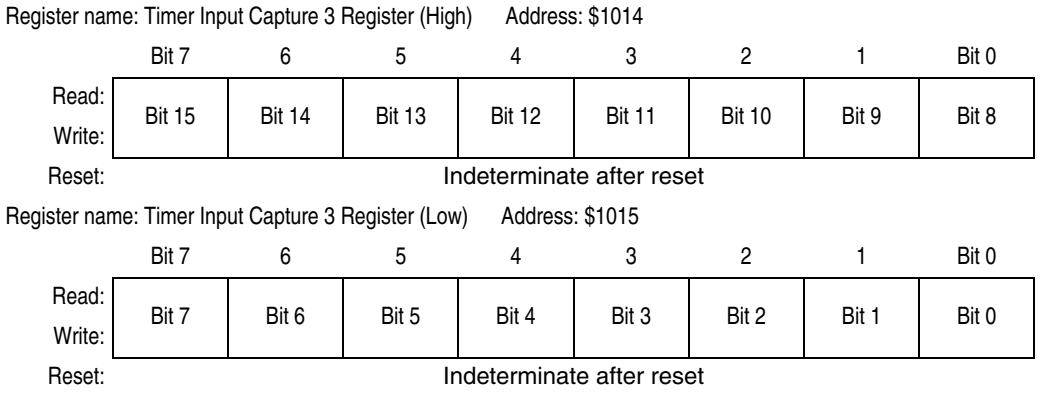


Figure 9-6. Timer Input Capture 3 Register Pair (TIC3)

9.3.3 Timer Input Capture 4/Output Compare 5 Register

Use TI4/O5 as either an input capture register or an output compare register, depending on the function chosen for the PA3 pin. To enable it as an input capture pin, set the I4/O5 bit in the pulse accumulator control register (PACTL) to logic level 1. To use it as an output compare register, set the I4/O5 bit to a logic level 0. Refer to 9.7 Pulse Accumulator.

Register name: Timer Input Capture 4/Output Compare 5 (High) Address: $101E

	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 Input Capture 4/Output Compare 5 (Low) Address: $101F

	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 9-7. Timer Input Capture 4/Output Compare 5 Register Pair (TI4/O5)

9.4 Output Compare

Use the output compare (OC) function to program an action to occur at a specific time — when the 16-bit counter reaches a specified value. For each of the five output compare functions, there is a separate 16-bit compare register and a dedicated 16-bit comparator. The value in the compare register is compared to the value of the free-running counter on every bus cycle. When the compare register matches the counter value, an output compare status flag is set. The flag can be used to initiate the automatic actions for that output compare function.

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.

The five 16-bit read/write output compare registers are: TOC1, TOC2, TOC3, and TOC4, and the TI4/O5. TI4/O5 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.

9.4.1 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)

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

	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 3 Register (Low) Address: $101B

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

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

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

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

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

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

9.4.2 Timer Compare Force Register

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

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

Address: $100B

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

= Unimplemented

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

FOC[1:5] — Force Output Comparison Bit

When the FOC bit associated with an output compare circuit is set, the output compare circuit immediately performs the action it is programmed to do when an output match occurs.

0 = Not affected
1 = Output x action occurs

Bits [2:0] — Unimplemented

Always read 0

9.4.3 Output Compare Mask Register

Use OC1M with OC1 to specify the bits of port A that are affected by a successful OC1 compare. The bits of the OC1M register correspond to PA[7:3].

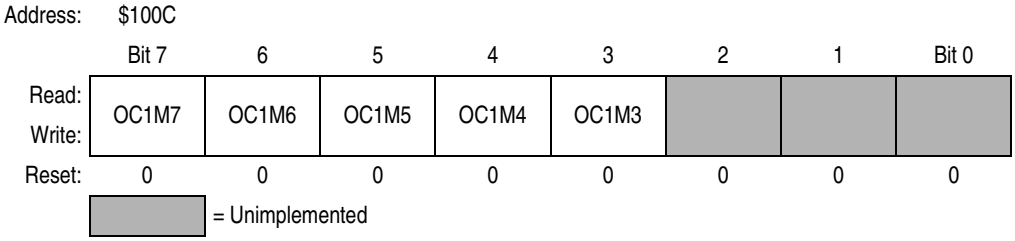


Figure 9-13. Output Compare 1 Mask Register (OC1M)

OC1M[7:3] — Output Compare Masks

0 = OC1 disabled
1 = OC1 enabled to control the corresponding pin of port A

Bits [2:0] — Unimplemented

Always read 0

9.4.4 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.

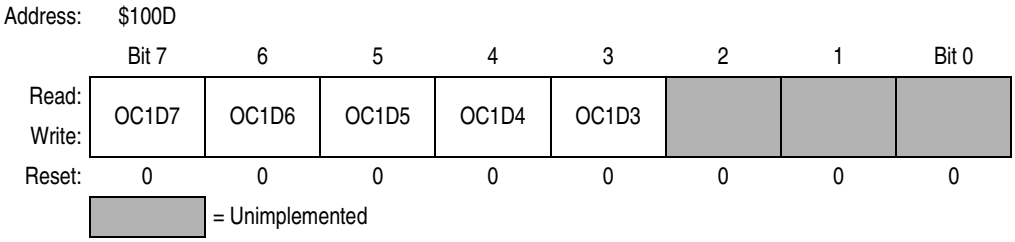


Figure 9-14. Output Compare 1 Data Register (OC1D)

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

Bits [2:0] — Unimplemented

Always read 0

9.4.5 Timer Counter Register

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

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: Write:	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5
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

9.4.7 Timer Interrupt Mask 1 Register

Use this 8-bit register to enable or inhibit the timer input capture and output compare interrupts.

Address: $1022

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OC1I	OC2I	OC3I	OC4I	I4/O5I	IC1I	IC2I	IC3I
Reset:	0	0	0	0	0	0	0	0

Figure 9-17. Timer Interrupt Mask 1 Register (TMSK1)

OC1I–OC4I — Output Compare x Interrupt Enable Bits
If the OCxI enable bit is set when the OCxF flag bit is set, a hardware interrupt sequence is requested.

I4/O5I — Input Capture 4/Output Compare 5 Interrupt Enable Bit
When I4/O5 in PACTL is 1, I4/O5I is the input capture 4 interrupt enable bit. When I4/O5 in PACTL is 0, I4/O5I is the output compare 5 interrupt enable bit.

IC1I–IC3I — Input Capture x Interrupt Enable Bits
If the ICxI enable bit is set when the ICxF flag bit is set, a hardware interrupt sequence is requested.

NOTE

Bits in TMSK1 correspond bit for bit with flag bits in TFLG1. Bits in TMSK1 enable the corresponding interrupt sources.

9.4.8 Timer Interrupt Flag 1 Register

Bits in this register indicate when timer system events have occurred. Coupled with the bits of TMSK1, the bits of TFLG1 allow the timer subsystem to operate in either a polled or interrupt driven system. Each bit of TFLG1 corresponds to a bit in TMSK1 in the same position.

Address: $1023

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OC1F	OC2F	OC3F	OC4F	I4/O5F	IC1F	IC2F	IC3F
Reset:	0	0	0	0	0	0	0	0

Figure 9-18. Timer Interrupt Flag 1 Register (TFLG1)

Clear flags by writing a 1 to the corresponding bit position(s).

OC1F–OC4F — Output Compare x Flag
Set each time the counter matches output compare x value

I4/O5F — Input Capture 4/Output Compare 5 Flag
Set by IC4 or OC5, depending on the function enabled by I4/O5 bit in PACTL

IC1F–IC3F — Input Capture x Flag
Set each time a selected active edge is detected on the ICx input line

9.4.9 Timer Interrupt Mask 2 Register

Use this 8-bit register to enable or inhibit timer overflow and real-time interrupts. The timer prescaler control bits are included in this register.

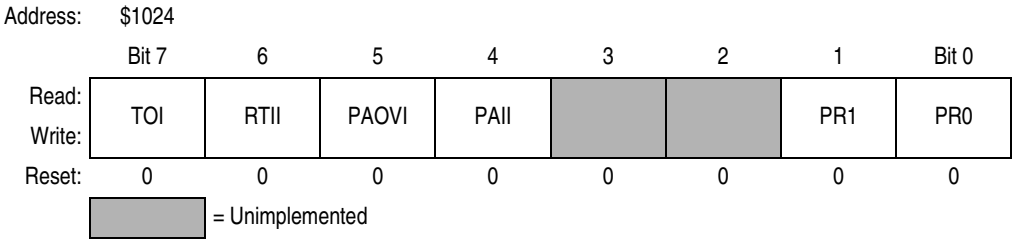


Figure 9-19. Timer Interrupt Mask 2 Register (TMSK2)

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

Refer to 9.5 Real-Time Interrupt (RTI).

PAOVI — Pulse Accumulator Overflow Interrupt Enable Bit

Refer to 9.7.3 Pulse Accumulator Status and Interrupt Bits.

PAII — Pulse Accumulator Input Edge Interrupt Enable Bit

Refer to 9.7.3 Pulse Accumulator Status and Interrupt Bits.

Bits [3:2] — Unimplemented

Always read 0

PR[1:0] — Timer Prescaler Select Bits

These bits are used to select the prescaler divide-by ratio. In normal modes, PR[1:0] can be written only once, and the write must be within 64 cycles after reset. Refer to Table 9-1 and Table 9-4 for specific timing values.

Table 9-4. Timer Prescale

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. Bits in TMSK2 enable the corresponding interrupt sources.

9.4.10 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.

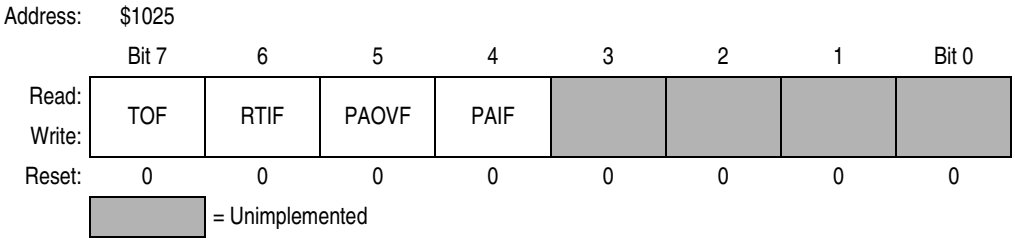


Figure 9-20. Timer Interrupt Flag 2 Register (TFLG2)

Clear flags by writing a 1 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 9.5 Real-Time Interrupt (RTI).

PAOVF — Pulse Accumulator Overflow Interrupt Flag
Refer to 9.7 Pulse Accumulator.

PAIF — Pulse Accumulator Input Edge Interrupt Flag
Refer to 9.7 Pulse Accumulator.

Bits [3:0] — Unimplemented
Always read 0

9.5 Real-Time Interrupt (RTI)

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 9-5, which shows the periodic real-time interrupt rates.

Table 9-5. RTI Rates

RTR[1:0]	E = 3 MHz	E = 2 MHz	E = 1 MHz	E = X MHz
0 0	2.731 ms	4.096 ms	8.192 ms	$(E/2^{13})$
0 1	5.461 ms	8.192 ms	16.384 ms	$(E/2^{14})$
1 0	10.923 ms	16.384 ms	32.768 ms	$(E/2^{15})$
1 1	21.845 ms	32.768 ms	65.536 ms	$(E/2^{16})$

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.

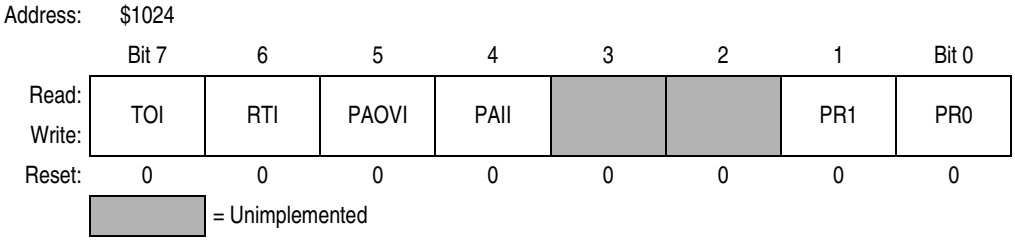


Figure 9-21. Timer Interrupt Mask 2 Register (TMSK2)

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.

9.5.2 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.

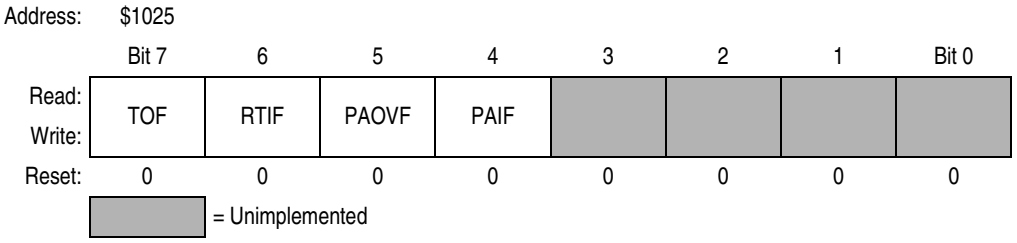


Figure 9-22. Timer Interrupt Flag 2 Register (TFLG2)

Clear flags by writing a 1 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 1 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 9.7 Pulse Accumulator.

PAIF — Pulse Accumulator Input Edge Interrupt Flag
Refer to 9.7 Pulse Accumulator.

Bits [3:0] — Unimplemented
Always read 0

9.5.3 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.

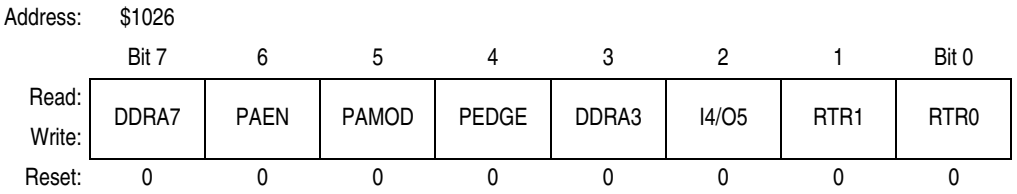


Figure 9-23. Pulse Accumulator Control Register (PACTL)

DDRA7 — Data Direction for Port A Bit 7
Refer to Chapter 6 Parallel Input/Output (I/O) Ports.

PAEN — Pulse Accumulator System Enable Bit
Refer to 9.7 Pulse Accumulator.

PAMOD — Pulse Accumulator Mode Bit
Refer to 9.7 Pulse Accumulator.

PEDGE — Pulse Accumulator Edge Control Bit

Refer to 9.7 Pulse Accumulator.

DDRA3 — Data Direction for Port A Bit 3

Refer to Chapter 6 Parallel Input/Output (I/O) Ports.

I4/O5 — Input Capture 4/Output Compare Bit

Refer to 9.7 Pulse Accumulator.

RTR[1:0] — RTI Interrupt Rate Select Bits

These two bits determine the rate at which the RTI system requests interrupts. The RTI system is driven by an E divided by 2^{13} rate clock that is compensated so it is independent of the timer prescaler. These two control bits select an additional division factor. Refer to Table 9-5.

9.6 Computer Operating Properly (COP) Watchdog Function

The clocking chain for the COP function, tapped off of the main timer divider chain, is only superficially related to the main timer system. The CR[1:0] bits in the OPTION register and the NOCOP bit in the CONFIG register determine the status of the COP function. One additional register, COPRST, is used to arm and clear the COP watchdog reset system. Refer to Chapter 5 Resets and Interrupts for a more detailed discussion of the COP function.

9.7 Pulse Accumulator

The M68HC11 Family of MCUs has an 8-bit counter that can be configured to operate either as a simple event counter or for gated time accumulation, depending on the state of the PAMOD bit in the PACTL register. Refer to the pulse accumulator block diagram, Figure 9-24. In the event counting mode, the 8-bit counter is clocked to increasing values by an external pin. The maximum clocking rate for the external event counting mode is the E clock divided by two. In gated time accumulation mode, a free-running E-clock divide-by-64 signal drives the 8-bit counter, but only while the external PAI pin is activated. Refer to Table 9-6. The pulse accumulator counter can be read or written at any time.

Table 9-6. Pulse Accumulator Timing

Crystal Frequency	E Clock	Cycle Time	E ÷ 64	PACNT Overflow
4.0 MHz	1 MHz	1000 ns	64 µs	16.384 ms
8.0 MHz	2 MHz	500 ns	32 µs	8.192 ms
12.0 MHz	3 MHz	333 ns	21.33 µs	5.461 ms

Pulse accumulator control bits are also located within two timer registers, TMSK2 and TFLG2, as described in the following paragraphs.

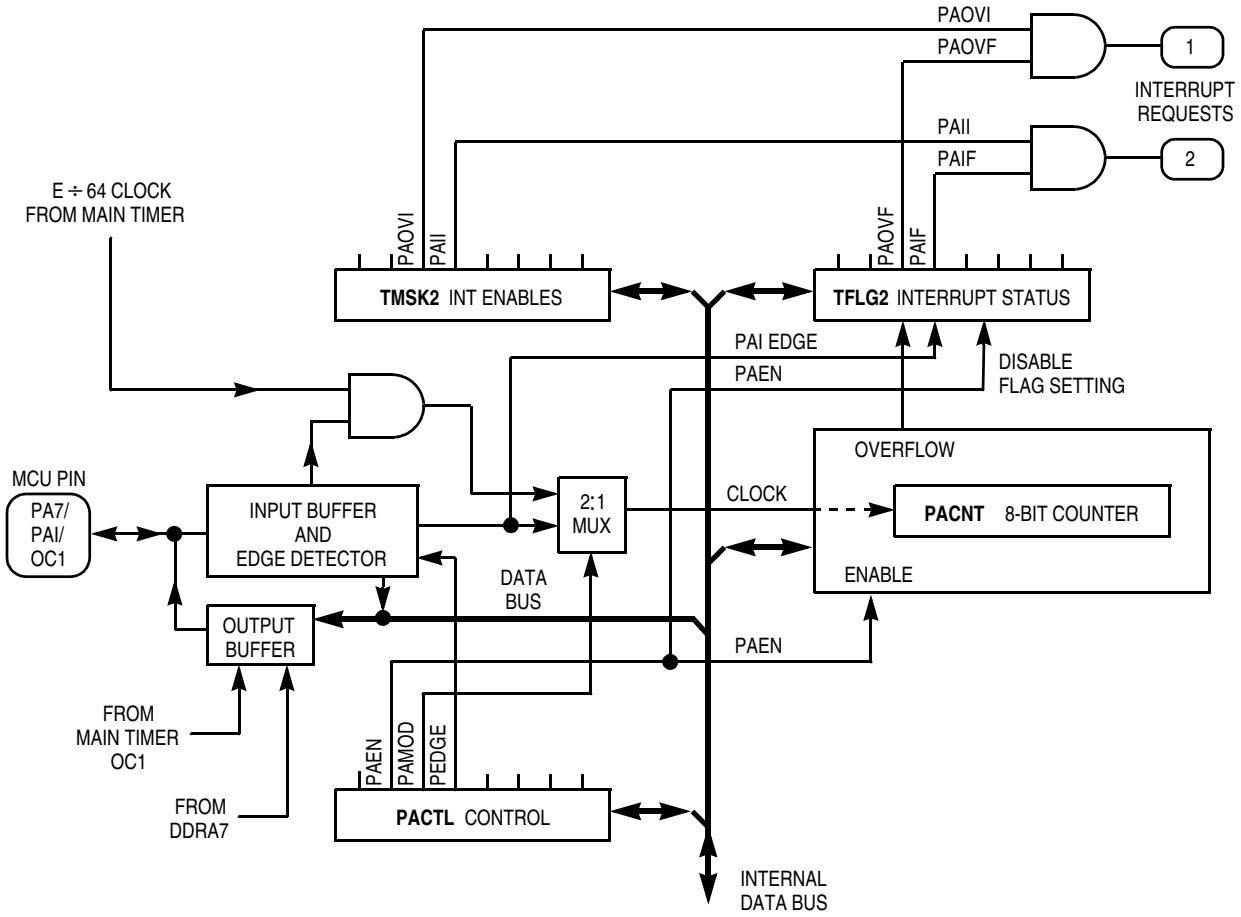


Figure 9-24. Pulse Accumulator

9.7.1 Pulse Accumulator Control Register

Four of this register's bits control an 8-bit pulse accumulator system. Another bit enables either the OC5 function or the IC4 function, while two other bits select the rate for the real-time interrupt system.

Address: $1026

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0
Reset:	0	0	0	0	0	0	0	0

Figure 9-25. Pulse Accumulator Control Register (PACTL)

DDRA7 — Data Direction for Port A Bit 7

Refer to Chapter 6 Parallel Input/Output (I/O) Ports.

PAEN — Pulse Accumulator System Enable Bit

0 = Pulse accumulator disabled
1 = Pulse accumulator enabled

PAMOD — Pulse Accumulator Mode Bit

0 = Event counter
1 = Gated time accumulation

PEDGE — Pulse Accumulator Edge Control Bit

This bit has different meanings depending on the state of the PAMOD bit, as shown in Table 9-7.

Table 9-7. Pulse Accumulator Edge Control

PAMOD	PEDGE	Action on Clock
0	0	PAI falling edge increments the counter.
0	1	PAI rising edge increments the counter.
1	0	A 0 on PAI inhibits counting.
1	1	A 1 on PAI inhibits counting.

DDRA3 — Data Direction for Port A Bit 3

Refer to Chapter 6 Parallel Input/Output (I/O) Ports.

I4/O5 — Input Capture 4/Output Compare 5 Bit

0 = Output compare 5 function enable (no IC4)
1 = Input capture 4 function enable (no OC5)

RTR[1:0] — RTI Interrupt Rate Select Bits

Refer to 9.5 Real-Time Interrupt (RTI).

9.7.2 Pulse Accumulator Count Register

This 8-bit read/write register contains the count of external input events at the PAI input or the accumulated count. The PACNT is readable even if PAI is not active in gated time accumulation mode. The counter is not affected by reset and can be read or written at any time. Counting is synchronized to the internal PH2 clock so that incrementing and reading occur during opposite half cycles.

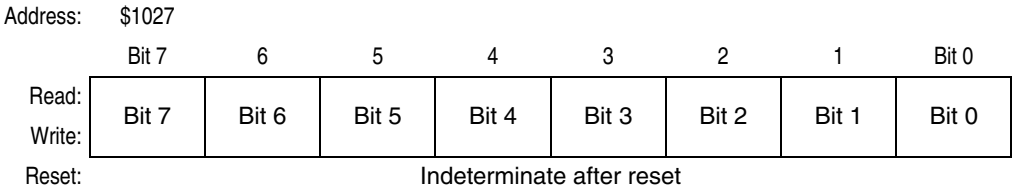


Figure 9-26. Pulse Accumulator Count Register (PACNT)

9.7.3 Pulse Accumulator Status and Interrupt Bits

The pulse accumulator control bits, PAOVI and PAII, PAOVF and PAIF, are located within timer registers TMSK2 and TFLG2.

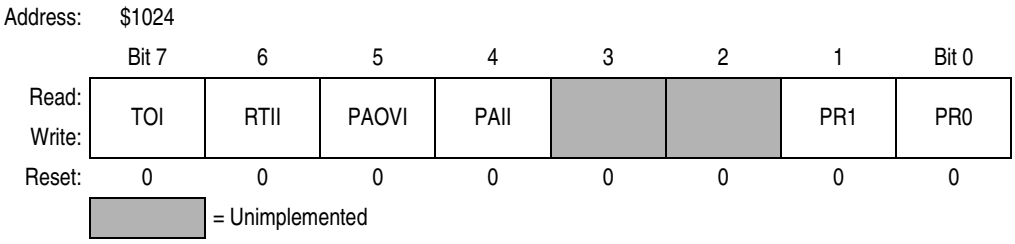


Figure 9-27. Timer Interrupt Mask 2 Register (TMSK2)

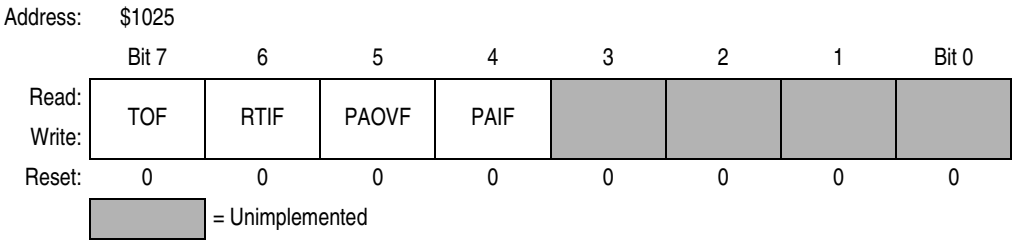


Figure 9-28. Timer Interrupt Flag 2 Register (TFLG2)

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.

Chapter 10
Electrical Characteristics

10.1 Introduction

This section contains electrical specifications for the M68HC11 E-series devices.

10.2 Maximum Ratings for Standard and Extended Voltage Devices

Maximum ratings are the extreme limits to which the microcontroller unit (MCU) can be exposed without permanently damaging it.

NOTE

This device is not guaranteed to operate properly at the maximum ratings. Refer to 10.5 DC Electrical Characteristics, 10.6 Supply Currents and Power Dissipation, 10.7 MC68L11E9/E20 DC Electrical Characteristics, and 10.8 MC68L11E9/E20 Supply Currents and Power Dissipation for guaranteed operating conditions.

Rating	Symbol	Value	Unit
Supply voltage	V_{DD}	–0.3 to +7.0	V
Input voltage	V_{In}	–0.3 to +7.0	V
Current drain per pin(1) excluding V_{DD}, V_{SS}, AV_{DD}, V_{RH}, V_{RL}, and $\overline{XIRQ}/V_{PPE}$	I_D	25	mA
Storage temperature	T_{STG}	–55 to +150	°C

1. One pin at a time, observing maximum power dissipation limits

NOTE

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that V_{In} and V_{Out} be constrained to the range $V_{SS} \le (V_{In}$ or $V_{Out}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}).

10.3 Functional Operating Range

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

10.4 Thermal Characteristics

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

1. This is an approximate value, neglecting $P_{I/O}$.
2. For most applications, $P_{I/O} \leq P_{INT}$ and can be neglected.
3. K is a constant pertaining to the device. Solve for K with a known T_A and a measured P_D (at equilibrium). Use this value of K to solve for P_D and T_J iteratively for any value of T_A.

10.5 DC Electrical Characteristics

Characteristics[1]	Symbol	Min	Max	Unit
Output voltage[2] I_{Load} = ±±10.0 µA All outputs except XTAL All outputs except XTAL, $\overline{\text{RESET}}$, and MODA	V_{OL}, V_{OH}	— V_{DD} –0.1	0.1 —	V
Output high voltage[2] I_{Load} = –0.8 mA, V_{DD} = 4.5 V All outputs except XTAL, $\overline{\text{RESET}}$, and MODA	V_{OH}	V_{DD} –0.8	—	V
Output low voltage I_{Load} = 1.6 mA All outputs except XTAL	V_{OL}	—	0.4	V
Input high voltage All inputs except $\overline{\text{RESET}}$ $\overline{\text{RESET}}$	V_{IH}	0.7 × V_{DD} 0.8 × V_{DD}	V_{DD} + 0.3 V_{DD} + 0.3	V
Input low voltage, all inputs	V_{IL}	V_{SS} –0.3	0.2 × V_{DD}	V
I/O ports, 3-state leakage V_{In} = V_{IH} or V_{IL} PA7, PA3, PC[7:0], PD[5:0], AS/STRA, MODA/$\overline{\text{LIR}}$, $\overline{\text{RESET}}$	I_{OZ}	—	±10	µA
Input leakage current[3] V_{In} = V_{DD} or V_{SS} PA[2:0], $\overline{\text{IRQ}}$, $\overline{\text{XIRQ}}$ MODB/V_{STBY} ($\overline{\text{XIRQ}}$ on EPROM-based devices)	I_{In}	— —	±1 ±10	µA
RAM standby voltage, power down	V_{SB}	4.0	V_{DD}	V
RAM standby current, power down	I_{SB}	—	10	µA
Input capacitance PA[2:0], PE[7:0], $\overline{\text{IRQ}}$, $\overline{\text{XIRQ}}$, EXTAL PA7, PA3, PC[7:0], PD[5:0], AS/STRA, MODA/$\overline{\text{LIR}}$, $\overline{\text{RESET}}$	C_{In}	— —	8 12	pF
Output load capacitance All outputs except PD[4:1] PD[4:1]	C_L	— —	90 100	pF

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted
2. V_{OH} specification for $\overline{\text{RESET}}$ and MODA is not applicable because they are open-drain pins. V_{OH} specification not applicable to ports C and D in wired-OR mode.
3. Refer to 10.13 Analog-to-Digital Converter Characteristics and 10.14 MC68L11E9/E20 Analog-to-Digital Converter Characteristics for leakage current for port E.

10.6 Supply Currents and Power Dissipation

Characteristics[1]	Symbol	Min	Max	Unit
Run maximum total supply current[2] Single-chip mode2 MHz 3 MHz Expanded multiplexed mode2 MHz 3 MHz	I_{DD}	— — — —	15 27 27 35	mA
Wait maximum total supply current[2] (all peripheral functions shut down) Single-chip mode2 MHz 3 MHz Expanded multiplexed mode2 MHz 3 MHz	W_{IDD}	— — — —	6 15 10 20	mA
Stop maximum total supply current[2] Single-chip mode, no clocks–40°C to +85°C > +85°C to +105°C > +105°C to +125°C	S_{IDD}	— — —	25 50 100	µA
Maximum power dissipation Single-chip mode2 MHz 3 MHz Expanded multiplexed mode2 MHz 3 MHz	P_D	— — — —	85 150 150 195	mW

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted
2. EXTAL is driven with a square wave, and
   t_{CYC}= 500 ns for 2 MHz rating
   t_{CYC}= 333 ns for 3 MHz rating
   $V_{IL} \leq 0.2$ V
   $V_{IH} \geq V_{DD} - 0.2$ V
   no dc loads

10.7 MC68L11E9/E20 DC Electrical Characteristics

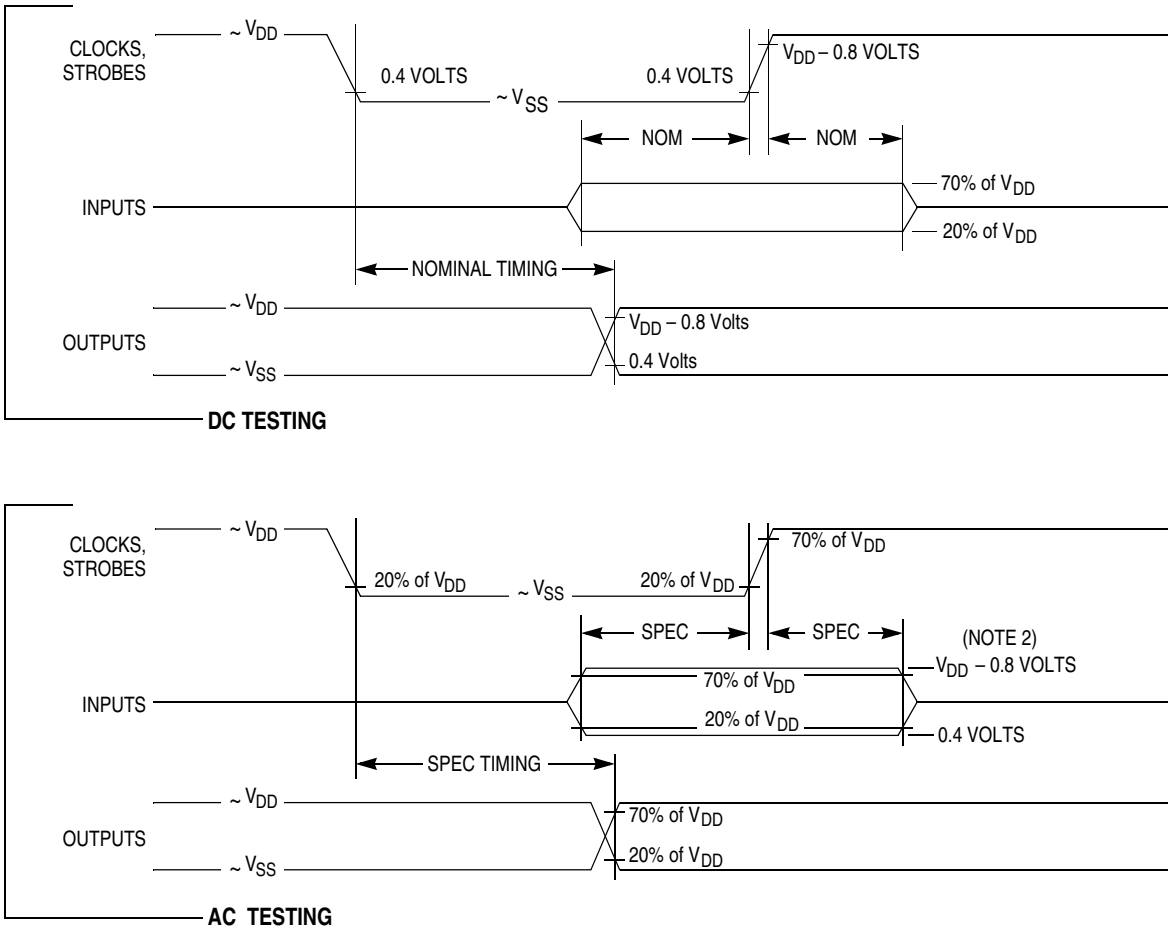
Characteristics[1]	Symbol	Min	Max	Unit
Output voltage[2] I_{Load} = ±±10.0 µA All outputs except XTAL All outputs except XTAL, $\overline{RESET}$, and MODA	V_{OL}, V_{OH}	— V_{DD} –0.1	0.1 —	V
Output high voltage[2] I_{Load} = –0.5 mA, V_{DD} = 3.0 V I_{Load} = –0.8 mA, V_{DD} = 4.5 V All outputs except XTAL, $\overline{RESET}$, and MODA	V_{OH}	V_{DD} –0.8	—	V
Output low voltage I_{Load} = 1.6 mA, V_{DD} = 5.0 V I_{Load} = 1.0 mA, V_{DD} = 3.0 V All outputs except XTAL	V_{OL}	—	0.4	V
Input high voltage All inputs except $\overline{RESET}$ $\overline{RESET}$	V_{IH}	0.7 × V_{DD} 0.8 × V_{DD}	V_{DD} + 0.3 V_{DD} + 0.3	V
Input low voltage, all inputs	V_{IL}	V_{SS} –0.3	0.2 × V_{DD}	V
I/O ports, 3-state leakage V_{In} = V_{IH} or V_{IL} PA7, PA3, PC[7:0], PD[5:0], AS/STRA, MODA/$\overline{LIR}$, $\overline{RESET}$	I_{OZ}	—	±10	µA
Input leakage current[3] V_{In} = V_{DD} or V_{SS} PA[2:0], $\overline{IRQ}$, $\overline{XIRQ}$ MODB/V_{STBY} ($\overline{XIRQ}$ on EPROM-based devices)	I_{In}	— —	±1 ±10	µA
RAM standby voltage, power down	V_{SB}	2.0	V_{DD}	V
RAM standby current, power down	I_{SB}	—	10	µA
Input capacitance PA[2:0], PE[7:0], $\overline{IRQ}$, $\overline{XIRQ}$, EXTAL PA7, PA3, PC[7:0], PD[5:0], AS/STRA, MODA/$\overline{LIR}$, $\overline{RESET}$	I	— —	8 12	pF
Output load capacitance All outputs except PD[4:1] PD[4:1]	C_L	— —	90 100	pF

1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted
2. V_{OH} specification for $\overline{RESET}$ and MODA is not applicable because they are open-drain pins. V_{OH} specification not applicable to ports C and D in wired-OR mode.
3. Refer to 10.13 Analog-to-Digital Converter Characteristics and 10.14 MC68L11E9/E20 Analog-to-Digital Converter Characteristics for leakage current for port E.

10.8 MC68L11E9/E20 Supply Currents and Power Dissipation

Characteristic[1]	Symbol	1 MHz	2 MHz	Unit
Run maximum total supply current[2] Single-chip mode V_{DD} = 5.5 V V_{DD} = 3.0 V Expanded multiplexed mode V_{DD} = 5.5 V V_{DD} = 5.5 V	I_{DD}	 8 4 14 7	 15 8 27 14	mA
Wait maximum total supply current[2] (all peripheral functions shut down) Single-chip mode V_{DD} = 5.5 V V_{DD} = 3.0 V Expanded multiplexed mode V_{DD} = 5.5 V V_{DD} = 3.0 V	W_{IDD}	 3 1.5 5 2.5	 6 3 10 5	mA
Stop maximum total supply current[2] Single-chip mode, no clocks V_{DD} = 5.5 V V_{DD} = 3.0 V	S_{IDD}	 50 25	 50 25	µA
Maximum power dissipation Single-chip mode 2 MHz 3 MHz Expanded multiplexed mode 2 MHz 3 MHz	P_D	 44 12 77 21	 85 24 150 42	mW

1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted
2. EXTAL is driven with a square wave, and
   t_{CYC}= 500 ns for 2 MHz rating
   t_{CYC}= 333 ns for 3 MHz rating
   $V_{IL} \leq 0.2$ V
   $V_{IH} \geq V_{DD} - 0.2$ V
   no dc loads



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

Figure 10-1. Test Methods

10.9 Control Timing

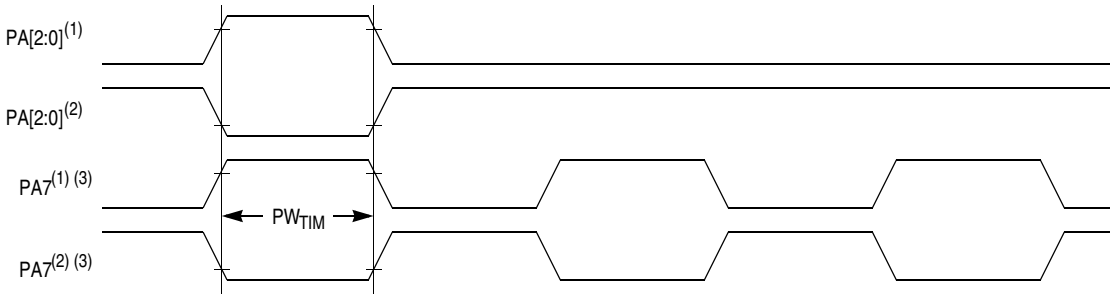
Characteristic[1] [2]	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of operation	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	333	—	ns
Crystal frequency	f_{XTAL}	—	4.0	—	8.0	—	12.0	MHz
External oscillator frequency	$4 f_o$	dc	4.0	dc	8.0	dc	12.0	MHz
Processor control setup time t_{PCSU} = 1/4 t_{CYC}+ 50 ns	t_{PCSU}	300	—	175	—	133	—	ns
Reset input pulse width To guarantee external reset vector Minimum input time (can be pre-empted by internal reset)	PW_{RSTL}	8 1	— —	8 1	— —	8 1	— —	t_{CYC}
Mode programming setup time	t_{MPS}	2	—	2	—	2	—	t_{CYC}
Mode programming hold time	t_{MPH}	10	—	10	—	10	—	ns
Interrupt pulse width, $\overline{IRQ}$ edge-sensitive mode PW_{IRQ} = t_{CYC} + 20 ns	PW_{IRQ}	1020	—	520	—	353	—	ns
Wait recovery startup time	t_{WRS}	—	4	—	4	—	4	t_{CYC}
Timer pulse width input capture pulse accumulator input PW_{TIM} = t_{CYC} + 20 ns	PW_{TIM}	1020	—	520	—	353	—	ns

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. $\overline{RESET}$ is recognized during the first clock cycle it is held low. Internal circuitry then drives the pin low for four clock cycles, releases the pin, and samples the pin level two cycles later to determine the source of the interrupt. Refer to Chapter 5 Resets and Interrupts for further detail.

10.10 MC68L11E9/E20 Control Timing

Characteristic(1) (2)	Symbol	1.0 MHz		2.0 MHz		Unit
		Min	Max	Min	Max	
Frequency of operation	f_o	dc	1.0	dc	2.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	ns
Crystal frequency	f_{XTAL}	—	4.0	—	8.0	MHz
External oscillator frequency	4 f_o	dc	4.0	dc	8.0	MHz
Processor control setup time t_{PCSU} = 1/4 t_{CYC}+ 75 ns	t_{PCSU}	325	—	200	—	ns
Reset input pulse width To guarantee external reset vector Minimum input time (can be pre-empted by internal reset)	PW_{RSTL}	8 1	— —	8 1	— —	t_{CYC}
Mode programming setup time	t_{MPS}	2	—	2	—	t_{CYC}
Mode programming hold time	t_{MPH}	10	—	10	—	ns
Interrupt pulse width, $\overline{IRQ}$ edge-sensitive mode PW_{IRQ} = t_{CYC} + 20 ns	PW_{IRQ}	1020	—	520	—	ns
Wait recovery startup time	t_{WRS}	—	4	—	4	t_{CYC}
Timer pulse width input capture pulse accumulator input PW_{TIM} = t_{CYC} + 20 ns	PW_{TIM}	1020	—	520	—	ns

1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, T_A = T_L to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. $\overline{RESET}$ is recognized during the first clock cycle it is held low. Internal circuitry then drives the pin low for four clock cycles, releases the pin, and samples the pin level two cycles later to determine the source of the interrupt. Refer to Chapter 5 Resets and Interrupts for further detail.



Notes:
1. Rising edge sensitive input
2. Falling edge sensitive input
3. Maximum pulse accumulator clocking rate is E-clock frequency divided by 2.

Figure 10-2. Timer Inputs

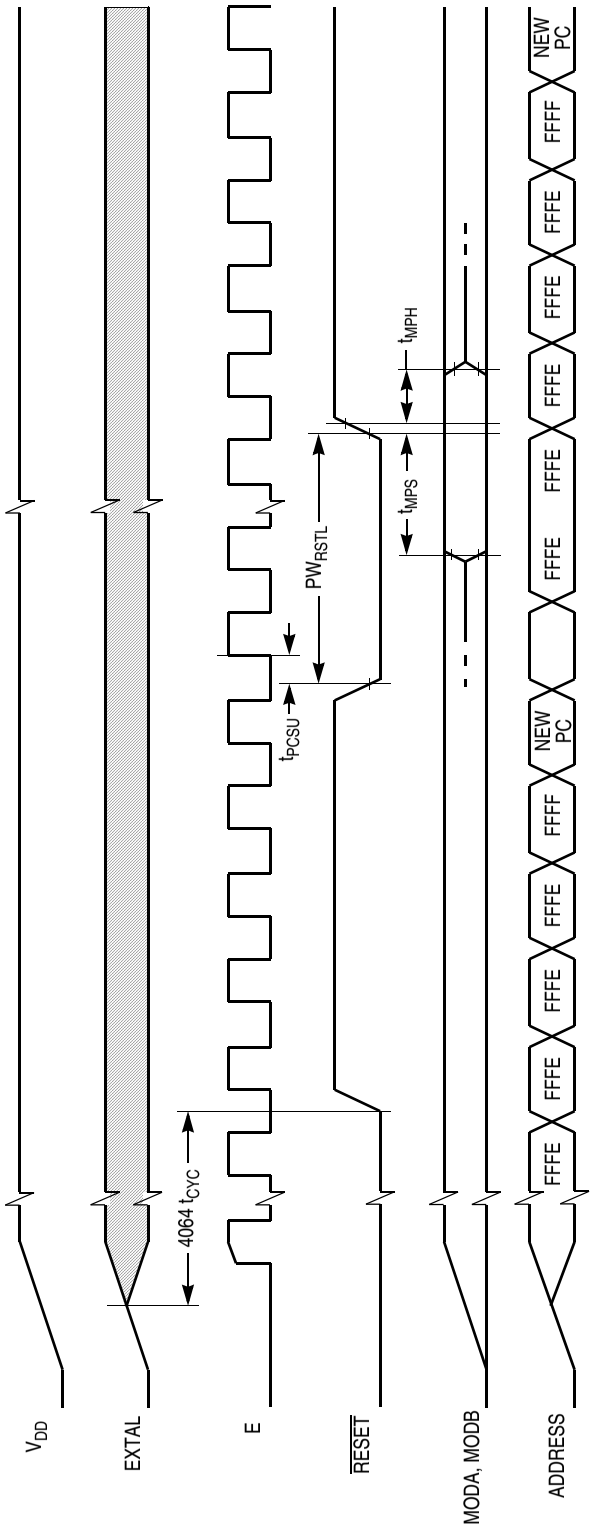
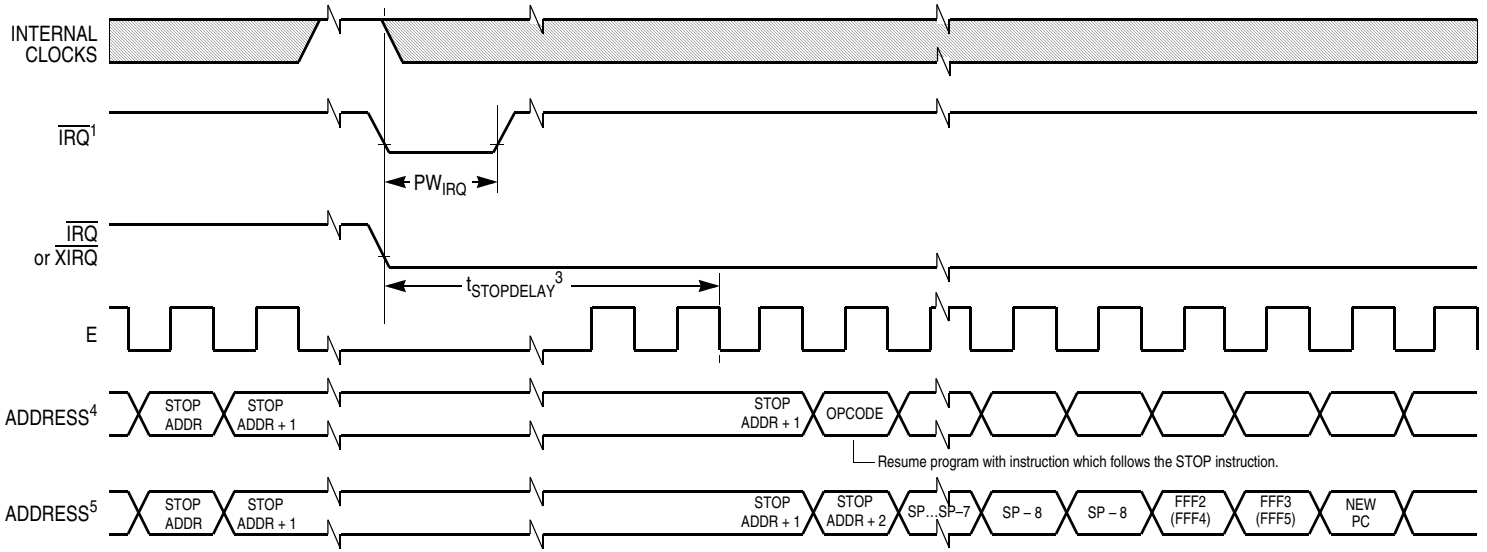
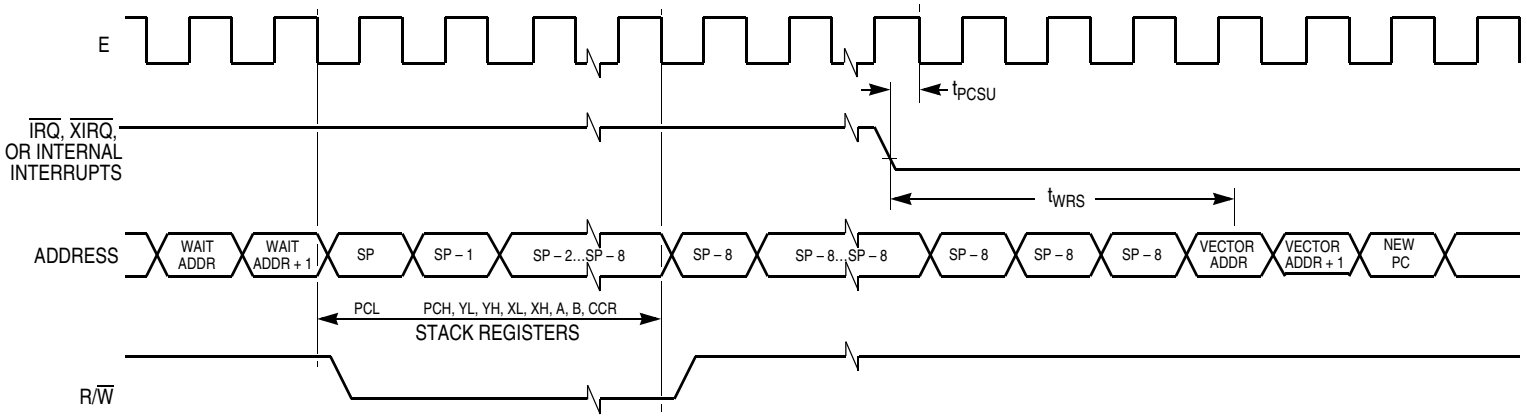


Figure 10-3. POR External Reset Timing Diagram



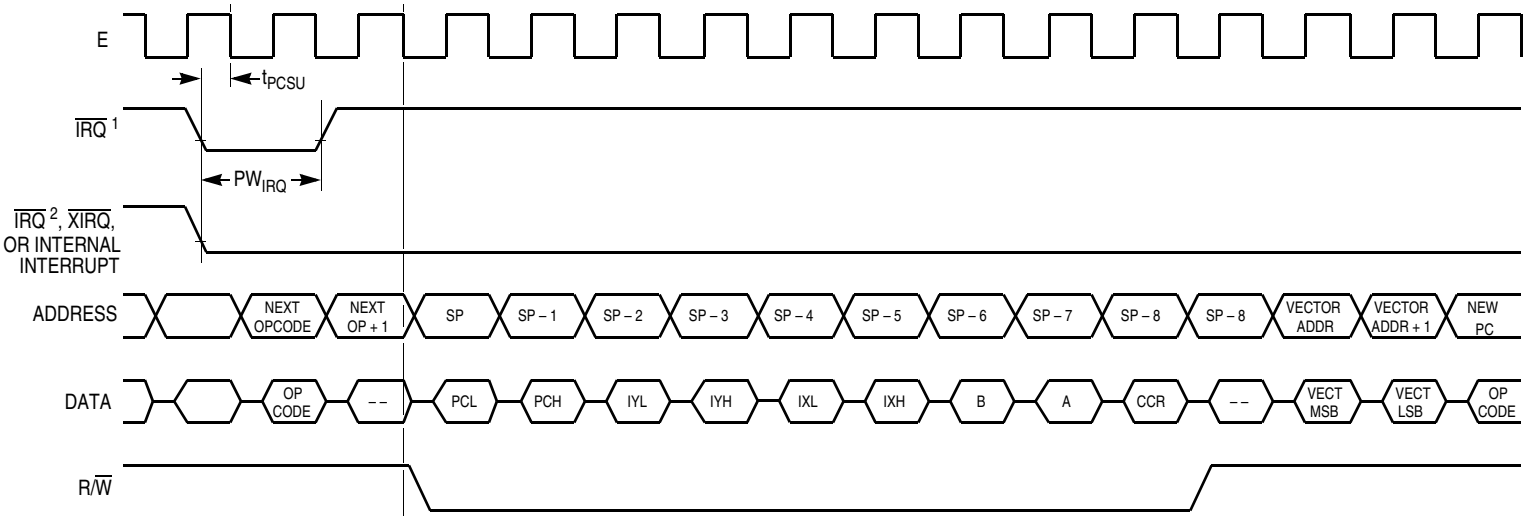
Notes:
1. Edge Sensitive $\overline{IRQ}$ pin (IRQE bit = 1)
2. Level sensitive $\overline{IRQ}$ pin (IRQE bit = 0)
3. $t_{STOPDELAY}$ = 4064 t_{CYC} if DLY bit = 1 or 4 t_{CYC} if DLY = 0.
4. $\overline{XIRQ}$ with X bit in CCR = 1.
5. $\overline{IRQ}$ or ($\overline{XIRQ}$ with X bit in CCR = 0).

Figure 10-4. STOP Recovery Timing Diagram



Note: RESET also causes recovery from WAIT.

Figure 10-5. WAIT Recovery from Interrupt Timing Diagram



Notes:
1. Edge sensitive $\overline{IRQ}$ pin (IRQE bit = 1)
2. Level sensitive $\overline{IRQ}$ pin (IRQE bit = 0)

Figure 10-6. Interrupt Timing Diagram

10.11 Peripheral Port Timing

Characteristic[1] [2]	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of operation E-clock frequency	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	333	—	ns
Peripheral data setup time MCU read of ports A, C, D, and E	t_{PDSU}	100	—	100	—	100	—	ns
Peripheral data hold time MCU read of ports A, C, D, and E	t_{PDH}	50	—	50	—	50	—	ns
Delay time, peripheral data write t_{PWD} = 1/4 t_{CYC}+ 100 ns MCU writes to port A MCU writes to ports B, C, and D	t_{PWD}	 — —	 200 350	 — —	 200 225	 — —	 200 183	ns
Port C input data setup time	t_{IS}	60	—	60	—	60	—	ns
Port C input data hold time	t_{IH}	100	—	100	—	100	—	ns
Delay time, E fall to STRB t_{DEB} = 1/4 t_{CYC}+ 100 ns	t_{DEB}	—	350	—	225	—	183	ns
Setup time, STRA asserted to E fall[3]	t_{AES}	0	—	0	—	0	—	ns
Delay time, STRA asserted to port C data output valid	t_{PCD}	—	100	—	100	—	100	ns
Hold time, STRA negated to port C data	t_{PCH}	10	—	10	—	10	—	ns
3-state hold time	t_{PCZ}	—	150	—	150	—	150	ns

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Ports C and D timing is valid for active drive. (CWOM and DWOM bits are not set in PIOC and SPCR registers, respectively.)
3. If this setup time is met, STRB acknowledges in the next cycle. If it is not met, the response may be delayed one more cycle.

10.12 MC68L11E9/E20 Peripheral Port Timing

Characteristic[1] [2]	Symbol	1.0 MHz		2.0 MHz		Unit
		Min	Max	Min	Max	
Frequency of operation E-clock frequency	f_o	dc	1.0	dc	2.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	ns
Peripheral data setup time MCU read of ports A, C, D, and E	t_{PDSU}	100	—	100	—	ns
Peripheral data hold time MCU read of ports A, C, D, and E	t_{PDH}	50	—	50	—	ns
Delay time, peripheral data write t_{PWD} = 1/4 t_{CYC}+ 150 ns MCU writes to port A MCU writes to ports B, C, and D	t_{PWD}	 — —	 250 400	 — —	 250 275	ns
Port C input data setup time	t_{IS}	60	—	60	—	ns
Port C input data hold time	t_{IH}	100	—	100	—	ns
Delay time, E fall to STRB t_{DEB} = 1/4 t_{CYC}+ 150 ns	t_{DEB}	—	400	—	275	ns
Setup time, STRA asserted to E fall[3]	t_{AES}	0	—	0	—	ns
Delay time, STRA asserted to port C data output valid	t_{PCD}	—	100	—	100	ns
Hold time, STRA negated to port C data	t_{PCH}	10	—	10	—	ns
3-state hold time	t_{PCZ}	—	150	—	150	ns

1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, T_A = T_L to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Ports C and D timing is valid for active drive. (CWOM and DWOM bits are not set in PIOC and SPCR registers, respectively.)
3. If this setup time is met, STRB acknowledges in the next cycle. If it is not met, the response may be delayed one more cycle.

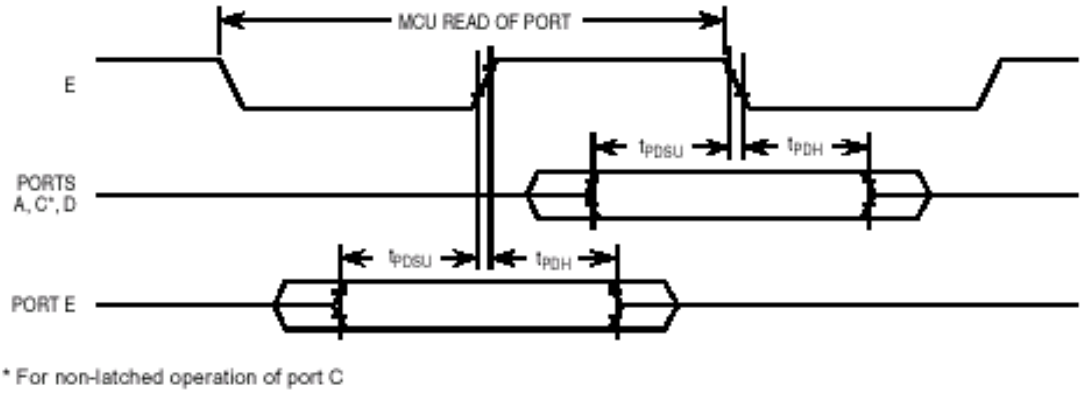


Figure 10-7. Port Read Timing Diagram

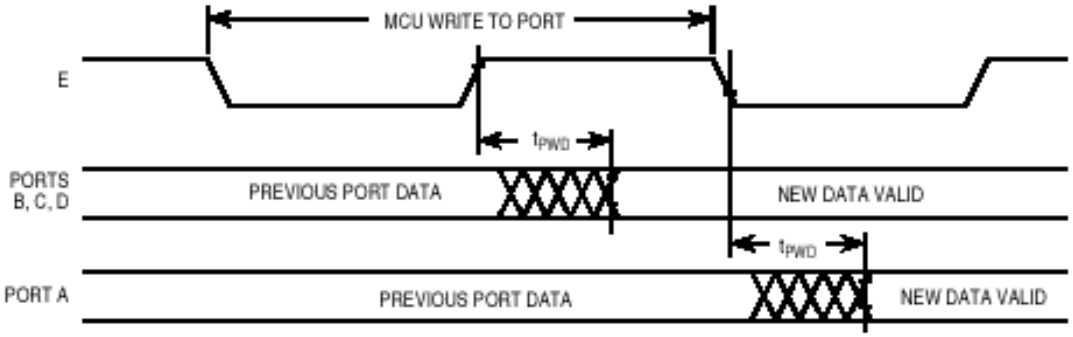


Figure 10-8. Port Write Timing Diagram

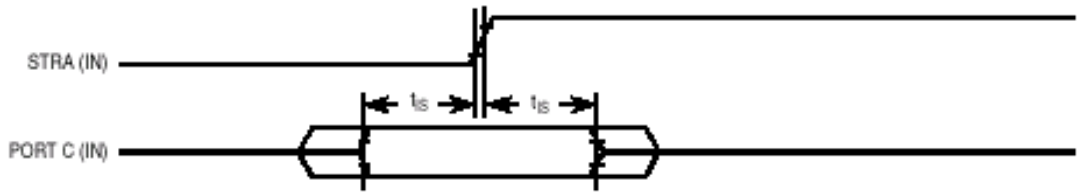


Figure 10-9. Simple Input Strobe Timing Diagram

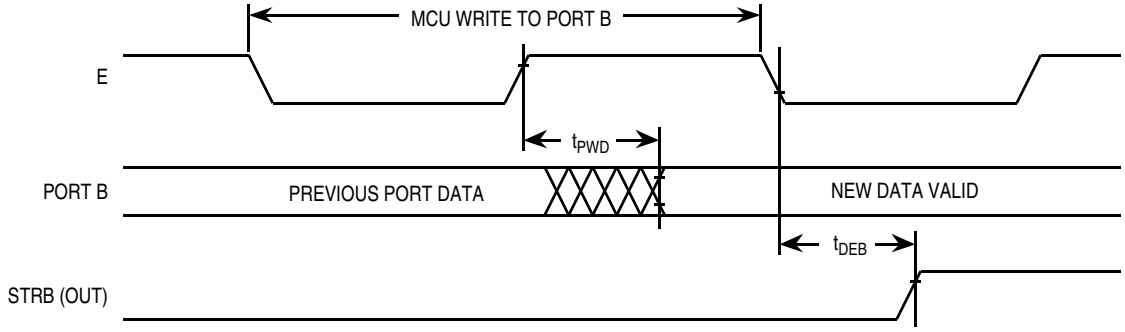
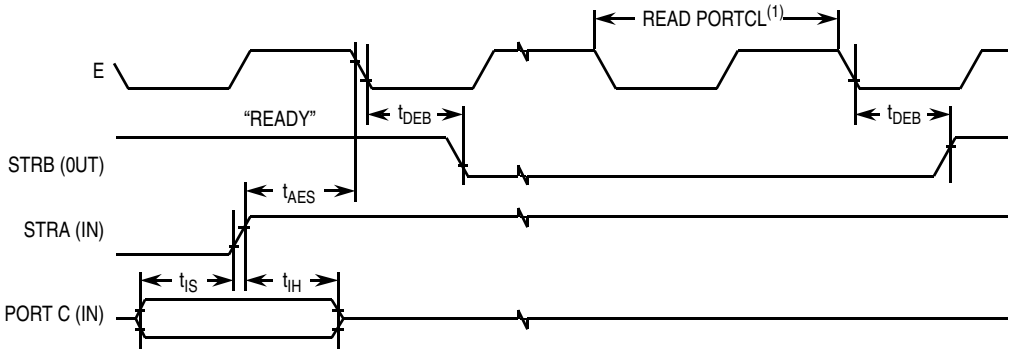
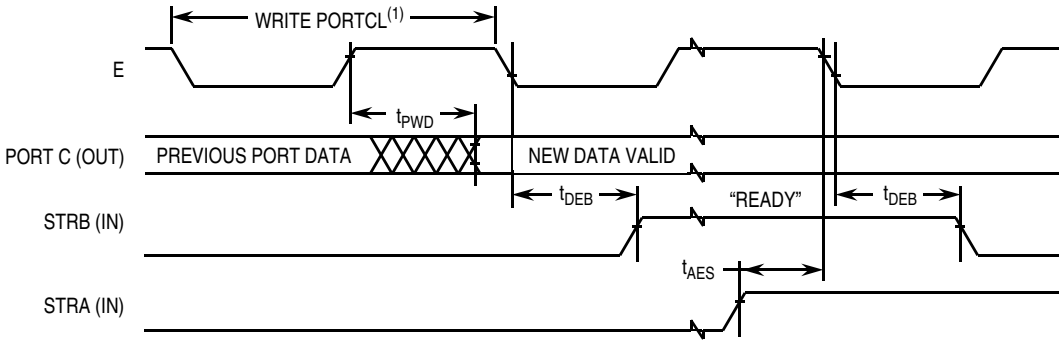


Figure 10-10. Simple Output Strobe Timing Diagram



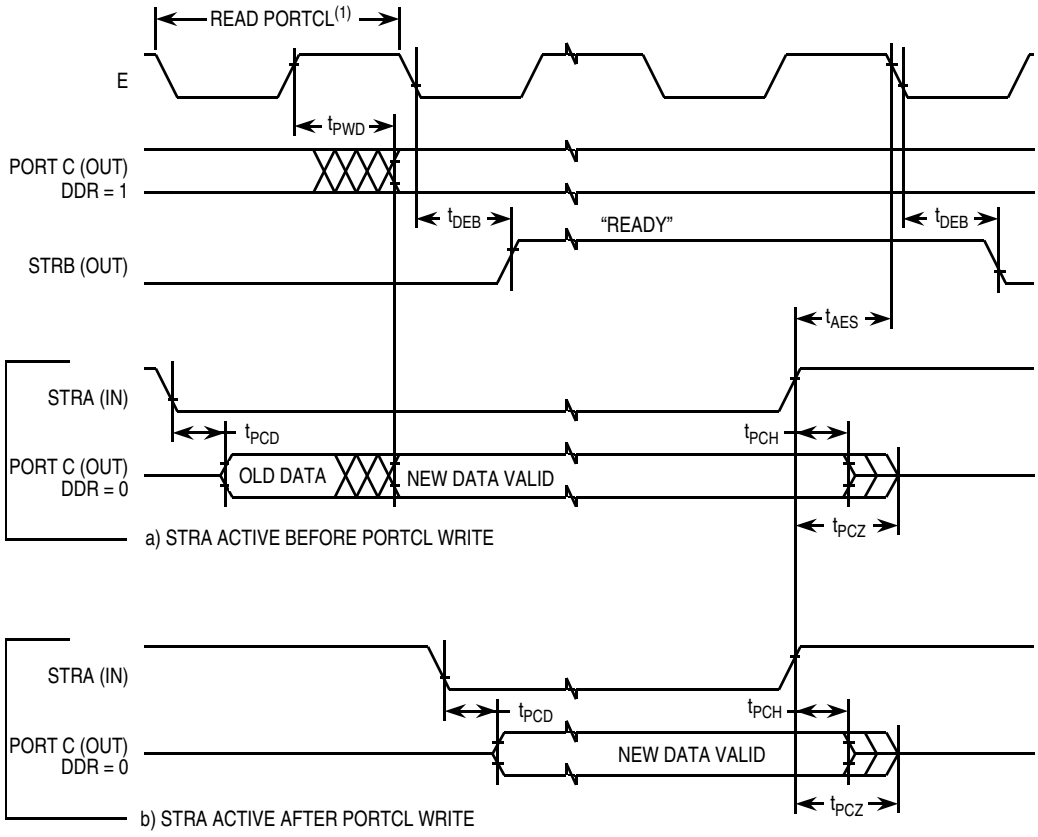
Notes:
1. After reading PIOC with STAF set
2. Figure shows rising edge STRA (EGA = 1) and high true STRB (INVB = 1).

Figure 10-11. Port C Input Handshake Timing Diagram



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

Figure 10-12. Port C Output Handshake 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 10-13. 3-State Variation of Output Handshake Timing Diagram (STRA Enables Output Buffer)

10.13 Analog-to-Digital Converter Characteristics

Characteristic[1]	Parameter[2]	Min	Absolute	2.0 MHz Max	3.0 MHz Max	Unit
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_{RL}	—	V_{RH}	V_{RH}	V
V_{RH}	Maximum analog reference voltage[3]	V_{RL}	—	V_{DD} +0.1	V_{DD} +0.1	V
V_{RL}	Minimum analog reference voltage[2]	V_{SS} –0.1	—	V_{RH}	V_{RH}	V
ΔV_R	Minimum difference between V_{RH} and V_{RL}[2]	3	—	—	—	V
Conversion time	Total time to perform a single A/D conversion: E clock Internal RC oscillator	— —	32 —	— t_{CYC}+32	— t_{CYC}+32	t_{CYC} µs
Monotonicity	Conversion result never decreases with an increase in input voltage; has no missing codes	—	Guaranteed	—	—	—
Zero input reading	Conversion result when $V_{In} = V_{RL}$	00	—	—	—	Hex
Full scale reading	Conversion result when $V_{In} = V_{RH}$	—	—	FF	FF	Hex
Sample acquisition time	Analog input acquisition sampling time: E clock Internal RC oscillator	— —	12 —	— 12	— 12	t_{CYC} µs
Sample/hold capacitance	Input capacitance during sample PE[7:0]	—	20 typical	—	—	pF
Input leakage	Input leakage on A/D pins PE[7:0] V_{RL}, V_{RH}	— —	— —	400 1.0	400 1.0	nA µA

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, 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_R, but accuracy is tested and guaranteed at ΔV_R = 5 V ±10%.

10.14 MC68L11E9/E20 Analog-to-Digital Converter Characteristics

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

1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, 750 kHz ≤ E ≤ 2.0 MHz, unless otherwise noted
2. Source impedances greater than 10 kΩ affect accuracy adversely because of input leakage.

10.15 Expansion Bus Timing Characteristics

Num	Characteristic[1]	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
			Min	Max	Min	Max	Min	Max	
	Frequency of operation (E-clock frequency)	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
1	Cycle time	t_{CYC}	1000	—	500	—	333	—	ns
2	Pulse width, E low[2], PW_{EL} = 1/2 t_{CYC}–23 ns	PW_{EL}	477	—	227	—	146	—	ns
3	Pulse width, E high[2], PW_{EH} = 1/2 t_{CYC}–28 ns	PW_{EH}	472	—	222	—	141	—	ns
4a	E and AS rise time	t_r	—	20	—	20	—	20	ns
4b	E and AS fall time	t_f	—	20	—	20	—	15	ns
9	Address hold time[2] [3]a, t_{AH} = 1/8 t_{CYC}–29.5 ns	t_{AH}	95.5	—	33	—	26	—	ns
12	Non-multiplexed address valid time to E rise $t_{AV} = PW_{EL} – (t_{ASD} + 80\ ns)$[2] [3]a	t_{AV}	281.5	—	94	—	54	—	ns
17	Read data setup time	t_{DSR}	30	—	30	—	30	—	ns
18	Read data hold time, max = t_{MAD}	t_{DHR}	0	145.5	0	83	0	51	ns
19	Write data delay time, t_{DDW} = 1/8 t_{CYC}+ 65.5 ns[2] [3]a	t_{DDW}	—	190.5	—	128		71	ns
21	Write data hold time, t_{DHW} = 1/8 t_{CYC}–29.5 ns[2] [3]a	t_{DHW}	95.5	—	33	—	26	—	ns
22	Multiplexed address valid time to E rise $t_{AVM} = PW_{EL} – (t_{ASD} + 90\ ns)$[2] [3]a	t_{AVM}	271.5	—	84	—	54	—	ns
24	Multiplexed address valid time to AS fall $t_{ASL} = PW_{ASH} – 70\ ns$[2]	t_{ASL}	151	—	26	—	13	—	ns
25	Multiplexed address hold time t_{AHL} = 1/8 t_{CYC}–29.5 ns[2] [3]b	t_{AHL}	95.5	—	33	—	31	—	ns
26	Delay time, E to AS rise, t_{ASD} = 1/8 t_{CYC}–9.5 ns[2] [3]a	t_{ASD}	115.5	—	53	—	31	—	ns
27	Pulse width, AS high, PW_{ASH} = 1/4 t_{CYC}–29 ns[2]	PW_{ASH}	221	—	96	—	63	—	ns
28	Delay time, AS to E rise, t_{ASED} = 1/8 t_{CYC}–9.5 ns[2] [3]b	t_{ASED}	115.5	—	53	—	31	—	ns
29	MPU address access time[3]a $t_{ACCA} = t_{CYC} – (PW_{EL} – t_{AVM}) – t_{DSR} – t_f$	t_{ACCA}	744.5	—	307	—	196	—	ns
35	MPU access time, $t_{ACCE} = PW_{EH} – t_{DSR}$	t_{ACCE}	—	442	—	192		111	ns
36	Multiplexed address delay (Previous cycle MPU read) $t_{MAD} = t_{ASD} + 30\ ns$[2] [3]a	t_{MAD}	145.5	—	83	—	51	—	ns

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Formula only for dc to 2 MHz
3. Input clocks with duty cycles other than 50% affect bus performance. Timing parameters affected by input clock duty cycle are identified by (a) and (b). To recalculate the approximate bus timing values, substitute the following expressions in place of 1/8 t_{CYC}in the above formulas, where applicable:
   (a) (1–dc) × 1/4 t_{CYC}
   (b) dc × 1/4 t_{CYC}

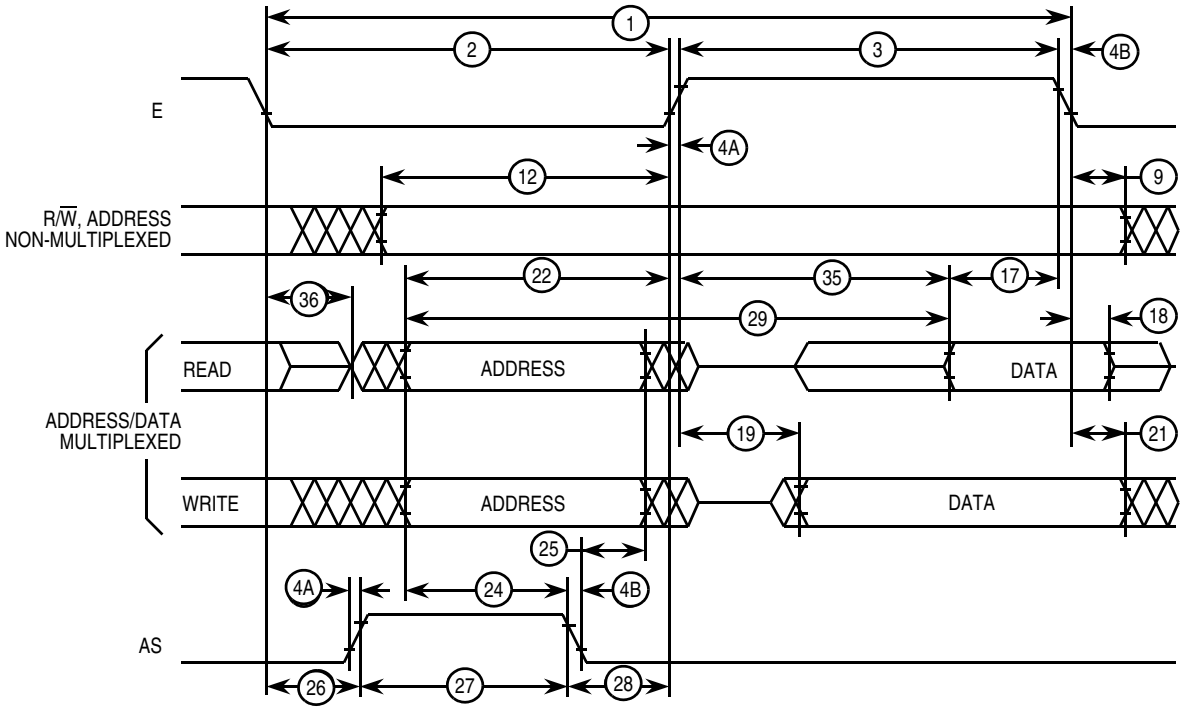
Where:
dc is the decimal value of duty cycle percentage (high time)

10.16 MC68L11E9/E20 Expansion Bus Timing Characteristics

Num	Characteristic[1]	Symbol	1.0 MHz		2.0 MHz		Unit
			Min	Max	Min	Max	
	Frequency of operation (E-clock frequency)	f_o	dc	1.0	dc	2.0	MHz
1	Cycle time	t_{CYC}	1000	—	500	—	ns
2	Pulse width, E low, PW_{EL} = 1/2 t_{CYC}–25 ns	PW_{EL}	475	—	225	—	ns
3	Pulse width, E high, PW_{EH} = 1/2 t_{CYC}–30 ns	PW_{EH}	470	—	220	—	ns
4a	E and AS rise time	t_r	—	25	—	25	ns
4b	E and AS fall time	t_f	—	25	—	25	ns
9	Address hold time[(2) (2)a], t_{AH} = 1/8 t_{CYC}–30 ns	t_{AH}	95	—	33	—	ns
12	Non-multiplexed address valid time to E rise $t_{AV} = PW_{EL} – (t_{ASD} + 80\ ns)$[(2)a]	t_{AV}	275	—	88	—	ns
17	Read data setup time	t_{DSR}	30	—	30	—	ns
18	Read data hold time , max = t_{MAD}	t_{DHR}	0	150	0	88	ns
19	Write data delay time, t_{DDW} = 1/8 t_{CYC}+ 70 ns[(2)a]	t_{DDW}	—	195	—	133	ns
21	Write data hold time, t_{DHW} = 1/8 t_{CYC}–30 ns[(2)a]	t_{DHW}	95	—	33	—	ns
22	Multiplexed address valid time to E rise $t_{AVM} = PW_{EL} – (t_{ASD} + 90\ ns)$[(2)a]	t_{AVM}	268	—	78	—	ns
24	Multiplexed address valid time to AS fall $t_{ASL} = PW_{ASH} – 70\ ns$	t_{ASL}	150	—	25	—	ns
25	Multiplexed address hold time, t_{AHL} = 1/8 t_{CYC}–30 ns[(2)b]	t_{AHL}	95	—	33	—	ns
26	Delay time, E to AS rise, t_{ASD} = 1/8 t_{CYC}–5 ns[(2)a]	t_{ASD}	120	—	58	—	ns
27	Pulse width, AS high, PW_{ASH} = 1/4 t_{CYC}–30 ns	PW_{ASH}	220	—	95	—	ns
28	Delay time, AS to E rise, t_{ASED} = 1/8 t_{CYC}–5 ns[(2)b]	t_{ASED}	120	—	58	—	ns
29	MPU address access time[(3)a] $t_{ACCA} = t_{CYC} – (PW_{EL} – t_{AVM}) – t_{DSR} – t_f$	t_{ACCA}	735	—	298	—	ns
35	MPU access time, $t_{ACCE} = PW_{EH} – t_{DSR}$	t_{ACCE}	—	440	—	190	ns
36	Multiplexed address delay (Previous cycle MPU read) $t_{MAD} = t_{ASD} + 30\ ns$[(2)a]	t_{MAD}	150	—	88	—	ns

1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Input clocks with duty cycles other than 50% affect bus performance. Timing parameters affected by input clock duty cycle are identified by (a) and (b). To recalculate the approximate bus timing values, substitute the following expressions in place of 1/8 t_{CYC}in the above formulas, where applicable:
   (a) (1–dc) × 1/4 t_{CYC}
   (b) dc × 1/4 t_{CYC}

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



Note: Measurement points shown are 20% and 70% of V_{DD}.

Figure 10-14. Multiplexed Expansion Bus Timing Diagram

10.17 Serial Peripheral Interface Timing Characteristics

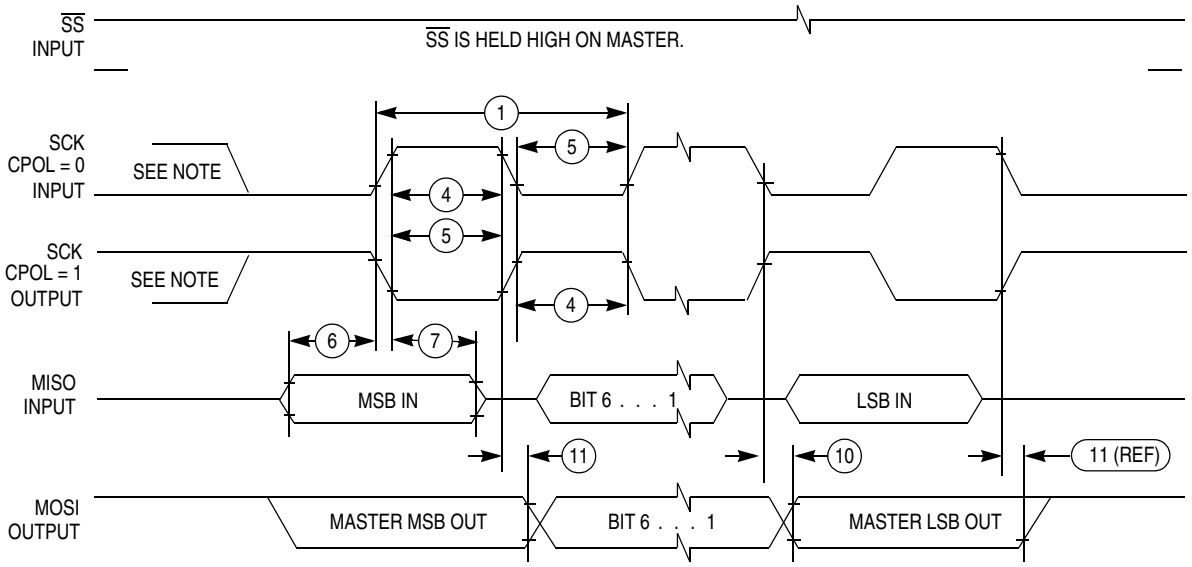
Num	Characteristic[1]	Symbol	E9		E20		Unit
			Min	Max	Min	Max	
	Frequency of operation E clock	f_o	dc	3.0	dc	3.0	MHz
	E-clock period	t_{CYC}	333	—	333	—	ns
	Operating frequency Master Slave	$f_{op(m)}$ $f_{op(s)}$	$f_o/32$ dc	$f_o/2$ f_o	$f_o/128$ dc	$f_o/2$ f_o	MHz
1	Cycle time Master Slave	$t_{CYC(m)}$ $t_{CYC(s)}$	2 1	32 —	2 1	128 —	t_{CYC}
2	Enable lead time[2] Slave	$t_{lead(s)}$	1	—	1	—	t_{CYC}
3	Enable lag time[2] Slave	$t_{lag(s)}$	1	—	1	—	t_{CYC}
4	Clock (SCK) high time Master Slave	$t_{w(SCKH)m}$ $t_{w(SCKH)s}$	$t_{CYC}–25$ 1/2 $t_{CYC}–25$	16 t_{CYC} —	$t_{CYC}–25$ 1/2 $t_{CYC}–25$	64 t_{CYC} —	ns
5	Clock (SCK) low time Master Slave	$t_{w(SCKL)m}$ $t_{w(SCKL)s}$	$t_{CYC}–25$ 1/2 $t_{CYC}–25$	16 t_{CYC} —	$t_{CYC}–25$ 1/2 $t_{CYC}–25$	64 t_{CYC} —	ns
6	Data setup time (inputs) Master Slave	$t_{su(m)}$ $t_{su(s)}$	30 30	— —	30 30	— —	ns
7	Data hold time (inputs) Master Slave	$t_{h(m)}$ $t_{h(s)}$	30 30	— —	30 30	— —	ns
8	Slave access time CPHA = 0 CPHA = 1	t_a	0 0	40 40	0 0	40 40	ns
9	Disable time (hold time to high-impedance state) Slave	t_{dis}	—	50	—	50	ns
10	Data valid[3] (after enable edge)	t_v	—	50	—	50	ns
11	Data hold time (outputs) (after enable edge)	t_{ho}	0	—	0	—	ns

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Time to data active from high-impedance state
3. Assumes 200 pF load on SCK, MOSI, and MISO pins

10.18 MC68L11E9/E20 Serial Peirpheral Interface Characteristics

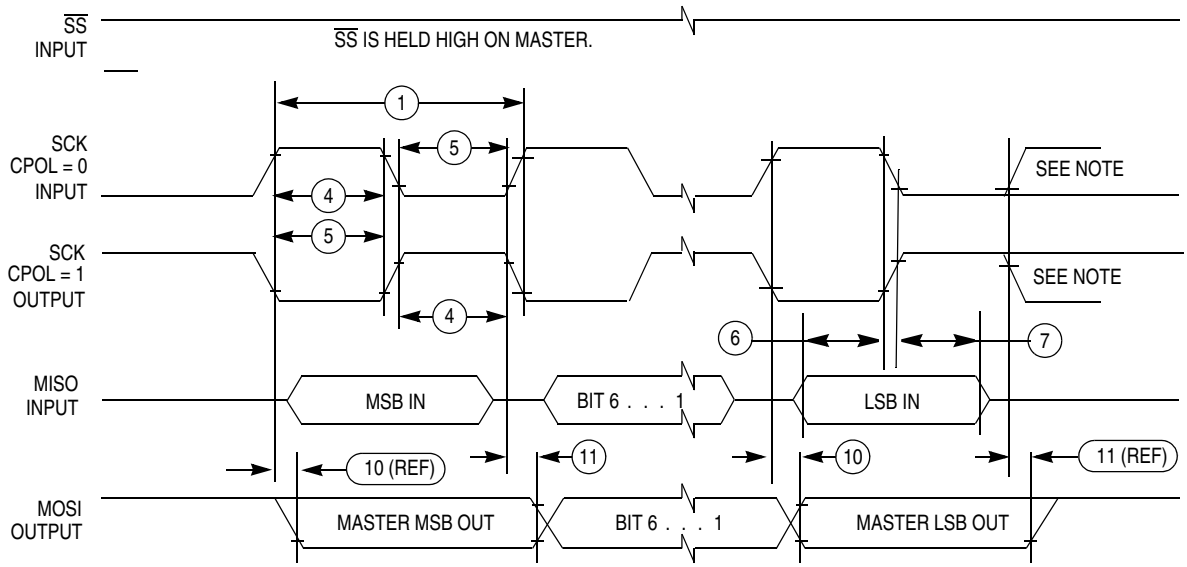
Num	Characteristic[1]	Symbol	E9		E20		Unit
			Min	Max	Min	Max	
	Frequency of operation E clock	f_o	dc	2.0	dc	2.0	MHz
	E-clock period	t_{CYC}	500	—	500	—	ns
	Operating frequency Master Slave	$f_{op(m)}$ $f_{op(s)}$	f_o/32 dc	f_o/2 f_o	f_o/128 dc	f_o/2 f_o	MHz
1	Cycle time Master Slave	$t_{CYC(m)}$ $t_{CYC(s)}$	2 1	32 —	2 1	128 —	t_{CYC}
2	Enable lead time[2] Slave	$t_{lead(s)}$	1	—	1	—	t_{CYC}
3	Enable lag time[2] Slave	$t_{lag(s)}$	1	—	1	—	t_{CYC}
4	Clock (SCK) high time Master Slave	$t_{w(SCKH)m}$ $t_{w(SCKH)s}$	t_{CYC}–30 1/2 t_{CYC}–30	16 t_{CYC} —	t_{CYC}–30 1/2 t_{CYC}–30	64 t_{CYC} —	ns
5	Clock (SCK) low time Master Slave	$t_{w(SCKL)m}$ $t_{w(SCKL)s}$	t_{CYC}–30 1/2 t_{CYC}–30	16 t_{CYC} —	t_{CYC}–30 1/2 t_{CYC}–30	64 t_{CYC} —	ns
6	Data setup time (inputs) Master Slave	$t_{su(m)}$ $t_{su(s)}$	40 40	— —	40 40	— —	ns
7	Data hold time (inputs) Master Slave	$t_{h(m)}$ $t_{h(s)}$	40 40	— —	40 40	— —	ns
8	Slave access time CPHA = 0 CPHA = 1	t_a	0 0	50 50	0 0	50 50	ns
9	Disable time (hold time to high-impedance state) Slave	t_{dis}	—	60	—	60	ns
10	Data valid[3] (after enable edge)	t_v	—	60	—	60	ns
11	Data hold time (outputs) (after enable edge)	t_{ho}	0	—	0	—	ns

1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, T_A = T_L to T_H, all timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted
2. Time to data active from high-impedance state
3. Assumes 100 pF load on SCK, MOSI, and MISO pins



Note: This first clock edge is generated internally but is not seen at the SCK pin.

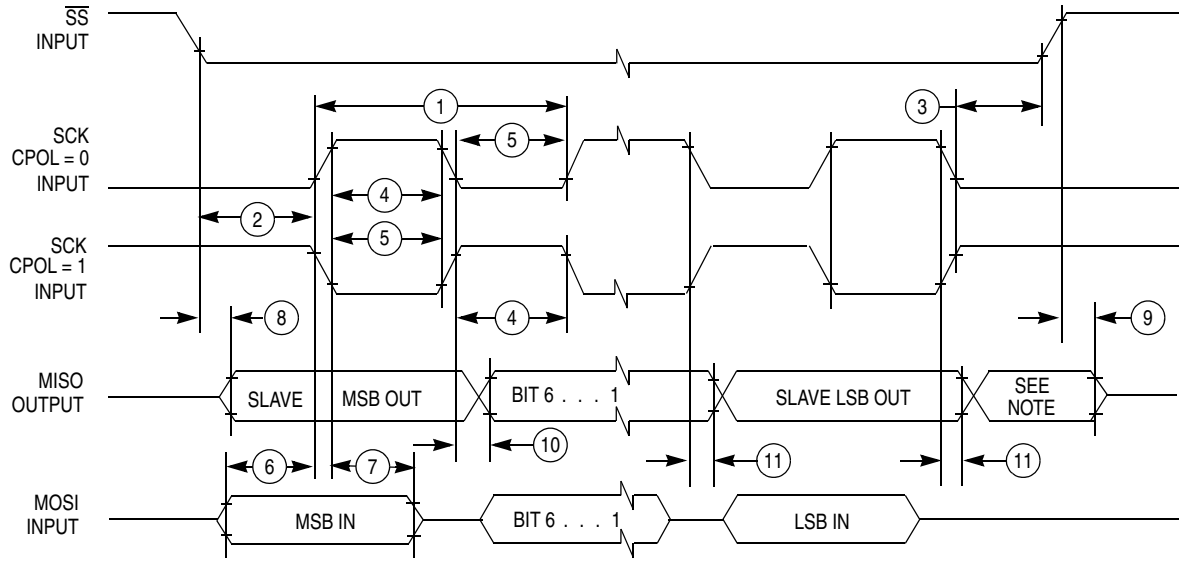
A) SPI Master Timing (CPHA = 0)



Note: This first clock edge is generated internally but is not seen at the SCK pin.

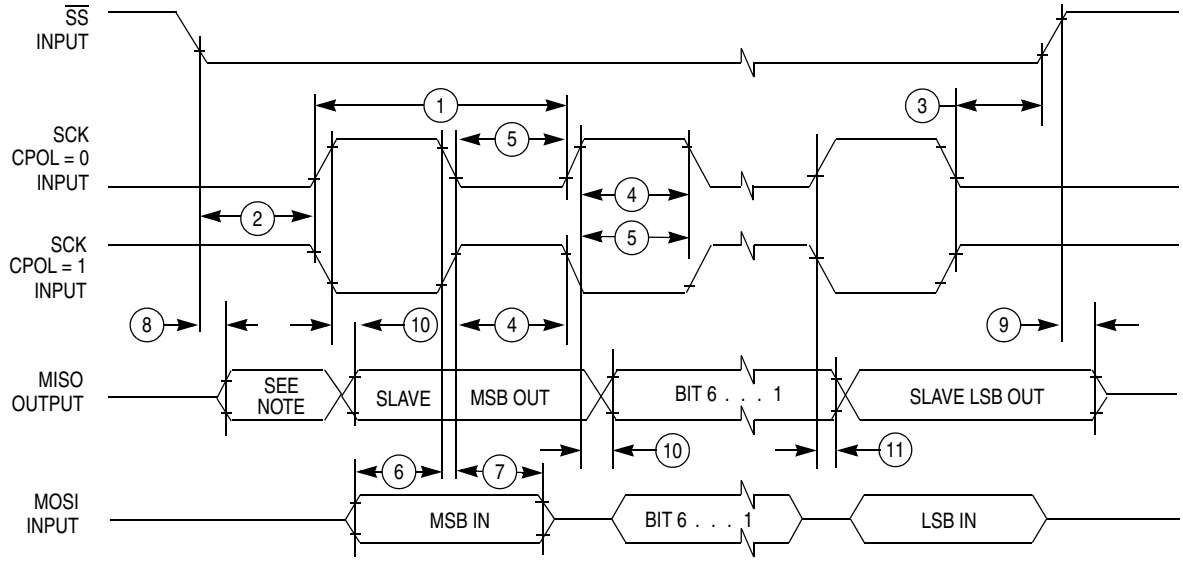
B) SPI Master Timing (CPHA = 1)

Figure 10-15. SPI Timing Diagram (Sheet 1 of 2)



Note: Not defined but normally MSB of character just received

A) SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character previously transmitted

B) SPI Slave Timing (CPHA = 1)

Figure 11-15. SPI Timing Diagram (Sheet 2 of 2)

10.19 EEPROM Characteristics

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

1. V_{DD} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H
2. The RC oscillator (RCO) must be enabled (by setting the CSEL bit in the OPTION register) for EEPROM programming and erasure when the E-clock frequency is below 1.0 MHz.

10.20 MC68L11E9/E20 EEPROM Characteristics

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

1. V_{DD} = 3.0 Vdc to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H
2. The RC oscillator (RCO) must be enabled (by setting the CSEL bit in the OPTION register) for EEPROM programming and erasure.

10.21 EPROM Characteristics

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

1. V_{DD} = 5.0 Vdc ± 10%
2. During EPROM programming of the MC68HC711E9 device, the V_{PPE} pin circuitry may latch-up and be damaged if the input current is not limited to 10 mA. For more information please refer to MC68HC711E9 8-Bit Microcontroller Unit Mask Set Errata 3 (Freescale document order number 68HC711E9MSE3.
3. Typically, a 1-kΩ series resistor is sufficient to limit the programming current for the MC68HC711E9. A 100-Ω series resistor is sufficient to limit the programming current for the MC68HC711E20.

Chapter 11
Ordering Information and Mechanical Specifications

11.1 Introduction

This section provides ordering information for the E-series devices grouped by:

- Standard devices
- Custom ROM devices
- Extended voltage devices

In addition, mechanical specifications for the following packaging options:

- 52-pin plastic-leaded chip carrier (PLCC)
- 52-pin windowed ceramic-leaded chip carrier (CLCC)
- 64-pin quad flat pack (QFP)
- 52-pin thin quad flat pack (TQFP)
- 56-pin shrink dual in-line package with .070-inch lead spacing (SDIP)
- 48-pin plastic DIP (.100-inch lead spacing), MC68HC811E2 only

11.2 Standard Device Ordering Information

Description	CONFIG	Temperature	Frequency	MC Order Number
52-pin plastic leaded chip carrier (PLCC)				
BUFFALO ROM	$0F	–40°C to +85°C	2 MHz	MC68HC11E9BCFN2
			3 MHz	MC68HC11E9BCFN3
No ROM	$0D	–40°C to +85°C	2 MHz	MC68HC11E1CFN2
			3 MHz	MC68HC11E1CFN3
		–40°C to +105°C	2 MHz	MC68HC11E1VFN2
		–40°C to +125°C	2 MHz	MC68HC11E1MFN2
No ROM, no EEPROM	$0C	–40°C to +85°C	2 MHz	MC68HC11E0CFN2
			3 MHz	MC68HC11E0CFN3
		–40°C to +105°C	2 MHz	MC68HC11E0VFN2
		–40°C to +125°C	2 MHz	MC68HC11E0MFN2

Description	CONFIG	Temperature	Frequency	MC Order Number
52-pin plastic leaded chip carrier (PLCC) (Continued)				
OTPROM	$0F	–40°C to +85°C	2 MHz	MC68HC711E9CFN2
			3 MHz	MC68HC711E9CFN3
		–40°C to +105°C	2 MHz	MC68HC711E9VFN2
		–40°C to +125°C	2 MHz	MC68HC711E9MFN2
OTPROM, enhanced security feature	$0F	–40°C to +85°C	2 MHz	MC68S711E9CFN2
20 Kbytes OTPROM	$0F	0°C to +70°C	3 MHz	MC68HC711E20FN3
		–40°C to +85°C	2 MHz	MC68HC711E20CFN2
			3 MHz	MC68HC711E20CFN3
		–40°C to +105°C	2 MHz	MC68HC711E20VFN2
		–40°C to +125°C	2 MHz	MC68HC711E20MFN2
No ROM, 2 Kbytes EEPROM	$FF	0°C to +70°C	2 MHz	MC68HC811E2FN2
		–40°C to +85°C	2 MHz	MC68HC811E2CFN2
		–40°C to +105°C	2 MHz	MC68HC811E2VFN2
		–40°C to +125°C	2 MHz	MC68HC811E2MFN2
64-pin quad flat pack (QFP)				
BUFFALO ROM	$0F	–40°C to +85°C	2 MHz	MC68HC11E9BCFU2
			3 MHz	MC68HC11E9BCFU3
No ROM	$0D	–40°C to +85°C	2 MHz	MC68HC11E1CFU2
			3 MHz	MC68HC11E1CFU3
		–40°C to +105°C	2 MHz	MC68HC11E1VFU2
No ROM, no EEPROM	$0C	–40°C to +85°C	2 MHz	MC68HC11E0CFU2
		–40°C to +105°C	2 MHz	MC68HC11E0VFU2
20 Kbytes OTPROM	$0F	0°C to +70°°C	3 MHz	MC68HC711E20FU3
		–40°C to +85°C	2 MHz	MC68HC711E20CFU2
			3 MHz	MC68HC711E20CFU3
		–40°C to +105°C	2 MHz	MC68HC711E20VFU2
		–40°C to +125°C	2 MHz	MC68HC711E20MFU2
52-pin thin quad flat pack (TQFP)				
BUFFALO ROM	$0F	–40°C to +85°C	2 MHz	MC68HC11E9BCPB2
			3 MHz	MC68HC11E9BCPB3

Description	CONFIG	Temperature	Frequency	MC Order Number
52-pin windowed ceramic leaded chip carrier (CLCC)				
EPROM	$0F	–40°C to +85°C	2 MHz	MC68HC711E9CFS2
			3 MHz	MC68HC711E9CFS3
		–40°C to +105°C	2 MHz	MC68HC711E9VFS2
		–40°C to +125°C	2 MHz	MC68HC711E9VFS2
20 Kbytes EPROM	$0F	0°C o +70°°C	3 MHz	MC68HC711E20FS3
		–40°C to +85°C	2 MHz	MC68HC711E20CFS2
			3 MHz	MC68HC711E20CFS3
		–40°C to +105°C	2 MHz	MC68HC711E20VFS2
		–40°C to +125°C	2 MHz	MC68HC711E20MFS2
48-pin dual in-line package (DIP) — MC68HC811E2 only				
No ROM, 2 Kbytes EEPROM	$FF	0°C to +70°°C	2 MHz	MC68HC811E2P2
		–40°C to +85°C	2 MHz	MC68HC811E2CP2
		–40°C to +105°C	2 MHz	MC68HC811E2VP2
		–40°C to +125°C	2 MHz	MC68HC811E2MP2
56-pin dual in-line package with 0.70-inch lead spacing (SDIP)				
BUFFALO ROM	$0F	–40°C to +85°C	2 MHz	MC68HC11E9BCB2
			3 MHz	MC68HC11E9BCB3
No ROM	$0D	–40°C to +85°C	2 MHz	MC68HC11E1CB2
			3 MHz	MC68HC11E1CB3
		–40°C to +105°C	2 MHz	MC68HC11E1VB2
		–40°C to +125°C	2 MHz	MC68HC11E1MB2
No ROM, no EEPROM	$0C	–40°C to +85°C	2 MHz	MC68HC11E0CB2
			3 MHz	MC68HC11E0CB3
		–40°C to +105°C	2 MHz	MC68HC11E0VB2
		–40°C to +125°C	2 MHz	MC68HC11E0MB2

11.3 Custom ROM Device Ordering Information

Description	Temperature	Frequency	MC Order Number
52-pin plastic leaded chip carrier (PLCC)			
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

Description	Temperature	Frequency	MC Order Number
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

64-pin quad flat pack (QFP)

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

64-pin quad flat pack (continued)

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

52-pin thin quad flat pack (10 mm x 10 mm)

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

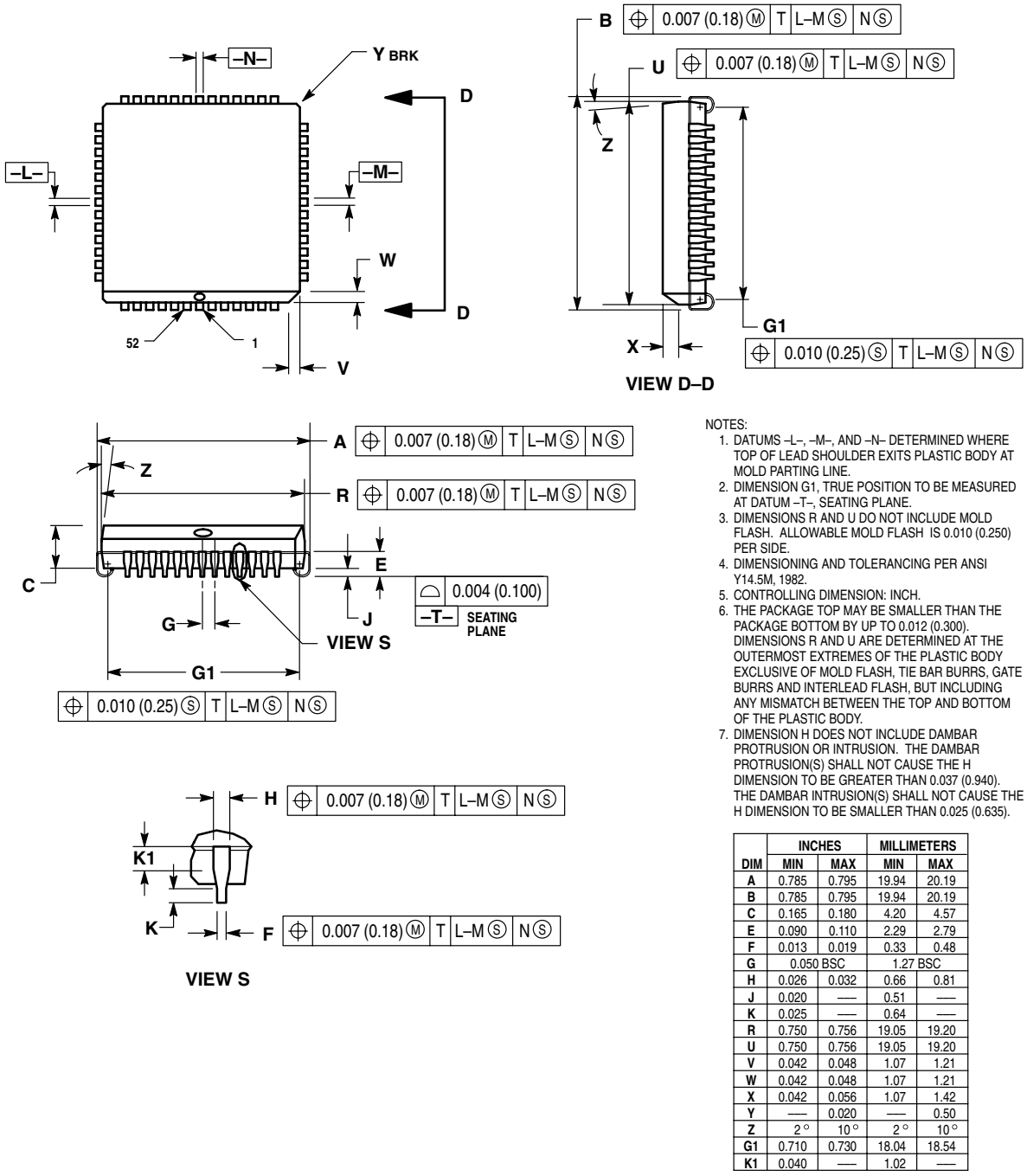
56-pin dual in-line package with 0.70-inch lead spacing (SDIP)

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

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

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

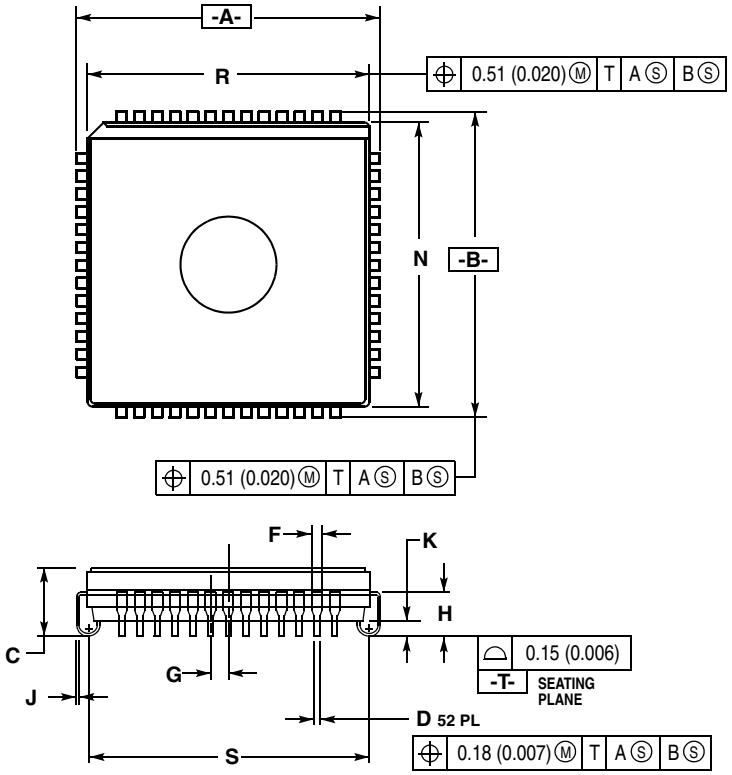
11.5 52-Pin Plastic-Leaded Chip Carrier (Case 778)



NOTES:
1. DATUMS –L–, –M–, AND –N– DETERMINED WHERE TOP OF LEAD SHOULDER EXITS PLASTIC BODY AT MOLD PARTING LINE.
2. DIMENSION G1, TRUE POSITION TO BE MEASURED AT DATUM –T–, SEATING PLANE.
3. DIMENSIONS R AND U DO NOT INCLUDE MOLD FLASH. ALLOWABLE MOLD FLASH IS 0.010 (0.250) PER SIDE.
4. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
5. CONTROLLING DIMENSION: INCH.
6. THE PACKAGE TOP MAY BE SMALLER THAN THE PACKAGE BOTTOM BY UP TO 0.012 (0.300). DIMENSIONS R AND U ARE DETERMINED AT THE OUTERMOST EXTREMES OF THE PLASTIC BODY EXCLUSIVE OF MOLD FLASH, TIE BAR BURRS, GATE BURRS AND INTERLEAD FLASH, BUT INCLUDING ANY MISMATCH BETWEEN THE TOP AND BOTTOM OF THE PLASTIC BODY.
7. DIMENSION H DOES NOT INCLUDE DAMBAR PROTRUSION OR INTRUSION. THE DAMBAR PROTRUSION(S) SHALL NOT CAUSE THE H DIMENSION TO BE GREATER THAN 0.037 (0.940). THE DAMBAR INTRUSION(S) SHALL NOT CAUSE THE H DIMENSION TO BE SMALLER THAN 0.025 (0.635).

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.785	0.795	19.94	20.19
B	0.785	0.795	19.94	20.19
C	0.165	0.180	4.20	4.57
E	0.090	0.110	2.29	2.79
F	0.013	0.019	0.33	0.48
G	0.050 BSC		1.27 BSC	
H	0.026	0.032	0.66	0.81
J	0.020	—	0.51	—
K	0.025	—	0.64	—
R	0.750	0.756	19.05	19.20
U	0.750	0.756	19.05	19.20
V	0.042	0.048	1.07	1.21
W	0.042	0.048	1.07	1.21
X	0.042	0.056	1.07	1.42
Y	—	0.020	—	0.50
Z	2°	10°	2°	10°
G1	0.710	0.730	18.04	18.54
K1	0.040	—	1.02	—

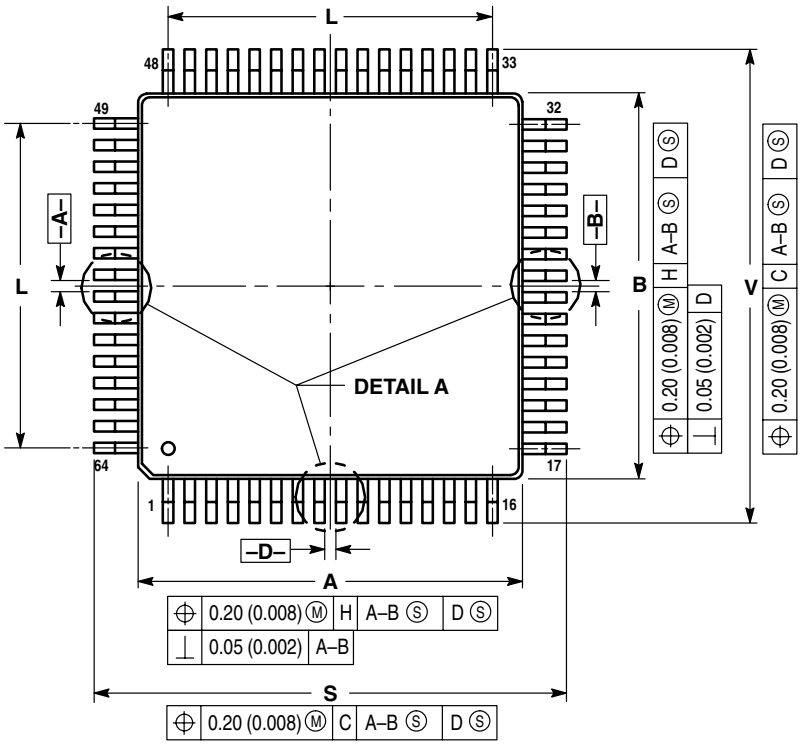
11.6 52-Pin Windowed Ceramic-Leaded Chip Carrier (Case 778B)

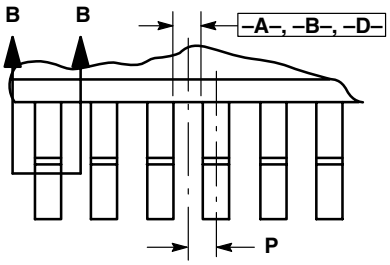


NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION R AND N DO NOT INCLUDE GLASS PROTRUSION. GLASS PROTRUSION TO BE 0.25 (0.010) MAXIMUM.
4. ALL DIMENSIONS AND TOLERANCES INCLUDE LEAD TRIM OFFSET AND LEAD

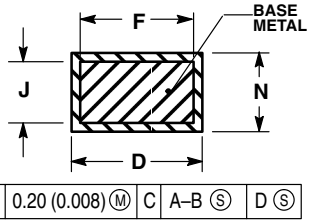
	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.785	0.795	19.94	20.19
B	0.785	0.795	19.94	20.19
C	0.165	0.200	4.20	5.08
D	0.017	0.021	0.44	0.53
F	0.026	0.032	0.67	0.81
G	0.050 BSC		1.27 BSC	
H	0.090	0.130	2.29	3.30
J	0.006	0.010	0.16	0.25
K	0.035	0.045	0.89	1.14
N	0.735	0.756	18.67	19.20
R	0.735	0.756	18.67	19.20
S	0.690	0.730	17.53	18.54

11.7 64-Pin Quad Flat Pack (Case 840C)

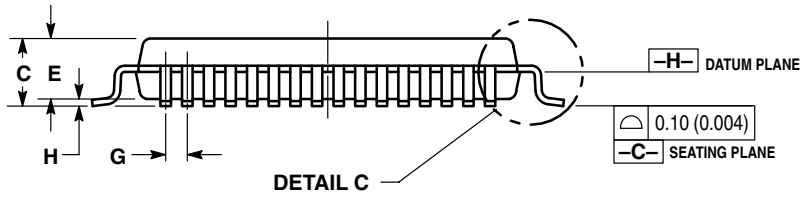




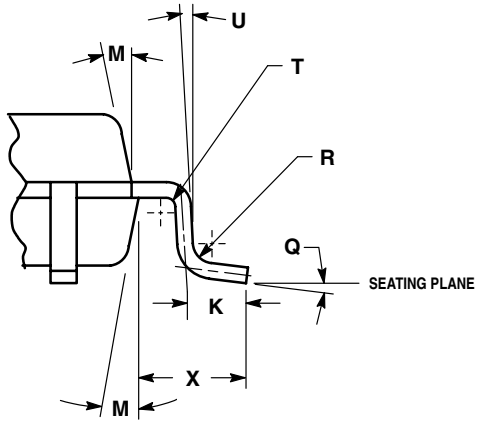
DETAIL A



SECTION B–B



DETAIL C

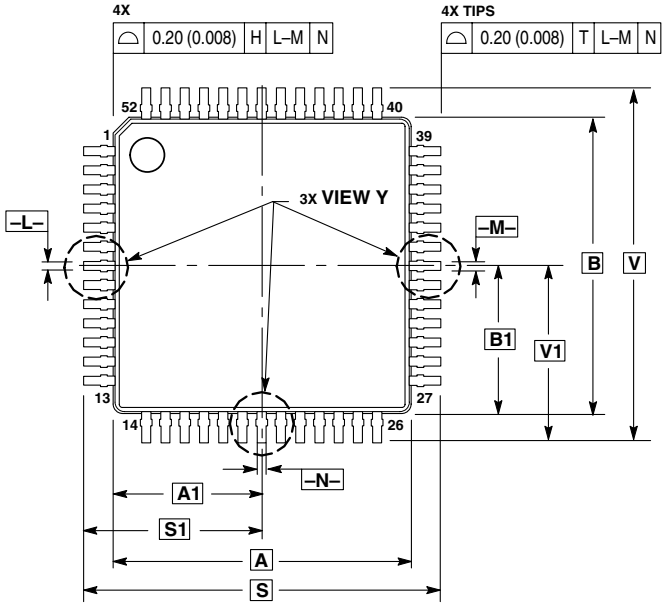


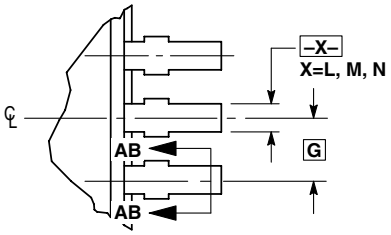
DETAIL C

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 A–B AND –D– TO BE DETERMINED AT DATUM PLANE –H–.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE –C–.
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 D DIMENSION TO EXCEED 0.53 (0.021). DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.
8. DIMENSION K IS TO BE MEASURED FROM THE THEORETICAL INTERSECTION OF LEAD FOOT AND LEG CENTERLINES.

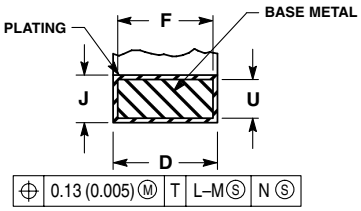
	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	13.90	14.10	0.547	0.555
B	13.90	14.10	0.547	0.555
C	2.07	2.46	0.081	0.097
D	0.30	0.45	0.012	0.018
E	2.00	2.40	0.079	0.094
F	0.30	—	0.012	—
G	0.80 BSC		0.031 BSC	
H	0.067	0.250	0.003	0.010
J	0.130	0.230	0.005	0.090
K	0.50	0.66	0.020	0.026
L	12.00 REF		0.472 REF	
M	5 °	10 °	5 °	10 °
N	0.130	0.170	0.005	0.007
P	0.40 BSC		0.016 BSC	
Q	2 °	8 °	2 °	8 °
R	0.13	0.30	0.005	0.012
S	16.20	16.60	0.638	0.654
T	0.20 REF		0.008 REF	
U	0 °	—	0 °	—
V	16.20	16.60	0.638	0.654
X	1.10	1.30	0.043	0.051

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

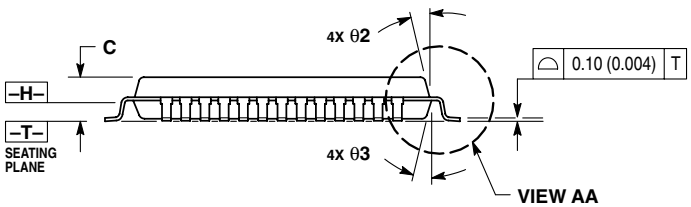


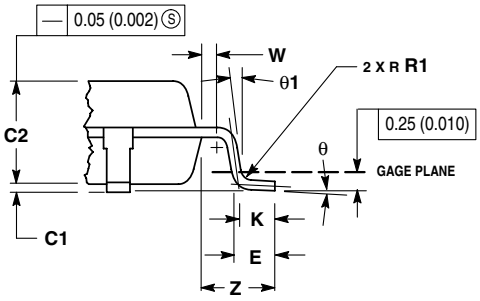


VIEW Y



SECTION AB–AB
ROTATED 90° CLOCKWISE



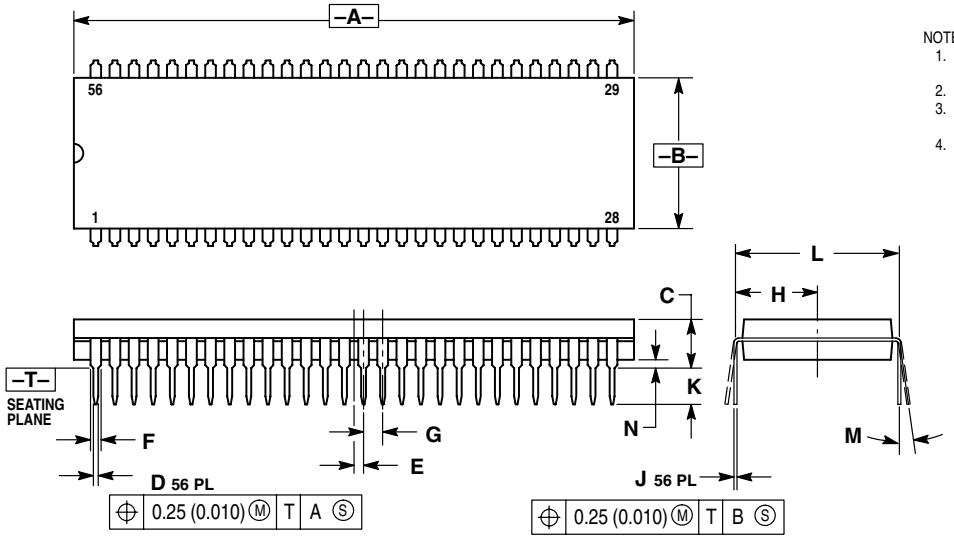


VIEW AA

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).

	MILLIMETERS		INCHES	
DIM	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°

11.9 56-Pin Dual in-Line Package (Case 859)



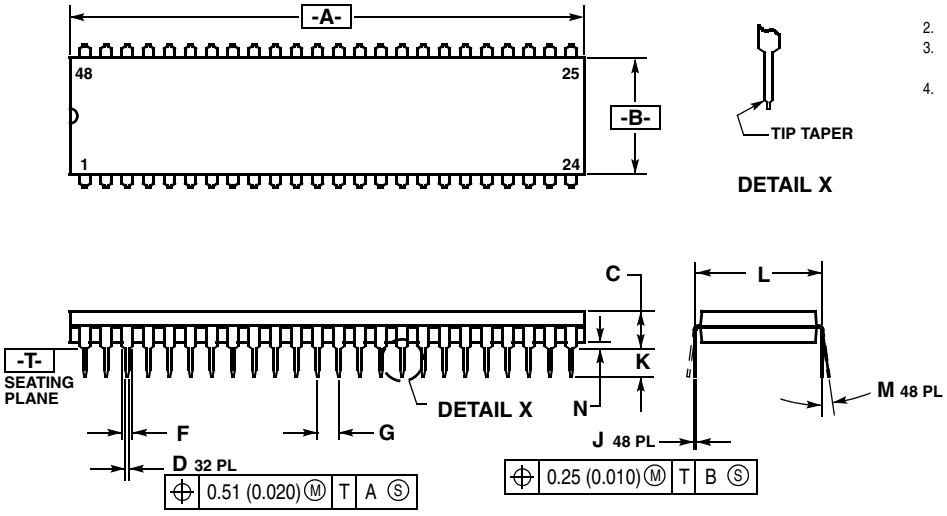
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIMENSIONS A AND B DO NOT INCLUDE MOLD FLASH. MAXIMUM MOLD FLASH 0.25 (0.010)

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	2.035	2.065	51.69	52.45
B	0.540	0.560	13.72	14.22
C	0.155	0.200	3.94	5.08
D	0.014	0.022	0.36	0.56
E	0.035 BSC		0.89 BSC	
F	0.032	0.046	0.81	1.17
G	0.070 BSC		1.778 BSC	
H	0.300 BSC		7.62 BSC	
J	0.008	0.015	0.20	0.38
K	0.115	0.135	2.92	3.43
L	0.600 BSC		15.24 BSC	
M	0 °	15 °	0 °	15 °
N	0.020	0.040	0.51	1.02

11.10 48-Pin Plastic DIP (Case 767)

NOTE

The MC68HC811E2 is the only member of the E series that is offered in a 48-pin plastic dual in-line package.



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIMENSIONS A AND B DO NOT INCLUDE MOLD FLASH. MAXIMUM MOLD FLASH 0.25 (0.010).

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	2.415	2.445	61.34	62.10
B	0.540	0.560	13.72	14.22
C	0.155	0.200	3.94	5.08
D	0.014	0.022	0.36	0.55
F	0.040	0.060	1.02	1.52
G	0.100 BSC		2.54 BSC	
H	0.070 BSC		1.79 BSC	
J	0.008	0.015	0.20	0.38
K	0.115	0.150	2.92	3.81
L	0.600 BSC		15.24 BSC	
M	0×	15×	0×	15×
N	0.020	0.040	0.51	1.01

Appendix A
Development Support

A.1 Introduction

This section provides information on the development support offered for the E-series devices.

A.2 M68HC11 E-Series Development Tools

Device	Package	Emulation Module[1] [2]	Flex Cable[1] [2]	MMDS11 Target Head[1] [2]	SPGMR Programming Adapter[3]
MC68HC11E9 MC68HC711E9	52 FN	M68EM11E20	M68CBL11C	M68TC11E20FN52	M68PA11E20FN52
	52 PB	M68EM11E20	M68CBL11C	M68TC11E20PB52	M68PA11E20PB52
	56 B	M68EM11E20	M68CBL11B	M68TC11E20B56	M68PA11E20B56
	64 FU	M68EM11E20	M68CBL11C	M68TC11E20FU64	M68PA11E20FU64
MC68HC11E20 MC68HC711E20	52 FN	M68EM11E20	M68CBL11C	M68TC11E20FN52	M68PA11E20FN52
	64 FU	M68EM11E20	M68CBL11C	M68TC11E20FU64	M68PA11E20FU64
MC68HC811E2	48 P	M68EM11E20	M68CBL11B	M68TB11E20P48	M68PA11A8P48
	52 FN	M68EM11E20	M68CBL11C	M68TC11E20FN52	M68PA11E20FN52

1. Each MMDS11 system consists of a system console (M68MMDS11), an emulation module, a flex cable, and a target head.
2. A complete EVS consists of a platform board (M68HC11PFB), an emulation module, a flex cable, and a target head.
3. Each SPGMR system consists of a universal serial programmer (M68SPGMR11) and a programming adapter. It can be used alone or in conjunction with the MMDS11.

A.3 EVS — Evaluation System

The EVS is an economical tool for designing, debugging, and evaluating target systems based on the M68HC11. EVS features include:

- Monitor/debugger firmware
- One-line assembler/disassembler
- Host computer download capability
- Dual memory maps:
  - 64-Kbyte monitor map that includes 16 Kbytes of monitor EPROM
  - M68HC11 E-series user map that includes 64 Kbytes of emulation RAM
- MCU extension input/output (I/O) port for single-chip, expanded, and special-test operation modes
- RS-232C terminal and host I/O ports
- Logic analyzer connector

A.4 Modular Development System (MMDS11)

The M68MMDS11 modular development system (MMDS11) is an emulator system for developing embedded systems based on an M68HC11 microcontroller unit (MCU). The MMDS11 provides a bus state analyzer (BSA) and real-time memory windows. The unit's integrated development environment includes an editor, an assembler, user interface, and source-level debug. These features significantly reduce the time necessary to develop and debug an embedded MCU system. The unit's compact size requires a minimum of desk space.

The MMDS11 is one component of Freescale's modular approach to MCU-based product development. This modular approach allows easy configuration of the MMDS11 to fit a wide range of requirements. It also reduces development system cost by allowing the user to purchase only the modular components necessary to support the particular MCU derivative.

MMDS11 features include:

- Real-time, non-intrusive, in-circuit emulation at the MCU’s operating frequency
- Real-time bus state analyzer
  - 8 K x 64 real-time trace buffer
  - Display of real-time trace data as raw data, disassembled instructions, raw data and disassembled instructions, or assembly-language source code
  - Four hardware triggers for commencing trace and to provide breakpoints
  - Nine triggering modes
  - As many as 8190 pre- or post-trigger points for trace data
  - 16 general-purpose logic clips, four of which can be used to trigger the bus state analyzer sequencer
  - 16-bit time tag or an optional 24-bit time tag that reduces the logic clips traced from 16 to eight
- Four data breakpoints (hardware breakpoints)
- Hardware instruction breakpoints over either the 64-Kbyte M68HC11 memory map or over a 1-Mbyte bank switched memory map
- 32 real-time variables, nine of which can be displayed in the variables window. These variables may be read or written while the MCU is running
- 32 bytes of real-time memory can be displayed in the memory window. This memory may be read or written while the MCU is running
- 64 Kbytes of fast emulation memory (SRAM)
- Current-limited target input/output connections
- Six software-selectable oscillator clock sources: five internally generated frequencies and an external frequency via a bus analyzer logic clip
- Command and response logging to MS-DOS® disk files to save session history
- SCRIPT command for automatic execution of a sequence of MMDS11 commands
- Assembly or C-language source-level debugging with global variable viewing
- Host/emulator communications speeds as high as 57,600 baud for quick program loading

® MS-DOS is a registered trademark of Microsoft Corporation.

- Extensive on-line MCU information via the CHIPINFO command. View memory map, vectors, register, and pinout information pertaining to the device being emulated
- Host software supports:
  – An editor
  – An assembler and user interface
  – Source-level debug
  – Bus state analysis
  – IBM® mouse

A.5 SPGMR11 — Serial Programmer for M68HC11 MCUs

The SPGMR11 is a modular EPROM/EEPROM programming tool for all M68HC11 devices. The programmer features interchangeable adapters that allow programming of various M68HC11 package types.

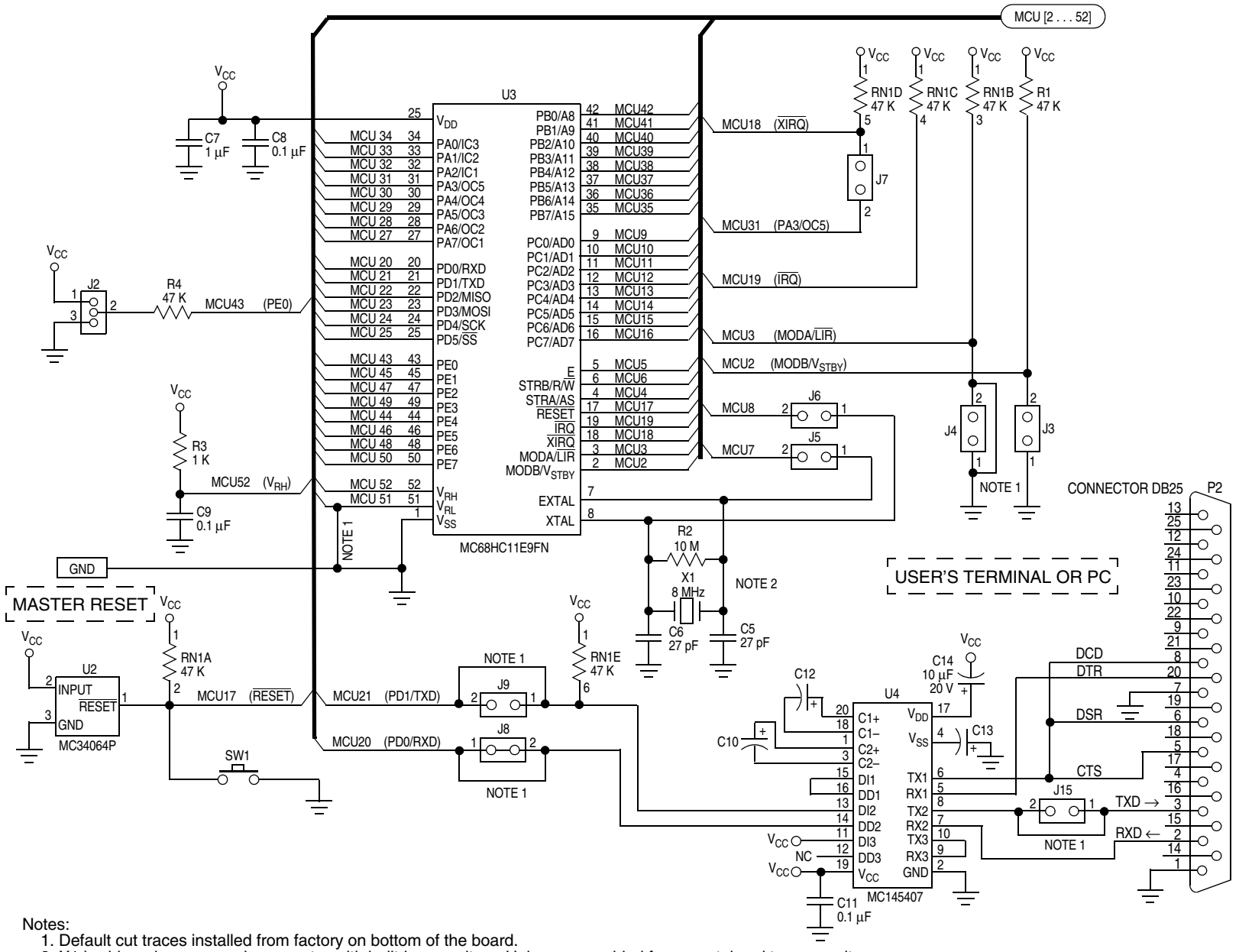
Programmer features include:

- Programs M68HC11 Family devices that contain an EPROM or EEPROM array.
- Can be operated as a stand-alone programmer connected to a host computer or connected between a host computer and the M68HC11 modular development system (MMDS11) station module
- Uses plug-in programming adapters to accommodate a variety of MCU devices and packages
- On-board programming voltage circuit eliminates the need for an external 12-volt supply.
- Includes programming software and a user's manual
- Includes a +5-volt power cable and a DB9 to DB25 connector adapter

® IBM is a registered trademark145 of International Business Machines Corporation.

Appendix B
EVBU Schematic

Refer to Figure B-1 for a schematic diagram of the M68HC11EVBU Universal Evaluation Board. This diagram is included for reference only.



Notes:
1. Default cut traces installed from factory on bottom of the board.
2. X1 is shipped as a ceramic resonator with built-in capacitors. Holes are provided for a crystal and two capacitors.

Figure B-1. EVBU Schematic 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

© Freescale Semiconductor, Inc., 2005. All rights reserved.

freescale™ semiconductor

Basic Bootstrap Mode

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

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

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

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

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

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

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

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

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

Bootstrap mode can also be used to interactively calibrate critical analog sensors. Since this calibration is done in the final assembled system, it can compensate for any errors in discrete interface circuitry and cabling between the sensor and the analog inputs to the MCU. Note that this calibration routine is a downloaded program that does not take up space in the normal application program.

Bootstrap Mode Logic

In the M68HC11 MCUs, very little logic is dedicated to the bootstrap mode. Consequently, this mode adds almost no extra cost to the MCU system. The biggest piece of circuitry for bootstrap mode is the small boot ROM. This ROM is 192 bytes in the original MC68HC11A8, but some of the newest members of the M68HC11 Family, such as the MC68HC711K4, have as much as 448 bytes to accommodate added features. Normally, this boot ROM is present in the memory map only when the MCU is reset in bootstrap mode to prevent interference with the user's normal memory space. The enable for this ROM is controlled by the read boot ROM (RBOOT) control bit in the highest priority interrupt (HPRIO) register. The RBOOT bit can be written by software whenever the MCU is in special test or special bootstrap modes; when the MCU is in normal modes, RBOOT reverts to 0 and becomes a read-only bit. All other logic in the MCU would be present whether or not there was a bootstrap mode.

Figure 1 shows the composite memory map of the MC68HC711E9 in its four basic modes of operation, including bootstrap mode. The active mode is determined by the mode A (MDA) and special mode (SMOD) control bits in the HPRIO control register. These control bits are in turn controlled by the state of the mode A (MODA) and mode B (MODB) pins during reset. Table 1 shows the relationship between the state of these pins during reset, the selected mode, and the state of the MDA, SMOD, and RBOOT control bits. Refer to the composite memory map and information in Table 1 for the following discussion.

The MDA control bit is determined by the state of the MODA pin as the MCU leaves reset. MDA selects between single-chip and expanded operating modes. When MDA is 0, a single-chip mode is selected, either normal single-chip mode or special bootstrap mode. When MDA is 1, an expanded mode is selected, either normal expanded mode or special test mode.

The SMOD control bit is determined by the inverted state of the MODB pin as the MCU leaves reset. SMOD controls whether a normal mode or a special mode is selected. When SMOD is 0, one of the two normal modes is selected, either normal single-chip mode or normal expanded mode. When SMOD is 1, one of the two special modes is selected, either special bootstrap mode or special test mode. When either special mode is in effect (SMOD = 1), certain privileges are in effect, for instance, the ability to write to the mode control bits and fetching the reset and interrupt vectors from $BFxx rather than $FFxx.

Table 1. Mode Selection Summary

Input Pins		Mode Selected	Control Bits in HPRIO		
MODB	MODA		RBOOT	SMOD	MDA
1	0	Normal single chip	0	0	0
0	0	Normal expanded	0	0	1
0	0	Special bootstrap	1	1	0
0	1	Special test	0	1	1

The alternate vector locations are achieved by simply driving address bit A14 low during all vector fetches if SMOD = 1. For special test mode, the alternate vector locations assure that the reset vector can be fetched from external memory space so the test system can control MCU operation. In special bootstrap mode, the small boot ROM is enabled in the memory map by RBOOT = 1 so the reset vector will be fetched from this ROM and the bootloader firmware will control MCU operation.

RBOOT is reset to 1 in bootstrap mode to enable the small boot ROM. In the other three modes, RBOOT is reset to 0 to keep the boot ROM out of the memory map. While in special test mode, SMOD = 1, which allows the RBOOT control bit to be written to 1 by software to enable the boot ROM for testing purposes.

Boot ROM Firmware

The main program in the boot ROM is the bootloader, which is automatically executed as a result of resetting the MCU in bootstrap mode. Some newer versions of the M68HC11 Family have additional utility programs that can be called from a downloaded program. One utility is available to program EPROM or OTP versions of the M68HC11. A second utility allows the contents of memory locations to be uploaded to a host computer. In the MC68HC711K4 boot ROM, a section of code is used by Freescale for stress testing the on-chip EEPROM. These test and utility programs are similar to self-test ROM programs in other MCUs except that the boot ROM does not use valuable space in the normal memory map.

Bootstrap firmware is also involved in an optional EEPROM security function on some versions of the M68HC11. This EEPROM security feature prevents a software pirate from seeing what is in the on-chip EEPROM. The secured state is invoked by programming the no security (NOSEC) EEPROM bit in the CONFIG register. Once this NOSEC bit is programmed to 0, the MCU will ignore the mode A pin and always come out of reset in normal single-chip mode or special bootstrap mode, depending on the state of the mode B pin. Normal single-chip mode is the usual way a secured part would be used. Special bootstrap mode is used to disengage the security function (only after the contents of EEPROM and RAM have been erased). Refer to the *M68HC11 Reference Manual*, Freescale document order number M68HC11RM/AD, for additional information on the security mode and complete listings of the boot ROMs that support the EEPROM security functions.

Automatic Selection of Baud Rate

The bootloader program in the MC68HC711E9 accommodates either of two baud rates.

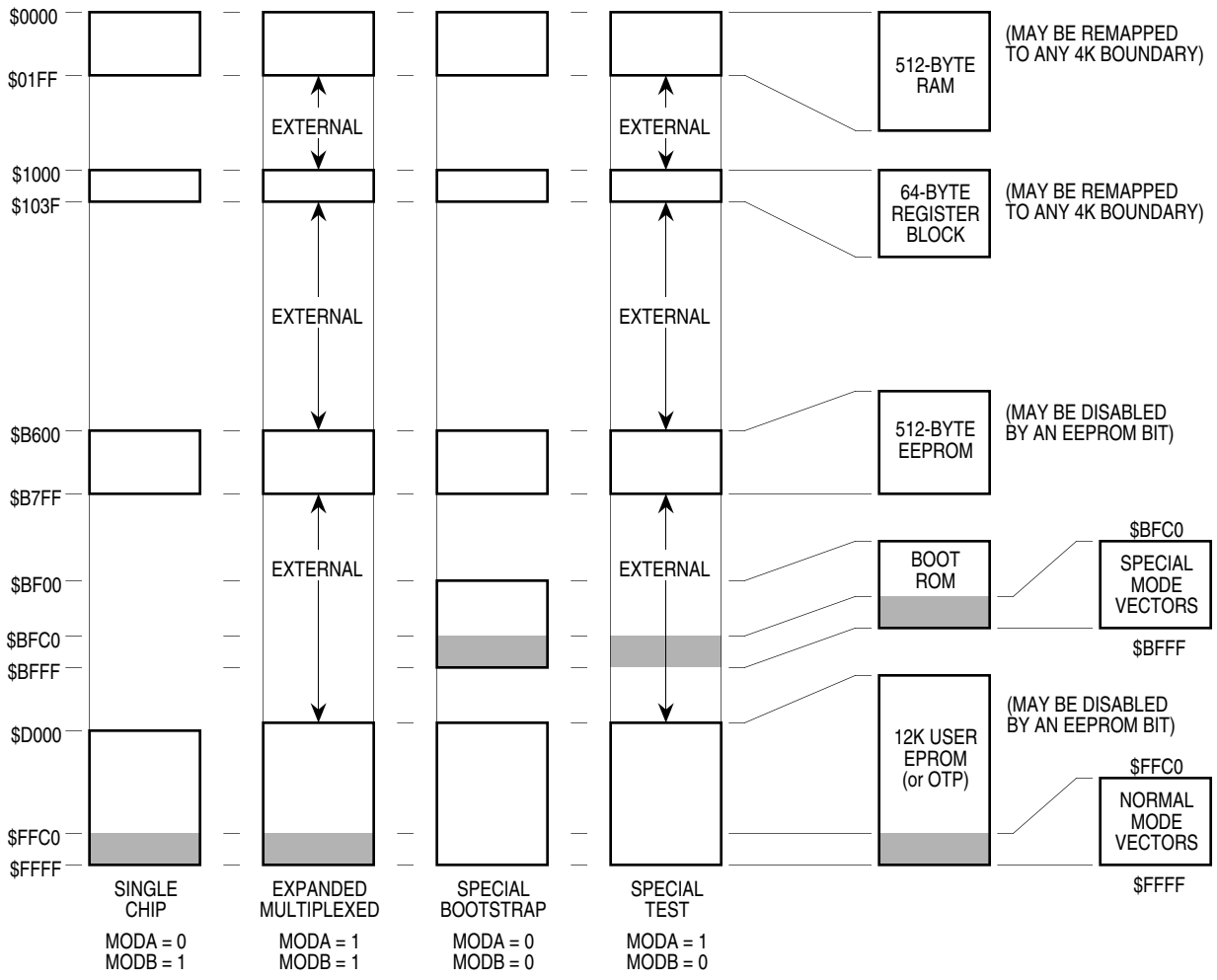
- The higher of these baud rates (7812 baud at a 2-MHz E-clock rate) is used in systems that operate from a binary frequency crystal such as 2^{23} Hz (8.389 MHz). At this crystal frequency, the baud rate is 8192 baud, which was used extensively in automotive applications.
- The second baud rate available to the M68HC11 bootloader is 1200 baud at a 2-MHz E-clock rate. Some of the newest versions of the M68HC11, including the MC68HC11F1 and MC68HC117K4, accommodate other baud rates using the same differentiation technique explained here. Refer to the reference numbers in square brackets in Figure 2 during the following explanation.

NOTE

Software can change some aspects of the memory map after reset.

Figure 2 shows how the bootloader program differentiates between the default baud rate (7812 baud at a 2-MHz E-clock rate) and the alternate baud rate (1200 baud at a 2-MHz E-clock rate). The host computer sends an initial $FF character, which is used by the bootloader to determine the baud rate that will be used for the downloading operation. The top half of Figure 2 shows normal reception of $FF. Receive data samples at [1] detect the falling edge of the start bit and then verify the start bit by taking a sample at the center of the start bit time. Samples are then taken at the middle of each bit time [2] to reconstruct the value of the received character (all 1s in this case). A sample is then taken at the middle of the stop bit time as a framing check (a 1 is expected) [3]. Unless another character immediately follows this $FF character, the receive data line will idle in the high state as shown at [4].

The bottom half of Figure 2 shows how the receiver will incorrectly receive the $FF character that is sent from the host at 1200 baud. Because the receiver is set to 7812 baud, the receive data samples are taken at the same times as in the upper half of Figure 2. The start bit at 1200 baud [5] is 6.5 times as long as the start bit at 7812 baud [6].



NOTE: Software can change some aspects of the memory map after reset.

Figure 1. MC68HC711E9 Composite Memory Map

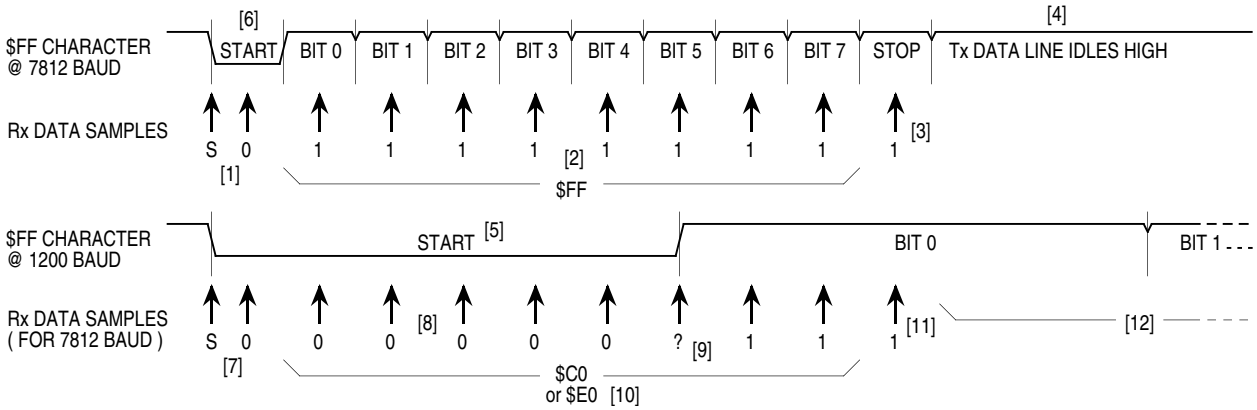


Figure 2. Automatic Detection of Baud Rate

Samples taken at [7] detect the failing edge of the start bit and verify it is a logic 0. Samples taken at the middle of what the receiver interprets as the first five bit times [8] detect logic 0s. The sample taken at the middle of what the receiver interprets as bit 5 [9] may detect either a 0 or a 1 because the receive data has a rising transition at about this time. The samples for bits 6 and 7 detect 1s, causing the receiver to think the received character was $C0 or $E0 [10] at 7812 baud instead of the $FF which was sent at 1200 baud. The stop bit sample detects a 1 as expected [11], but this detection is actually in the middle of bit 0 of the 1200 baud $FF character. The SCI receiver is not confused by the rest of the 1200 baud $FF character because the receive data line is high [12] just as it would be for the idle condition. If a character other than $FF is sent as the first character, an SCI receive error could result.

Main Bootloader Program

Figure 3 is a flowchart of the main bootloader program in the MC68HC711E9. This bootloader demonstrates the most important features of the bootloaders used on all M68HC11 Family members. For complete listings of other M68HC11 versions, refer to Listing 3. MC68HC711E9 Bootloader ROM at the end of this application note, and to **Appendix B** of the *M68HC11 Reference Manual*, Freescale document order number M68HC11RM/AD.

The reset vector in the boot ROM points to the start [1] of this program. The initialization block [2] establishes starting conditions and sets up the SCI and port D. The stack pointer is set because there are push and pull instructions in the bootloader program. The X index register is pointed at the start of the register block ($1000) so indexed addressing can be used. Indexed addressing takes one less byte of ROM space than extended instructions, and bit manipulation instructions are not available in extended addressing forms. The port D wire-OR mode (DWOM) bit in the serial peripheral interface control register (SPCR) is set to configure port D for wired-OR operation to minimize potential conflicts with external systems that use the PD1/TxD pin as an input. The baud rate for the SCI is initially set to 7812 baud at a 2-MHz E-clock rate but can automatically switch to 1200 baud based on the first character received. The SCI receiver and transmitter are enabled. The receiver is required by the bootloading process, and the transmitter is used to transmit data back to the host computer for optional verification. The last item in the initialization is to set an intercharacter delay constant used to terminate the download when the host computer stops sending data to the MC68HC711E9. This delay constant is stored in the timer output compare 1 (TOC1) register, but the on-chip timer is not used in the bootloader program. This example

illustrates the extreme measures used in the bootloader firmware to minimize memory usage. However, such measures are not usually considered good programming technique because they are misleading to someone trying to understand the program or use it as an example.

After initialization, a break character is transmitted [3] by the SCI. By connecting the TxD pin to the RxD pin (with a pullup because of port D wired-OR mode), this break will be received as a $00 character and cause an immediate jump [4] to the start of the on-chip EEPROM ($B600 in the MC68HC711E9). This feature is useful to pass control to a program in EEPROM essentially from reset. Refer to Common Bootstrap Mode Problems before using this feature.

If the first character is received as $FF, the baud rate is assumed to be the default rate (7812 baud at a 2-MHz E-clock rate). If $FF was sent at 1200 baud by the host, the SCI will receive the character as $E0 or $C0 because of the baud rate mismatch, and the bootloader will switch to 1200 baud [5] for the rest of the download operation. When the baud rate is switched to 1200 baud, the delay constant used to monitor the intercharacter delay also must be changed to reflect the new character time.

At [6], the Y index register is initialized to $0000 to point to the start of on-chip RAM. The index register Y is used to keep track of where the next received data byte will be stored in RAM. The main loop for loading begins at [7].

The number of data bytes in the downloaded program can be any number between 0 and 512 bytes (the size of on-chip RAM). This procedure is called "variable-length download" and is accomplished by ending the download sequence when an idle time of at least four character times occurs after the last character to be downloaded. In M68HC11 Family members which have 256 bytes of RAM, the download length is fixed at exactly 256 bytes plus the leading $FF character.

The intercharacter delay counter is started [8] by loading the delay constant from TOC1 into the X index register. The 19-E-cycle wait loop is executed repeatedly until either a character is received [9] or the allowed intercharacter delay time expires [10]. For 7812 baud, the delay constant is 10,241 E cycles (539 x 19 E cycles per loop). Four character times at 7812 baud is 10,240 E cycles (baud prescale of 4 x baud divider of 4 x 16 internal SCI clocks/bit time x 10 bit times/character x 4 character times). The delay from reset to the initial $FF character is not critical since the delay counter is not started until after the first character ($FF) is received.

To terminate the bootloading sequence and jump to the start of RAM without downloading any data to the on-chip RAM, simply send $FF and nothing else. This feature is similar to the jump to EEPROM at [4] except the $FF causes a jump to the start of RAM. This procedure requires that the RAM has been loaded with a valid program since it would make no sense to jump to a location in uninitialized memory.

After receiving a character, the downloaded byte is stored in RAM [11]. The data is transmitted back to the host [12] as an indication that the download is progressing normally. At [13], the RAM pointer is incremented to the next RAM address. If the RAM pointer has not passed the end of RAM, the main download loop (from [7] to [14]) is repeated.

When all data has been downloaded, the bootloader goes to [16] because of an intercharacter delay timeout [10] or because the entire 512-byte RAM has been filled [15]. At [16], the X and Y index registers are set up for calling the PROGRAM utility routine, which saves the user from having to do this in a downloaded program. The PROGRAM utility is fully explained in EPROM Programming Utility. The final step of the bootloader program is to jump to the start of RAM [17], which starts the user's downloaded program.

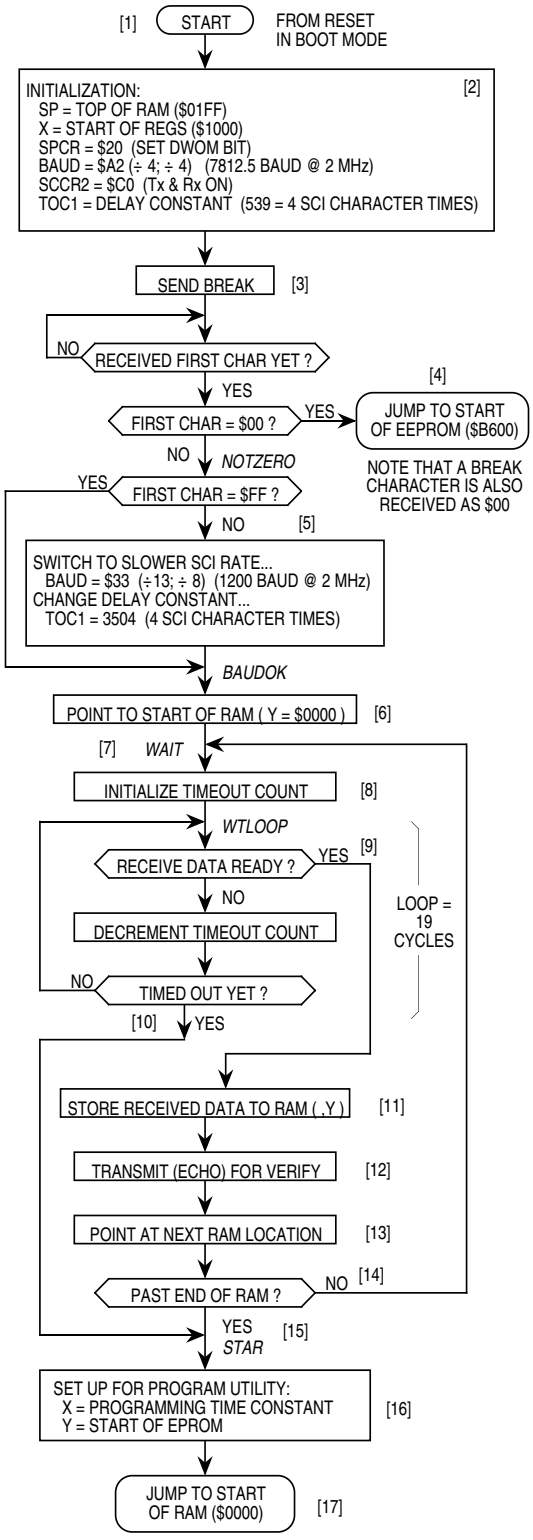


Figure 3. MC68HC711E9 Bootloader Flowchart

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.

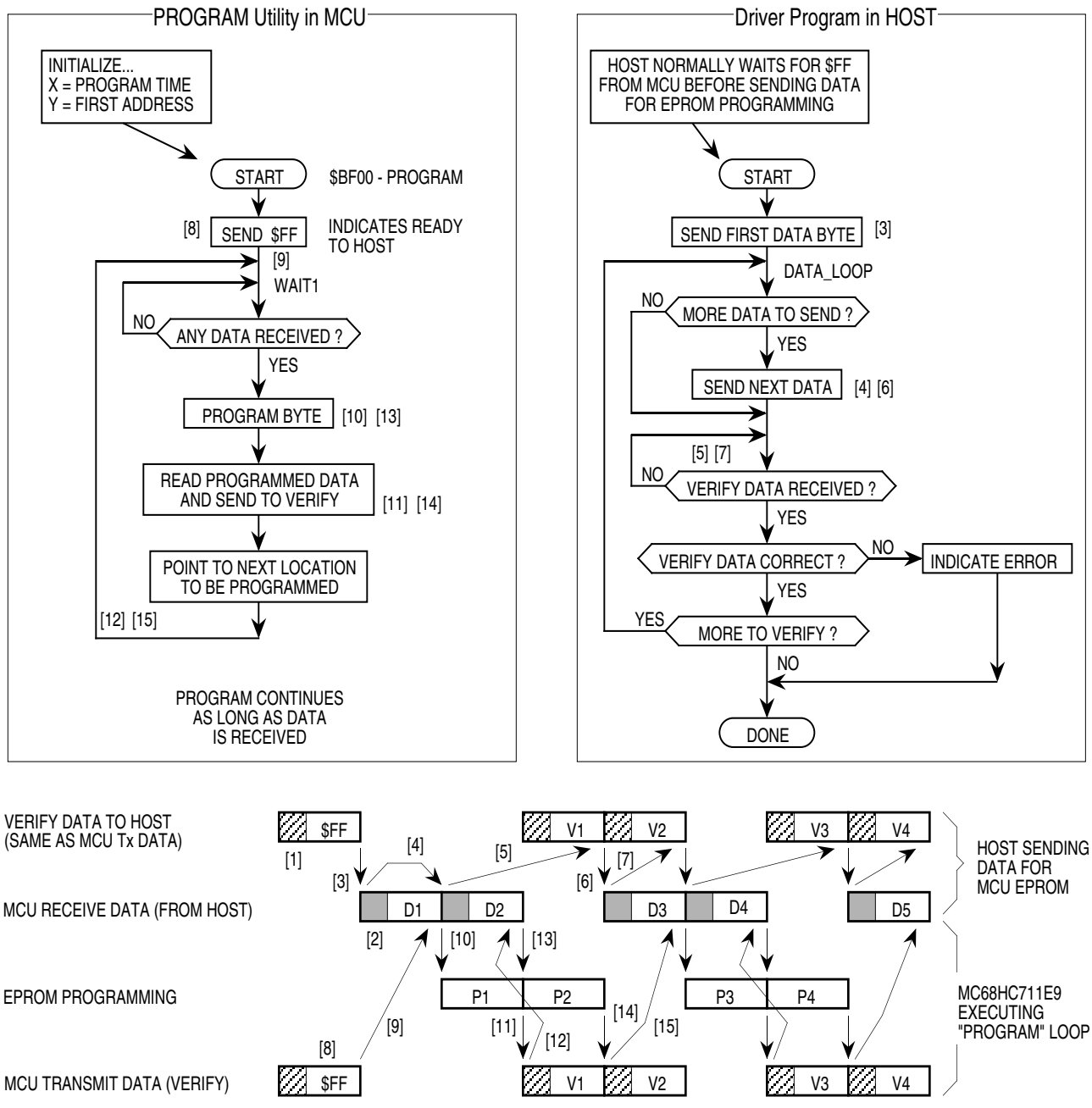


Figure 4. Host and MCU Activity during EPROM PROGRAM Utility

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.

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

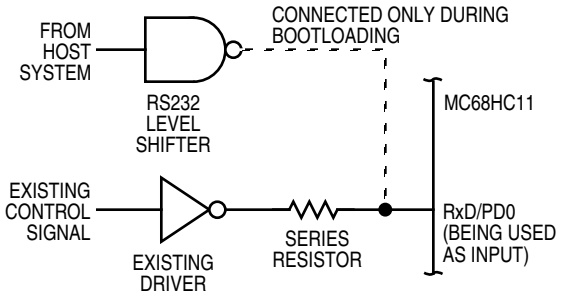


Figure 5. Preventing Driver Conflict

TxD Pin

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

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

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

Other

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

Driving Boot Mode from Another M68HC11

A second M68HC11 system can easily act as the host to drive bootstrap loading of an M68HC11 MCU. This method is used to examine and program non-volatile memories in target M68HC11s in Freescale EVMs. The following hardware and software example will demonstrate this and other bootstrap mode features.

The schematic in Figure 6 shows the circuitry for a simple EPROM duplicator for the MC68HC711E9. The circuitry is built in the wire-wrap area of an M68HC11EVBU evaluation board to simplify construction. The schematic shows only the important portions of the EVBU circuitry to avoid confusion. To see the complete EVBU schematic, refer to the *M68HC11EVBU Universal Evaluation Board User's Manual*, Freescale document order number M68HC11EVBU/D.

The default configuration of the EVBU must be changed to make the appropriate connections to the circuitry in the wire-wrap area and to configure the master MCU for bootstrap mode. A fabricated jumper must be installed at J6 to connect the XTAL output of the master MCU to the wire-wrap connector P5, which has been wired to the EXTAL input of the target MCU. Cut traces that short across J8 and J9 must be cut on the solder side of the printed circuit board to disconnect the normal SCI connections to the RS232 level translator (U4) of the EVBU. The J8 and J9 connections can be restored easily at a later time by installing fabricated jumpers on the component side of the board. A fabricated jumper must be installed across J3 to configure the master MCU for bootstrap mode.

One MC68HC711E9 is first programmed by other means with a desired 12-Kbyte program in its EPROM and a small duplicator program in its EEPROM. Alternately, the ROM program in an MC68HC11E9 can be copied into the EPROM of a target MC68HC711E9 by programming only the duplicator program into the EEPROM of the master MC68HC11E9. The master MCU is installed in the EVBU at socket U3. A blank MC68HC711E9 to be programmed is placed in the socket in the wire-wrap area of the EVBU (U6).

With the V_{PP} power switch off, power is applied to the EVBU system. As power is applied to the EVBU, the master MCU (U3) comes out of reset in bootstrap mode. Target MCU (U6) is held in reset by the PB7 output of master MCU (U3). The PB7 output of U3 is forced to 0 when U3 is reset. The master MCU will later release the reset signal to the target MCU under software control. The RxD and TxD pins of the target MCU (U6) are high-impedance inputs while U6 is in reset so they will not affect the TxD and RxD signals of the master MCU (U3) while U3 is coming out of reset. Since the target MCU is being held in reset with MODA and MODB at 0, it is configured for the PROG EPROM emulation mode, and PB7 is the output enable signal for the EPROM data I/O (input/output) pins. Pullup resistor R7 causes the port D pins, including RxD and TxD, to remain in the high-impedance state so they do not interfere with the RxD and TxD pins of the master MCU as it comes out of reset.

As U3 leaves reset, its mode pins select bootstrap mode so the bootloader firmware begins executing. A break is sent out the TxD pin of U3. Pullup resistor R10 and resistor R9 cause the break character to be seen at the RxD pin of U3. The bootloader performs a jump to the start of EEPROM in the master MCU (U3) and starts executing the duplicator program. This sequence demonstrates how to use bootstrap mode to pass control to the start of EEPROM after reset.

The complete listing for the duplicator program in the EEPROM of the master MCU is provided in Listing 1. MCU-to-MCU Duplicator Program.

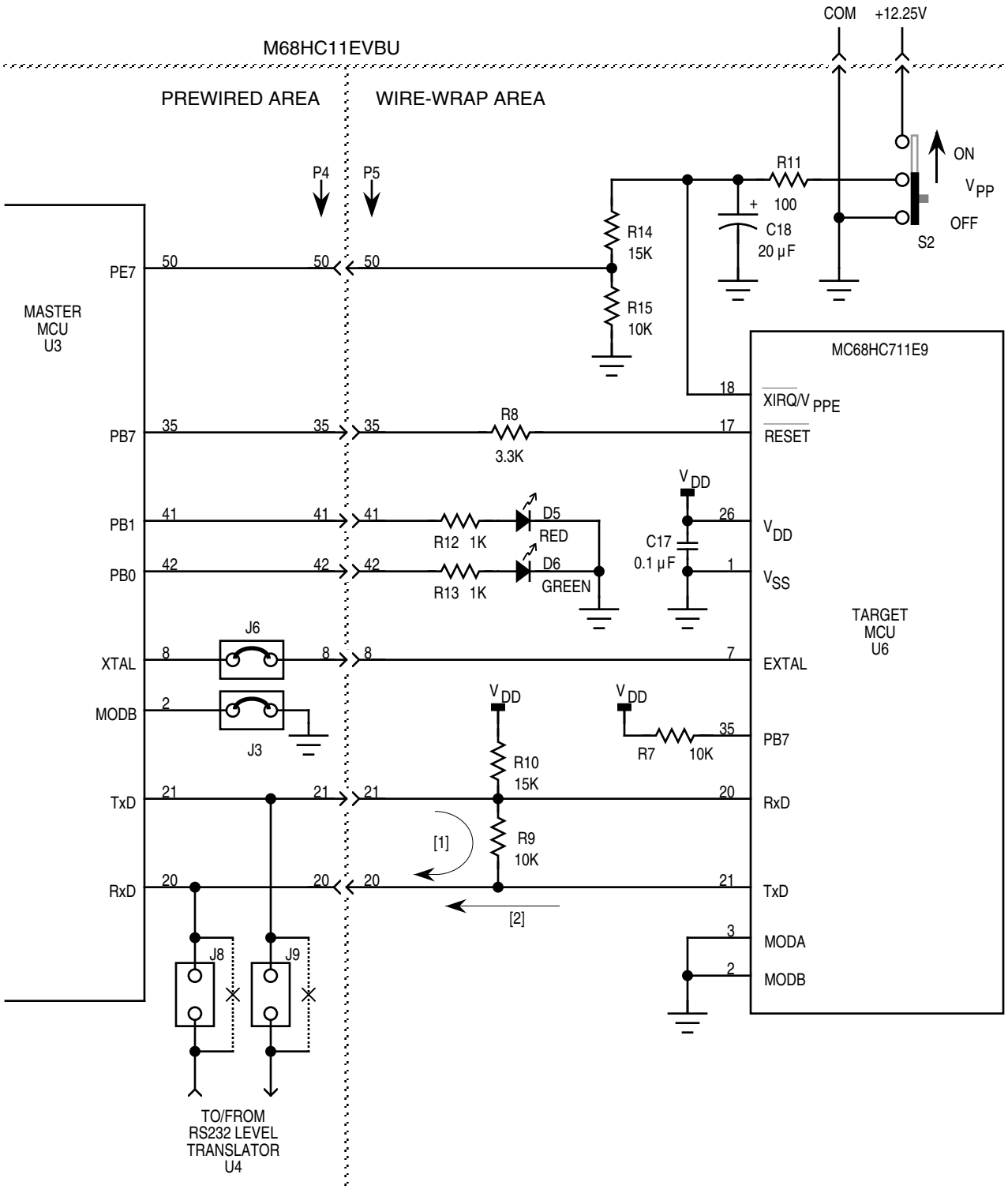


Figure 6. MCU-to-MCU EPROM Duplicator Schematic

The duplicator program in EEPROM clears the DWOM control bit to change port D (thus, TxD) of U3 to normal driven outputs. This configuration will prevent interference due to R9 when TxD from the target MCU (U6) becomes active. Series resistor R9 demonstrates how TxD of U3 can drive RxD of U3[1] and later TxD of U6 can drive RxD of U3 without a destructive conflict between the TxD output buffers.

As the target MCU (U6) leaves reset, its mode pins select bootstrap mode so the bootloader firmware begins executing. A break is sent out the TxD pin of U6. At this time, the TxD pin of U3 is at a driven high so R9 acts as a pullup resistor for TxD of the target MCU (U6). The break character sent from U6 is received by U3 so the duplicator program that is running in the EEPROM of the master MCU knows that the target MCU is ready to accept a bootloaded program.

The master MCU sends a leading $FF character to set the baud rate in the target MCU. Next, the master MCU passes a 3-instruction program to the target MCU and pauses so the bootstrap program in the target MCU will stop the loading process and jump to the start of the downloaded program. This sequence demonstrates the variable-length download feature of the MC68HC711E9 bootloader.

The short program downloaded to the target MCU clears the DWOM bit to change its TxD pin to a normal driven CMOS output and jumps to the EPROM programming utility in the bootstrap ROM of the target MCU.

Note that the small downloaded program did not have to set up the SCI or initialize any parameters for the EPROM programming process. The bootstrap software that ran prior to the loaded program left the SCI turned on and configured in a way that was compatible with the SCI in the master MCU (the duplicator program in the master MCU also did not have to set up the SCI for the same reason). The programming time and starting address for EPROM programming in the target MCU were also set to default values by the bootloader software before jumping to the start of the downloaded program.

Before the EPROM in the target MCU can be programmed, the V_{PP} power supply must be available at the $\overline{XIRQ}/V_{PPE}$ pin of the target MCU. The duplicator program running in the master MCU monitors this voltage (for presence or absence, not level) at PE7 through resistor divider R14–RI5. 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.

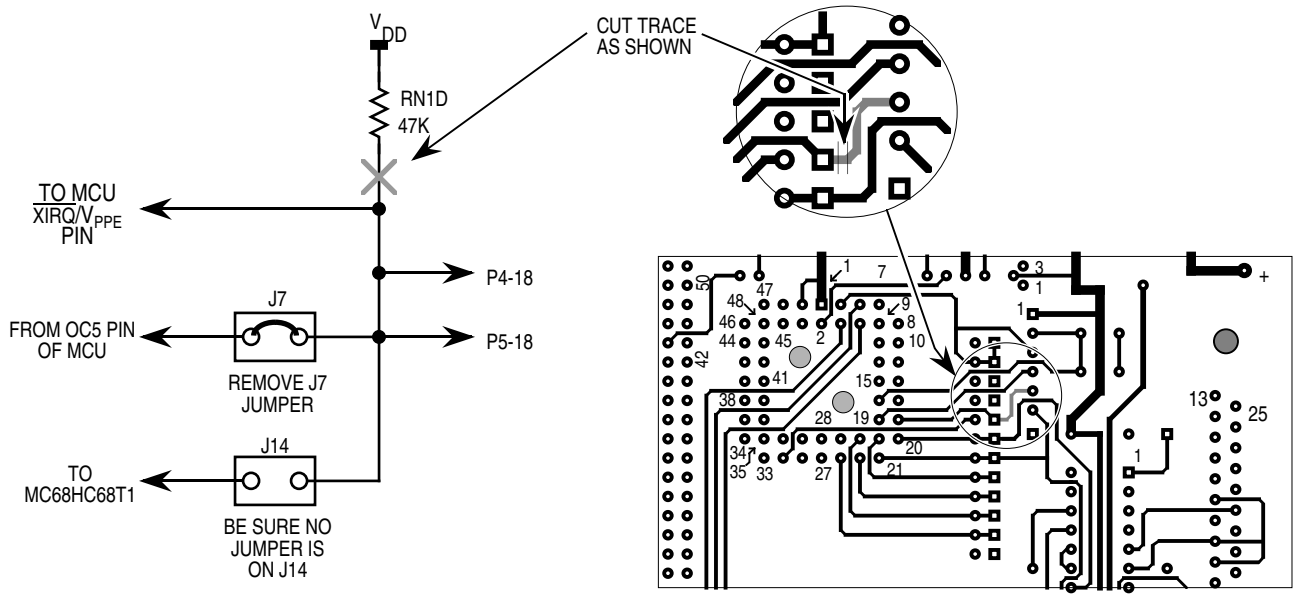


Figure 7. Isolating EVBU $\overline{\text{XIRQ}}$ Pin

Listing 1. MCU-to-MCU Duplicator Program

```
 1              ************************************************
 2              * 68HC711E9 Duplicator Program for AN1060
 3              ************************************************
 4
 5              *****
 6              * Equates - All reg addrs except INIT are 2-digit
 7              *           for direct addressing
 8              *****
 9 103D         INIT       EQU    $103D            RAM, Reg mapping
10 0028         SPCR       EQU    $28              DWOM in bit-5
11 0004         PORTB      EQU    $04              Red LED = bit-1, Grn = bit-0
12              * Reset of prog socket = bit-7
13 0080         RESET      EQU    %10000000
14 0002         RED        EQU    %00000010
15 0001         GREEN      EQU    %00000001
16 000A         PORTE      EQU    $0A              Vpp Sense in bit-7, 1=ON
17 002E         SCSR       EQU    $2E              SCI status register
18              * TDRE, TC, RDRF, IDLE; OR, NF, FE, -
19 0080         TDRE       EQU    %10000000
20 0020         RDRF       EQU    %00100000
21 002F         SCDR       EQU    $2F              SCI data register
22 BF00         PROGRAM    EQU    $BF00            EPROM prog utility in boot ROM
23 D000         EPSTRT     EQU    $D000            Starting address of EPROM
24
25 B600                    ORG    $B600            Start of EEPROM
26
```

```
27                ************************************************
28                *
29 B600 7F103D   BEGIN      CLR    INIT             Moves Registers to $0000-3F
30 B603 8604               LDAA   #$04             Pattern for DWOM off, no SPI
31 B605 9728               STAA   SPCR             Turns off DWOM in EVBU MCU
32 B607 8680               LDAA   #RESET
33 B609 9704               STAA   PORTB            Release reset to target MCU
34 B60B 132E20FC WT4BRK    BRCLR  SCSR RDRF WT4BRK Loop till char received
35 B60F 86FF               LDAA   #$FF             Leading char for bootload ...
36 B611 972F               STAA   SCDR             to target MCU
37 B613 CEB675             LDX    #BLPROG          Point at program for target
38 B616 8D53     BLLOOP    BSR    SEND1            Bootload to target
39 B618 8CB67D             CPX    #ENDBPR          Past end ?
40 B61B 26F9               BNE    BLLOOP           Continue till all sent
41                *****
42                * Delay for about 4 char times to allow boot related
43                * SCI communications to finish before clearing
44                * Rx related flags
45 B61D CE06A7             LDX    #1703            # of 6 cyc loops
46 B620 09       DLYLP     DEX                     [3]
47 B621 26FD               BNE    DLYLP            [3] Total loop time = 6 cyc
48 B623 962E               LDAA   SCSR             Read status (RDRF will be set)
49 B625 962F               LDAA   SCDR             Read SCI data reg to clear RDRF
50                *****
51                * Now wait for character from target to indicate it's ready for
52                * data to be programmed into EPROM
53 B627 132E20FC WT4FF     BRCLR  SCSR RDRF WT4FF  Wait for RDRF
54 B62B 962F               LDAA   SCDR             Clear RDRF, don't need data
55 B62D CED000             LDX    #EPSTRT          Point at start of EPROM
56                * Handle turn-on of Vpp
57 B630 18CE523D WT4VPP    LDY    #21053           Delay counter (about 200ms)
58 B634 150402             BCLR   PORTB RED        Turn off RED LED
59 B637 960A     DLYLP2    LDAA   PORTE            [3] Wait for Vpp to be ON
60 B639 2AF5               BPL    WT4VPP           [3] Vpp sense is on port E MSB
61 B63B 140402             BSET   PORTB RED        [6] Turn on RED LED
62 B63E 1809               DEY                     [4]
63 B640 26F5               BNE    DLYLP2           [3] Total loop time = 19 cyc
64                * Vpp has been stable for 200ms
65
66 B642 18CED000           LDY    #EPSTRT          X=Tx pointer, Y=verify pointer
67 B646 8D23               BSR    SEND1            Send first data to target
68 B648 8C0000   DATALP    CPX    #0               X points at $0000 after last
69 B64B 2702               BEQ    VERF             Skip send if no more
70 B64D 8D1C               BSR    SEND1            Send another data char
71 B64F 132E20FC VERF      BRCLR  SCSR RDRF VERF   Wait for Rx ready
72 B653 962F               LDAA   SCDR             Get char and clr RDRF
73 B655 18A100             CMPA   0,Y              Does char verify ?
74 B658 2705               BEQ    VERFOK           Skip error if OK
75 B65A 150403             BCLR   PORTB (RED+GREEN) Turn off LEDs
76 B65D 2007               BRA    DUNPRG           Done (programming failed)
77 B65F
78 B65F 1808     VERFOK    INY                     Advance verify pointer
79 B661 26E5               BNE    DATALP           Continue till all done
80 B663
81 B663 140401             BSET   PORTB GREEN      Grn LED ON
```

Listing 1. MCU-to-MCU Duplicator Program

```
 82 B666
 83 B666 150482   DUNPRG     BCLR   PORTB (RESET+RED) Red OFF, apply reset
 84 B669 20FE                BRA    *                 Done so just hang
 85 B66B
 86               ************************************************
 87               * Subroutine to get & send an SCI char.  Also
 88               *  advances pointer (X).
 89               ************************************************
 90 B66B A600     SEND1      LDAA   0,X               Get a character
 91 B66D 132E80FC TRDYLP     BRCLR  SCSR TDRE TRDYLP Wait for TDRE
 92 B671 972F                STAA   SCDR              Send character
 93 B673 08                  INX                     Advance pointer
 94 B674 39                  RTS                     ** Return **
 95
 96               ************************************************
 97               * Program to be bootloaded to target '711E9
 98               ************************************************
 99 B675 8604     BLPROG     LDAA   #$04              Pattern for DWOM off, no SPI
100 B677 B71028              STAA   $1028             Turns off DWOM in target MCU
101               * NOTE: Can't use direct addressing in target MCU because
102               *       regs are located at $1000.
103 B67A 7EBF00              JMP    PROGRAM           Jumps to EPROM prog routine
104 B67D          ENDBPR     EQU    *

Symbol Table:
Symbol Name     Value  Def.#  Line Number Cross Reference

BEGIN            B600 *00029
BLLOOP           B616 *00038  00040
BLPROG           B675 *00099  00037
DATALP           B648 *00068  00079
DLYLP            B620 *00046  00047
DLYLP2           B637 *00059  00063
DUNPRG           B666 *00083  00076
ENDBPR           B67D *00104  00039
EPSTRT           D000 *00023  00055 00066
GREEN            0001 *00015  00075 00081
INIT             103D *00009  00029
PORTB            0004 *00011  00033 00058 00061 00075 00081 00083
PORTE            000A *00016  00059
PROGRAM          BF00 *00022  00103
RDRF             0020 *00020  00034 00053 00071
RED              0002 *00014  00058 00061 00075 00083
RESET            0080 *00013  00032 00083
SCDR             002F *00021  00036 00049 00054 00072 00092
SCSR             002E *00017  00034 00048 00053 00071 00091
SEND1            B66B *00090  00038 00067 00070
SPCR             0028 *00010  00031
TDRE             0080 *00019  00091
TRDYLP           B66D *00091  00091
VERF             B64F *00071  00069 00071
VERFOK           B65F *00078  00074
WT4BRK           B60B *00034  00034
WT4FF            B627 *00053  00053
WT4VPP           B630 *00057  00060
```

```
              Errors: None
              Labels: 28
Last Program Address: $B67C
Last Storage Address: $0000
       Program Bytes: $007D  125
       Storage Bytes: $0000  0
```

Driving Boot Mode from a Personal Computer

In this example, a personal computer is used as the host to drive the bootloader of an MC68HC711E9. An M68HC11 EVBU is used for the target MC68HC711E9. A large program is transferred from the personal computer into the EPROM of the target MC68HC711E9.

Hardware

Figure 7 shows a small modification to the EVBU to accommodate the 12-volt (nominal) EPROM programming voltage. The $\overline{\text{XIRQ}}$ pin is connected to a pullup resistor, two jumpers, and the 60-pin connectors, P4 and P5. The object of the modification is to isolate the $\overline{\text{XIRQ}}$ pin and then connect it to the programming power supply. Carefully cut the trace on the solder side of the EVBU as indicated in Figure 7. This disconnects the pullup resistor RN1 D from $\overline{\text{XIRQ}}$ but leaves P4–18, P5–18, and jumpers J7 and J14 connected so the EVBU can still be used for other purposes after programming is done. Remove any fabricated jumpers from J7 and J14. The EVBU normally has a jumper at J7 to support the trace function

Figure 8 shows a small circuit that is added to the wire-wrap area of the EVBU. The 3-terminal jumper allows the $\overline{\text{XIRQ}}$ line to be connected to either the programming power supply or to a substitute pullup resistor for $\overline{\text{XIRQ}}$. The 100-ohm resistor is a current limiter to protect the 12-volt input of the MCU. The resistor and LED connected to P5 pin 9 (port C bit 0) is an optional indicator that lights when programming is complete.

Software

BASIC was chosen as the programming language due to its readability and availability in parallel versions on both the IBM® PC and the Macintosh®. The program demonstrates several programming techniques for use with an M68HC11 and is not necessarily intended to be a finished, commercial program. For example, there is little error checking, and the user interface is elementary. A complete listing of the BASIC program is included in Listing 2. BASIC Program for Personal Computer with moderate comments. The following paragraphs include a more detailed discussion of the program as it pertains to communicating with and programming the target MC68HC711E9. Lines 25–45 initialize and define the variables and array used in the program. Changes to this section would allow for other programs to be downloaded.

® IBM is a registered trademark of International Business Machines.
® Macintosh is a registered trademark of Apple Computers, Inc.

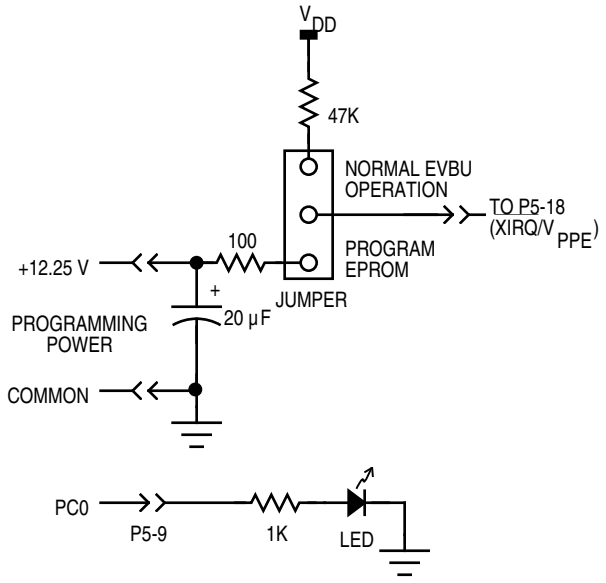


Figure 8. PC-to-MCU Programming Circuit

Lines 50–95 read in the small bootloader from DATA statements at the end of the listing. The source code for this bootloader is presented in the DATA statements. The bootloaded code makes port C bit 0 low, initializes the X and Y registers for use by the EPROM programming utility routine contained in the boot ROM, and then jumps to that routine. The hexadecimal values read in from the DATA statements are converted to binary values by a subroutine. The binary values are then saved as one string (BOOTCODE$).

The next long section of code (lines 97–1250) reads in the S records from an external disk file (in this case, BUF34.S19), converts them to integer, and saves them in an array. The techniques used in this section show how to convert ASCII S records to binary form that can be sent (bootloaded) to an M68HC11.

This S-record translator only looks for the S1 records that contain the actual object code. All other S-record types are ignored.

When an S1 record is found (lines 1000–1024), the next two characters form the hex byte giving the number of hex bytes to follow. This byte is converted to integer by the same subroutine that converted the bootloaded code from the DATA statements. This BYTECOUNT is adjusted by subtracting 3, which accounts for the address and checksum bytes and leaves just the number of object-code bytes in the record.

Starting at line 1100, the 2-byte (4-character) starting address is converted to decimal. This address is the starting address for the object code bytes to follow. An index into the CODE% array is formed by subtracting the base address initialized at the start of the program from the starting address for this S record.

A FOR-NEXT loop starting at line 1130 converts the object code bytes to decimal and saves them in the CODE% array. When all the object code bytes have been converted from the current S record, the program loops back to find the next S1 record.

A problem arose with the BASIC programming technique used. The draft versions of this program tried saving the object code bytes directly as binary in a string array. This caused "Out of Memory" or "Out of String Space" errors on both a 2-Mbyte Macintosh and a 640-Kbyte PC. The solution was to make the array an integer array and perform the integer-to-binary conversion on each byte as it is sent to the target part.

The one compromise made to accommodate both Macintosh and PC versions of BASIC is in lines 1500 and 1505. Use line 1500 and comment out line 1505 if the program is to be run on a Macintosh, and, conversely, use line 1505 and comment out line 1500 if a PC is used.

After the COM port is opened, the code to be bootloaded is modified by adding the $FF to the start of the string. $FF synchronizes the bootloader in the MC68HC711E9 to 1200 baud. The entire string is simply sent to the COM port by PRINTing the string. This is possible since the string is actually queued in BASIC's COM buffer, and the operating system takes care of sending the bytes out one at a time. The M68HC11 echoes the data received for verification. No automatic verification is provided, though the data is printed to the screen for manual verification.

Once the MCU has received this bootloaded code, the bootloader automatically jumps to it. The small bootloaded program in turn includes a jump to the EPROM programming routine in the boot ROM.

Refer to the previous explanation of the EPROM Programming Utility for the following discussion. The host system sends the first byte to be programmed through the COM port to the SCI of the MCU. The SCI port on the MCU buffers one byte while receiving another byte, increasing the throughput of the EPROM programming operation by sending the second byte while the first is being programmed.

When the first byte has been programmed, the MCU reads the EPROM location and sends the result back to the host system. The host then compares what was actually programmed to what was originally sent. A message indicating which byte is being verified is displayed in the lower half of the screen. If there is an error, it is displayed at the top of the screen.

As soon as the first byte is verified, the third byte is sent. In the meantime, the MCU has already started programming the second byte. This process of verifying and queueing a byte continues until the host finishes sending data. If the programming is completely successful, no error messages will have been displayed at the top of the screen. Subroutines follow the end of the program to handle some of the repetitive tasks. These routines are short, and the commenting in the source code should be sufficient explanation.

Modifications

This example programmed version 3.4 of the BUFFALO monitor into the EPROM of an MC68HC711E9; the changes to the BASIC program to download some other program are minor.

The necessary changes are:

1. In line 30, the length of the program to be downloaded must be assigned to the variable CODESIZE%.
2. Also in line 30, the starting address of the program is assigned to the variable ADRSTART.
3. In line 9570, the start address of the program is stored in the third and fourth items in that DATA statement in hexadecimal.
4. If any changes are made to the number of bytes in the boot code in the DATA statements in lines 9500–9580, then the new count must be set in the variable "BOOTCOUNT" in line 25.

Operation

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

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

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

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

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

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

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

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

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

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

® MS-DOS is a registered trademark of Microsoft Corporation in the United States and oth175190er countries.

Listing 2. BASIC Program for Personal Computer

```
1 ' ********************************************************************
2 ' *
3 ' *    E9BUF.BAS - A PROGRAM TO DEMONSTRATE THE USE OF THE BOOT MODE
4 ' *                   ON THE HC11 BY PROGRAMMING AN HC711E9 WITH
5 ' *                   BUFFALO 3.4
6 ' *
7 ' *                REQUIRES THAT THE S-RECORDS FOR BUFFALO (BUF34.S19)
8 ' *                   BE AVAILABLE IN THE SAME DIRECTORY OR FOLDER
9 ' *
10 '*                THIS PROGRAM HAS BEEN RUN BOTH ON A MS-DOS COMPUTER
11 '*                   USING QUICKBASIC 4.5 AND ON A MACINTOSH USING
12 '*                   QUICKBASIC 1.0.
14 '*
15 '********************************************************************
25 H$ = "0123456789ABCDEF"      'STRING TO USE FOR HEX CONVERSIONS
30 DEFINT B, I: CODESIZE% = 8192: ADRSTART= 57344!
35 BOOTCOUNT = 25               'NUMBER OF BYTES IN BOOT CODE
40 DIM CODE%(CODESIZE%)         'BUFFALO 3.4 IS 8K BYTES LONG
45 BOOTCODE$ = ""               'INITIALIZE BOOTCODE$ TO NULL
49 REM ***** READ IN AND SAVE THE CODE TO BE BOOT LOADED *****
50 FOR I = 1 TO BOOTCOUNT       '# OF BYTES IN BOOT CODE
55 READ Q$
60 A$ = MID$(Q$, 1, 1)
65 GOSUB 7000                   'CONVERTS HEX DIGIT TO DECIMAL
70 TEMP = 16 * X                'HANG ON TO UPPER DIGIT
75 A$ = MID$(Q$, 2, 1)
80 GOSUB 7000
85 TEMP = TEMP + X
90 BOOTCODE$ = BOOTCODE$ + CHR$(TEMP)    'BUILD BOOT CODE
95 NEXT I
96 REM ***** S-RECORD CONVERSION STARTS HERE *****
97 FILNAM$="BUF34.S19"          'DEFAULT FILE NAME FOR S-RECORDS
100 CLS
105 PRINT "Filename.ext of S-record file to be downloaded (";FILNAM$;") ";
107 INPUT Q$
110 IF Q$<>"" THEN FILNAM$=Q$
120 OPEN FILNAM$ FOR INPUT AS #1
130 PRINT : PRINT "Converting "; FILNAM$; " to binary..."
999 REM ***** SCANS FOR 'S1' RECORDS *****
1000 GOSUB 6000                  'GET 1 CHARACTER FROM INPUT FILE
1010 IF FLAG THEN 1250           'FLAG IS EOF FLAG FROM SUBROUTINE
1020 IF A$ <> "S" THEN 1000
1022 GOSUB 6000
1024 IF A$ <> "1" THEN 1000
1029 REM ***** S1 RECORD FOUND, NEXT 2 HEX DIGITS ARE THE BYTE COUNT *****
1030 GOSUB 6000
1040 GOSUB 7000                  'RETURNS DECIMAL IN X
1050 BYTECOUNT = 16 * X          'ADJUST FOR HIGH NIBBLE
1060 GOSUB 6000
1070 GOSUB 7000
1080 BYTECOUNT = BYTECOUNT + X   'ADD LOW NIBBLE
```

Listing 2. BASIC Program for Personal Computer

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

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

```
8491 '*         DECIMAL TO HEX CONVERSION
8492 '*                 INPUT:  K - INTEGER TO BE CONVERTED
8493 '*                 OUTPUT: HX$ - TWO CHARACTER STRING WITH HEX CONVERSION
8494 '*************************************************************************
8500 IF K > 255 THEN HX$="Too big":GOTO 8530
8510 HX$=MID$(H$,K\16+1,1)            'UPPER NIBBLE
8520 HX$=HX$+MID$(H$,(K MOD 16)+1,1)  'LOWER NIBBLE
8530 RETURN
9499 '******************** BOOT CODE ***************************************
9500 DATA 86, 23                 'LDAA   #$23
9510 DATA B7, 10, 02             'STAA   OPT2    make port C wire or
9520 DATA 86, FE                 'LDAA   #$FE
9530 DATA B7, 10, 03             'STAA   PORTC   light 1 LED on port C bit 0
9540 DATA C6, FF                 'LDAB   #$FF
9550 DATA F7, 10, 07             'STAB   DDRC    make port C outputs
9560 DATA CE, 0F, A0             'LDX    #4000   2msec at 2MHz
9570 DATA 18, CE, E0, 00         'LDY    #$E000  Start of BUFFALO 3.4
9580 DATA 7E, BF, 00             'JMP    $BF00   EPROM routine start address
9590 '*************************************************************************
```

Common Bootstrap Mode Problems

It is not unusual for a user to encounter problems with bootstrap mode because it is new to many users. By knowing some of the common difficulties, the user can avoid them or at least recognize and quickly correct them.

Reset Conditions vs. Conditions as Bootloaded Program Starts

It is common to confuse the reset state of systems and control bits with the state of these systems and control bits when a bootloaded program in RAM starts.

Between these times, the bootloader program is executed, which changes the states of some systems and control bits:

- The SCI system is initialized and turned on (Rx and Tx).
- The SCI system has control of the PD0 and PD1 pins.
- Port D outputs are configured for wire-OR operation.
- The stack pointer is initialized to the top of RAM.
- Time has passed (two or more SCI character times).
- Timer has advanced from its reset count value.

Users also forget that bootstrap mode is a special mode. Thus, privileged control bits are accessible, and write protection for some registers is not in effect. The bootstrap ROM is in the memory map. The DISR bit in the TEST1 control register is set, which disables resets from the COP and clock monitor systems.

Since bootstrap is a special mode, these conditions can be changed by software. The bus can even be switched from single-chip mode to expanded mode to gain access to external memories and peripherals.

Table 2. Summary of Boot-ROM-Related Features

MCU Part	BOOT ROM Revision (@$BFD1)	Mask Set I.D. (@$BFD2,3)	MCU Type I.D. (@$BFD4,5)	Security	Download Length	JMP on BRK or $00(1)	JMP to RAM(2)	Default RAM Location	PROGRAM(3) and UPLOAD(4) Utility	Notes
MC68HC11A0	—	—	Mask set #	—	256	$B600	$0000	$0000–FF	—	(5)
MC68HC11A1	—	—	Mask set #	—	256	$B600	$0000	$0000–FF	—	(5)
MC68HC11A8	—	—	Mask set #	—	256	$B600	$0000	$0000–FF	—	(5)
MC68SEC11A8	—	—	Mask set #	Yes	256	$B600	$0000	$0000–FF	—	(5)
MC68HC11D3	$00	ROM I.D. #	$11D3	—	0–192	$F000–ROM	—	$0040–FF	—	(6)
MC68HC711D3	$42(B)	$0000	$71D3	—	0–192	$F000–EPROM	—	$0040–FF	Yes	(6)
MC68HC811E2	—	$0000	$E2E2	—	256	$B600	$0000	$0000–FF	—	(5)
MC68SEC811E2	—	—	$E25C	Yes	256	$B600	$0000	$0000–FF	—	(5)
MC68HC11E0	—	ROM I.D. #	$E9E9	—	0–512	$B600	—	$0000–1FF	—	(5)
MC68HC11E1	—	ROM I.D. #	$E9E9	—	0–512	$B600	—	$0000–1FF	—	(5)
MC68HC11E9	—	ROM I.D. #	$E9E9	—	0–512	$B600	—	$0000–1FF	—	(5)
MC68SEC11E9	—	ROM I.D. #	$E95C	Yes	0–512	$B600	—	$0000–1FF	—	(5)
MC68HC711E9	$41(A)	$0000	$71E9	—	0–512	$B600	—	$0000–1FF	Yes	
MC68HC11F1	$42(B)	$0000	$F1F1	—	0–1024	$FE00	—	$0000–3FF	—	(6), (7)
MC68HC11K4	$30(0)	ROM I.D. #	$044B	—	0–768	$0D80	—	$0080–37F	—	(6), (8)
MC68HC711K4	$42(B)	$0000	$744B	—	0–768	$0D80	—	$0080–37F	Yes	(6), (8)

NOTES:

1. By sending $00 or a break as the first SCI character after reset in bootstrap mode, a jump (JMP) is executed to the address in this table rather than doing a download. Unless otherwise noted, this address is the start of EEPROM. Tying RxD to TxD and using a pullup resistor from TxD to V_{DD} will cause the SCI to see a break as the first received character.
2. If $55 is received as the first character after reset in bootstrap mode, a jump (JMP) is executed to the start of on-chip RAM rather than doing a download. This $55 character must be sent at the default baud rate (7812 baud @ E = 2 MHz). For devices with variable-length download, the same effect can be achieved by sending $FF and no other SCI characters. After four SCI character times, the download terminates, and a jump (JMP) to the start of RAM is executed.
   The jump to RAM feature is only useful if the RAM was previously loaded with a meaningful program.
3. A callable utility subroutine is included in the bootstrap ROM of the indicated versions to program bytes of on-chip EPROM with data received via the SCI.
4. A callable utility subroutine is included in the bootstrap ROM of the indicated versions to upload contents of on-chip memory to a host computer via the SCI.
5. The complete listing for this bootstrap ROM may be found in the *M68HC11 Reference Manual*, Freescale document order number M68HC11RM/AD.
6. The complete listing for this bootstrap ROM is available in the freeware area of the Freescale Web site.
7. Due to the extra program space needed for EEPROM security on this device, there are no pseudo-vectors for SCI, SPI, PAIF, PAOVF, TOF, OC5F, or OC4F interrupts.
8. This bootloader extends the automatic software detection of baud rates to include 9600 baud at 2-MHz E-clock rate.

Connecting RxD to V_{SS} Does Not Cause the SCI to Receive a Break

To force an immediate jump to the start of EEPROM, the bootstrap firmware looks for the first received character to be $00 (or break). The data reception logic in the SCI looks for a 1-to-0 transition on the RxD pin to synchronize to the beginning of a receive character. If the RxD pin is tied to ground, no 1-to-0 transition occurs. The SCI transmitter sends a break character when the bootloader firmware starts, and this break character can be fed back to the RxD pin to cause the jump to EEPROM. Since TxD is configured as an open-drain output, a pullup resistor is required.

$FF Character Is Required before Loading into RAM

The initial character (usually $FF) that sets the download baud rate is often forgotten.

Original M68HC11 Versions Required Exactly 256 Bytes to be Downloaded to RAM

Even users that know about the 256 bytes of download data sometimes forget the initial $FF that makes the total number of bytes required for the entire download operation equal to 256 + 1 or 257 bytes.

Variable-Length Download

When on-chip RAM surpassed 256 bytes, the time required to serially load this many characters became more significant. The variable-length download feature allows shorter programs to be loaded without sacrificing compatibility with earlier fixed-length download versions of the bootloader. The end of a download is indicated by an idle RxD line for at least four character times. If a personal computer is being used to send the download data to the MCU, there can be problems keeping characters close enough together to avoid tripping the end-of-download detect mechanism. Using 1200 as the baud rate rather than the faster default rate may help this problem.

Assemblers often produce S-record encoded programs which must be converted to binary before bootloading them to the MCU. The process of reading S-record data from a file and translating it to binary can be slow, depending on the personal computer and the programming language used for the translation. One strategy that can be used to overcome this problem is to translate the file into binary and store it into a RAM array before starting the download process. Data can then be read and downloaded without the translation or file-read delays.

The end-of-download mechanism goes into effect when the initial $FF is received to set the baud rate. Any amount of time may pass between reset and when the $FF is sent to start the download process.

EPROM/OTP Versions of M68HC11 Have an EPROM Emulation Mode

The conditions that configure the MCU for EPROM emulation mode are essentially the same as those for resetting the MCU in bootstrap mode. While $\overline{\text{RESET}}$ is low and mode select pins are configured for bootstrap mode (low), the MCU is configured for EPROM emulation mode.

The port pins that are used for EPROM data I/O lines may be inputs or outputs, depending on the pin that is emulating the EPROM output enable pin ($\overline{\text{OE}}$). To make these data pins appear as high-impedance inputs as they would on a non-EPROM part in reset, connect the $\overline{\text{PB7}}$/($\overline{\text{OE}}$) pin to a pullup resistor.

Bootloading a Program to Performa ROM Checksum

The bootloader ROM must be turned off before performing the checksum program. To remove the boot ROM from the memory map, clear the RBOOT bit in the HPRIO register. This is normally a write-protected bit that is 0, but in bootstrap mode it is reset to 1 and can be written. If the boot ROM is not disabled, the checksum routine will read the contents of the boot ROM rather than the user's mask ROM or EPROM at the same addresses.

Inherent Delays Caused by Double Buffering of SCI Data

This problem is troublesome in cases where one MCU is bootloading to another MCU.

Because of transmitter double buffering, there may be one character in the serial shifter as a new character is written into the transmit data register. In cases such as downloading in which this 2-character pipeline is kept full, a 2-character time delay occurs between when a character is written to the transmit data register and when that character finishes transmitting. A little more than one more character time delay occurs between the target MCU receiving the character and echoing it back. If the master MCU waits for the echo of each downloaded character before sending the next one, the download process takes about twice as long as it would if transmission is treated as a separate process or if verify data is ignored.

Boot ROM Variations

Different versions of the M68HC11 have different versions of the bootstrap ROM program. Table 3 summarizes the features of the boot ROMs in 16 members of the M68HC11 Family.

The boot ROMs for the MC68HC11F1, the MC68HC711K4, and the MC68HC11K4 allow additional choices of baud rates for bootloader communications. For the three new baud rates, the first character used to determine the baud rate is not $FF as it was in earlier M68HC11s. The intercharacter delay that terminates the variable-length download is also different for these new baud rates. Table 3 shows the synchronization characters, delay times, and baud rates as they relate to E-clock frequency.

Commented Boot ROM Listing

Listing 3. MC68HC711E9 Bootloader ROM contains a complete commented listing of the boot ROM program in the MC68HC711E9 version of the M68HC11. Other versions can be found in **Appendix B** of the *M68HC11 Reference Manual*.

Table 3. Bootloader Baud Rates

Sync Character	Timeout Delay	Baud Rates at E Clock =					
		2 MHz	2.1 MHz	3 MHz	3.15 MHz	4 MHz	4.2 MHz
$FF	4 characters	7812	8192	11,718	12,288	15,624	16,838
$FF	4 characters	1200	1260	1800	1890	2400	2520
$F0	4.9 characters	9600	10,080	14,400	15,120	19,200	20,160
$FD	17.3 characters	5208	5461	7812	8192	10,416	10,922
$FD	13 characters	3906	4096	5859	6144	7812	8192

Listing 3. MC68HC711E9 Bootloader ROM

```
                ****************************************************
                * BOOTLOADER FIRMWARE FOR 68HC711E9 - 21 Aug 89
                ****************************************************
                * Features of this bootloader are...
                *
                * Auto baud select between 7812.5 and 1200 (8 MHz)
                * 0 - 512 byte variable length download
                * Jump to EEPROM at $B600 if 1st download byte = $00
                * PROGRAM - Utility subroutine to program EPROM
                * UPLOAD - Utility subroutine to dump memory to host
                * Mask I.D. at $BFD4 = $71E9
                ****************************************************
                * Revision A -
                *
                * Fixed bug in PROGRAM routine where the first byte
                * programmed into the EPROM was not transmitted for
                * verify.
                * Also added to PROGRAM routine a skip of bytes
                * which were already programmed to the value desired.
                *
                * This new version allows variable length download
                * by quitting reception of characters when an idle
                * of at least four character times occurs
                *
                ****************************************************

                * EQUATES FOR USE WITH INDEX OFFSET = $1000
                *
0008            PORTD     EQU     $08
000E            TCNT      EQU     $0E
0016            TOC1      EQU     $16
0023            TFLG1     EQU     $23
                * BIT EQUATES FOR TFLG1
0080            OC1F      EQU     $80
                *
0028            SPCR      EQU     $28             (FOR DWOM BIT)
002B            BAUD      EQU     $2B
002D            SCCR2     EQU     $2D
002E            SCSR      EQU     $2E
002F            SCDAT     EQU     $2F
003B            PPROG     EQU     $3B
                * BIT EQUATES FOR PPROG
0020            ELAT      EQU     $20
0001            EPGM      EQU     $01
                *


                * MEMORY CONFIGURATION EQUATES
                *
B600            EEPMSTR   EQU     $B600           Start of EEPROM
B7FF            EEPMEND   EQU     $B7FF           End of EEPROM
                *
```

```
 52 D000           EPRMSTR  EQU    $D000          Start of EPROM
 53 FFFF           EPRMEND  EQU    $FFFF          End of EPROM
 54                *
 55 0000           RAMSTR   EQU    $0000
 56 01FF           RAMEND   EQU    $01FF
 57
 58                * DELAY CONSTANTS
 59                *
 60 0DB0           DELAYS   EQU    3504           Delay at slow baud
 61 021B           DELAYF   EQU    539            Delay at fast baud
 62                *
 63 1068           PROGDEL  EQU    4200           2 ms programming delay
 64                *                       At 2.1 MHz
 65
 66                ************************************************
 67 BF00                     ORG    $BF00
 68                ************************************************
 69
 70                * Next two instructions provide a predictable place
 71                * to call PROGRAM and UPLOAD even if the routines
 72                * change size in future versions.
 73                *
 74 BF00 7EBF13    PROGRAM  JMP    PRGROUT        EPROM programming utility
 75 BF03           UPLOAD   EQU    *              Upload utility
 76
 77                ************************************************
 78                * UPLOAD - Utility subroutine to send data from
 79                * inside the MCU to the host via the SCI interface.
 80                * Prior to calling UPLOAD set baud rate, turn on SCI
 81                * and set Y=first address to upload.
 82                * Bootloader leaves baud set, SCI enabled, and
 83                * Y pointing at EPROM start ($D000) so these default
 84                * values do not have to be changed typically.
 85                * Consecutive locations are sent via SCI in an
 86                * infinite loop. Reset stops the upload process.
 87                ************************************************
 88 BF03 CE1000             LDX    #$1000         Point to internal registers
 89 BF06 18A600   UPLOOP    LDAA   0,Y            Read byte
 90 BF09 1F2E80FC           BRCLR  SCSR,X $80 *   Wait for TDRE
 91 BF0D A72F               STAA   SCDAT,X        Send it
 92 BF0F 1808               INY
 93 BF11 20F3               BRA    UPLOOP         Next...
 94
 95                ************************************************
 96                * PROGRAM - Utility subroutine to program EPROM.
 97                * Prior to calling PROGRAM set baud rate, turn on SCI
 98                * set X=2ms prog delay constant, and set Y=first
 99                * address to program. SP must point to RAM.
100                * Bootloader leaves baud set, SCI enabled, X=4200
101                * and Y pointing at EPROM start ($D000) so these
102                * default values don't have to be changed typically.
103                * Delay constant in X should be equivalent to 2 ms
104                *  at 2.1 MHz X=4200; at 1 MHz X=2000.
105                * An external voltage source is required for EPROM
106                * programming.
```

Listing 3. MC68HC711E9 Bootloader ROM

```
107                * This routine uses 2 bytes of stack space
108                * Routine does not return. Reset to exit.
109                ************************************************
110 BF13           PRGROUT  EQU    *
111 BF13 3C                PSHX                    Save program delay constant
112 BF14 CE1000            LDX    #$1000           Point to internal registers
113 BF17
114                * Send $FF to indicate ready for program data
115
116 BF17 1F2E80FC          BRCLR  SCSR,X $80 *     Wait for TDRE
117 BF1B 86FF              LDAA   #$FF
118 BF1D A72F              STAA   SCDAT,X
119
120 BF1F           WAIT1    EQU    *
121 BF1F 1F2E20FC          BRCLR  SCSR,X $20 *     Wait for RDRF
122 BF23 E62F              LDAB   SCDAT,X          Get received byte
123 BF25 18E100            CMPB   $0,Y             See if already programmed
124 BF28 271D              BEQ    DONEIT           If so, skip prog cycle
125 BF2A 8620              LDAA   #ELAT            Put EPROM in prog mode
126 BF2C A73B              STAA   PPROG,X
127 BF2E 18E700            STAB   0,Y              Write the data
128 BF31 8621              LDAA   #ELAT+EPGM
129 BF33 A73B              STAA   PPROG,X          Turn on prog voltage
130 BF35 32                PULA                    Pull delay constant
131 BF36 33                PULB                    into D-reg
132 BF37 37                PSHB                    But also keep delay
133 BF38 36                PSHA                    keep delay on stack
134 BF39 E30E              ADDD   TCNT,X           Delay const + present TCNT
135 BF3B ED16              STD    TOC1,X           Schedule OC1 (2ms delay)
136 BF3D 8680              LDAA   #OC1F
137 BF3F A723              STAA   TFLG1,X          Clear any previous flag
138
139 BF41 1F2380FC          BRCLR  TFLG1,X OC1F * Wait for delay to expire
140 BF45 6F3B              CLR    PPROG,X          Turn off prog voltage
141                *
142 BF47           DONEIT   EQU    *
143 BF47 1F2E80FC          BRCLR  SCSR,X $80 *     Wait for TDRE
144 BF4B 18A600            LDAA   $0,Y             Read from EPROM and...
145 BF4E A72F              STAA   SCDAT,X          Xmit for verify
146 BF50 1808              INY                     Point at next location
147 BF52 20CB              BRA    WAIT1            Back to top for next
148                * Loops indefinitely as long as more data sent.
149
150                ************************************************
151                * Main bootloader starts here
152                ************************************************
153                * RESET vector points to here
154
155 BF54           BEGIN    EQU    *
156 BF54 8E01FF            LDS    #RAMEND          Initialize stack pntr
157 BF57 CE1000            LDX    #$1000           Point at internal regs
158 BF5A 1C2820            BSET   SPCR,X $20       Select port D wire-OR mode
159 BF5D CCA20C            LDD    #$A20C           BAUD in A, SCCR2 in B
160 BF60 A72B              STAA   BAUD,X           SCPx = ÷4, SCRx = ÷4
161                * Writing 1 to MSB of BAUD resets count chain
```

```
162 BF62 E72D              STAB   SCCR2,X        Rx and Tx Enabled
163 BF64 CC021B            LDD    #DELAYF        Delay for fast baud rate
164 BF67 ED16              STD    TOC1,X         Set as default delay
165
166                * Send BREAK to signal ready for download
167 BF69 1C2D01            BSET   SCCR2,X $01    Set send break bit
168 BF6C 1E0801FC          BRSET  PORTD,X $01 *  Wait for RxD pin to go low
169 BF70 1D2D01            BCLR   SCCR2,X $01    Clear send break bit
170 BF73
171 BF73 1F2E20FC          BRCLR  SCSR,X $20 *   Wait for RDRF
172 BF77 A62F              LDAA   SCDAT,X        Read data
173                * Data will be $00 if BREAK OR $00 received
174 BF79 2603              BNE    NOTZERO        Bypass JMP if not 0
175 BF7B 7EB600            JMP    EEPMSTR        Jump to EEPROM if it was 0
176 BF7E           NOTZERO EQU    *
177 BF7E 81FF              CMPA   #$FF           $FF will be seen as $FF
178 BF80 2708              BEQ    BAUDOK         If baud was correct
179                * Or else change to ÷104 (÷13 & ÷8) 1200 @ 2MHZ
180 BF82 1C2B33            BSET   BAUD,X $33     Works because $22 -> $33
181 BF85 CC0DB0            LDD    #DELAYS        And switch to slower...
182 BF88 ED16              STD    TOC1,X         delay constant
183 BF8A           BAUDOK  EQU    *
184 BF8A 18CE0000          LDY    #RAMSTR        Point at start of RAM
185
186 BF8E           WAIT    EQU    *
187 BF8E EC16              LDD    TOC1,X         Move delay constant to D
188 BF90           WTLOOP  EQU    *
189 BF90 1E2E2007          BRSET  SCSR,X $20 NEWONE  Exit loop if RDRF set
190 BF94 8F                XGDX                  Swap delay count to X
191 BF95 09                DEX                   Decrement count
192 BF96 8F                XGDX                  Swap back to D
193 BF97 26F7              BNE    WTLOOP         Loop if not timed out
194 BF99 200F              BRA    STAR           Quit download on timeout
195
196 BF9B           NEWONE  EQU    *
197 BF9B A62F              LDAA   SCDAT,X        Get received data
198 BF9D 18A700            STAA   $00,Y          Store to next RAM location
199 BFA0 A72F              STAA   SCDAT,X        Transmit it for handshake
200 BFA2 1808              INY                   Point at next RAM location
201 BFA4 188C0200          CPY    #RAMEND+1      See if past end
202 BFA8 26E4              BNE    WAIT           If not, Get another
203
204 BFAA           STAR    EQU    *
205 BFAA CE1068            LDX    #PROGDEL       Init X with programming delay
206 BFAD 18CED000          LDY    #EPRMSTR       Init Y with EPROM start addr
207 BFB1 7E0000            JMP    RAMSTR         ** EXIT to start of RAM **
208 BFB4
209                ************************************************
210                * Block fill unused bytes with zeros
211
212 BFB4 000000000000        BSZ    $BFD1-*
           000000000000
           000000000000
           000000000000
           0000000000
```

Listing 3. MC68HC711E9 Bootloader ROM

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

Symbol Name     Value  Def.#  Line Number Cross Reference

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

```
NEWONE            BF9B *00196  00189
NOTZERO           BF7E *00176  00174
OC1F              0080 *00034  00136 00139
PORTD             0008 *00029  00168
PPROG             003B *00041  00126 00129 00140
PRGROUT           BF13 *00110  00074
PROGDEL           1068 *00063  00205
PROGRAM           BF00 *00074
RAMEND            01FF *00056  00156 00201
RAMSTR            0000 *00055  00184 00207
SCCR2             002D *00038  00162 00167 00169
SCDAT             002F *00040  00091 00118 00122 00145 00172 00197 00199
SCSR              002E *00039  00090 00116 00121 00143 00171 00189
SPCR              0028 *00036  00158
STAR              BFAA *00204  00194
TCNT              000E *00030  00134
TFLG1             0023 *00032  00137 00139
TOC1              0016 *00031  00135 00164 00182 00187
UPLOAD            BF03 *00075
UPLOOP            BF06 *00089  00093
WAIT              BF8E *00186  00202
WAIT1             BF1F *00120  00147
WTLOOP            BF90 *00188  00193

                  Errors: None
                  Labels: 35
     Last Program Address: $BFFF
     Last Storage Address: $0000
            Program Bytes: $0100  256
            Storage Bytes: $0000  0
```

Listing 3. MC68HC711E9 Bootloader ROM

Freescale Semiconductor
Engineering Bulletin

EB184
Rev. 0.1, 07/2005

Enabling the Security Feature on the MC68HC711E9 Devices with PCbug11 on the M68HC711E9PGMR

By Edgar Saenz
Austin, Texas

Introduction

The PCbug11 software, needed along with the M68HC711E9PGMR to program MC68HC711E9 devices, is available from the download section of the Microcontroller Worldwide Web site:

http://www.freescale.com

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

Some Freescale 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 Freescale Literature Distribution Center, as well as the Worldwide Web at http://www.freescale.com. The file is also on the software download system and is called pcbug11.pdf.

© Freescale Semiconductor, Inc., 2005. All rights reserved.

freescale semiconductor

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 PCbug11this 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 adressing 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 PCbug11command 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*, Freescale document number EB187.

Step 8

You are now ready to enable the security feature on the MCHC711E9.

At the PCbug11 command prompt type: MS 103F 05.

Step 9

After the programming operation is complete, verifyng the CONFIG on the MCHC711E9 is not possible because in bootstrap mode the default value is always forced.

Step 10

The part is now in secure mode and whatever code you loaded into EEPROM will be erased if you tried to bring the microcontroller up in either expanded mode or bootstrap mode.

NOTE

It is important to note that the microcontroller will work properly in secure mode only in single chip mode.

NOTE

If the part is placed in bootstrap or expanded, the code in EEPROM and RAM will be erased and the microcontroller cannot be reused. The security software will constantly read the NOSEC bit and lock the part.

Freescale Semiconductor
Engineering Bulletin

EB188
Rev. 0.1, 07/2005

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.freescale.com

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

Some Freescale 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 Freescale Literature http://www.freescale.com. The file is also on the software download system and is called pcbug11.pdf.

© Freescale Semiconductor, Inc., 2005. All rights reserved.

freescale semiconductor

To Execute the Program

Once you have obtained PCbug11, use this step-by-step procedure.

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 your MC68HC811E2 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 volts and ground to GND on the programmer board's power connector, P1. Applying voltage to the V_{PP} pin is not necessary.

Step 2

Apply power to the programmer board by moving the +5-volt switch to the ON position.

From a DOS command line prompt, start PCbug11 this way:

- C:\PCBUG11\> PCBUG11 –A PORT = 1 when the E9PGMR connected to COM1 or
- C:\PCBUG11\> PCBUG11 –A PORT = 2 when the E9PGMR connected to COM2

PCbug11only supports COM ports 1 and 2.

Step 3

PCbug11 defaults to base ten 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 MC68HC811E2 EEPROM.

At the PCbug11 command prompt, type: MS 1035 00

Step 5

PCbug11 defaults to a 512-byte EEPROM array located at $B600. This must be changed since the EEPROM is, by default, located at $F800 on the MC68HC811E2.

At the PCbug11 command prompt, type: EEPROM 0

Then type: EEPROM F800 FFFF
EEPROM 103F 103F

This assumes you have not relocated the EEPROM by previously reprogramming the upper 4 bits of the CONFIG register. But if you have done this and your S records reside in an address range other than $F800 to $FFFF, you will need to first relocate the EEPROM.

Step 6

Erase the CONFIG to allow programming of NOSEC bit (bit 3). It is also recommended to program the EEPROM at this point before programming the CONFIG register. Refer to the engineering bulletin *Programming MC68HC811E2 Devices with PCbug11 and the M68HC711E9PGMR*, Freescale document number EB184.

At the PCbug11command prompt, type: EEPROM ERASE BULK 103F

Step 7

You are now ready to enable the security feature on the MCHC811E2.

At the PCbug11 command prompt, type: MS 103F 05

The value $05 assumes the EEPROM is to be mapped from $0800 to $0FFF.

Step 8

After the programming operation is complete, verifying the CONFIG on the MCHC811E2 is not possible because in bootstrap mode the default value is always forced.

Step 9

The part is now in secure mode and whatever code you loaded into EEPROM will be erased if you tried to bring the microcontroller up in either expanded mode or bootstrap mode. The microcontroller will work properly in the secure mode only in single chip mode.

NOTE

If the part is placed in bootstrap mode or expanded mode, the code in EEPROM and RAM will be erased the microcontroller can be reused.

Freescale Semiconductor
Engineering Bulletin

EB296
Rev. 0.1, 07/2005

Programming MC68HC711E9 Devices with PCbug11 and the M68HC11EVBU

By John Bodnar
Austin, Texas

Introduction

The PCbug1software, needed along with the M68HC11EVBU to program MC68HC711E9 devices, is available from the download section of the Microcontroller Worldwide Web site

http://www.freescale.com

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

Some Freescale 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 Freescale Literature Distribution Center, as well as the Worldwide Web at http://www.freescale.com. The file is also on the software download system and is called pcbug11.pdf.

© Freescale Semiconductor, Inc., 2005. All rights reserved.

freescale semiconductor

Programming Procedure

Once you have obtained PCbug11, use this step-by-step procedure to program your MC68HC711E9 part.

Step 1

- Before applying power to the EVBU, remove the jumper from J7 and place it across J3 to ground the MODB pin.
- Place a jumper across J4 to ground the MODA pin. This will force the EVBU into special bootstrap mode on power up.
- Remove the resident MC68HC11E9 MCU from the EVBU.
- Place your MC68HC711E9 in the open socket with the notched corner of the part aligned with the notch on the PLCC socket.
- Connect the EVBU to one of your PC COM ports. Apply +5 volts to V_{DD} and ground to GND on the power connector of your EVBU.

Also take note of P4 connector pin 18. In step 5, you will connect a +12-volt (at most +12.5 volts) programming voltage through a 100-Ω current limiting resistor to the XIRQ pin. Do not connect this programming voltage until you are instructed to do so in step 5.

Step 2

- From a DOS command line prompt, start PCbug11 with
  - C:\PCBUG11\> PCBUG11 –E PORT = 1 with the EVBU connected to COM1
  - C:\PCBUG11\> PCBUG11 –E PORT = 2 with the EVBU connected to COM2

PCbug11 only supports COM ports 1 and 2. If you have made the proper connections and have a high quality cable, you should quickly get a PCbug11 command prompt. If you do receive a Comms fault error, check your 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

- You must declare the addresses of the EPROM array to PCbug11. To do this, type:
  EPROM D000 FFFF

Step 5

You are now ready to download your program into the EPROM.

- Connect +12 volts (at most +12.5 volts) through a 100-Ω current limiting resistor to P4 connector pin 18, the XIRQ* pin.
- At the PCbug11 command prompt type: LOADS C:\MYPROG\ISHERE.S19

Substitute the name of your program into the command above. Use a full path name if your program is not located in the same directory as PCbug11.

Step 6

After the programming operation is complete, PCbug11 will display this message

Total bytes loaded: $xxxx

Total bytes programmed: $yyyy

- You should now remove the programming voltage from P4 connector pin 18, the XIRQ* pin.
- Each ORG directive in your assembly language source will cause a pair of these lines to be generated. For this operation, $yyyy will be incremented by the size of each block of code programmed into the EPROM of the MC68HC711E9.
- PCbug11 will display the above message whether or not the programming operation was successful. As a precaution, you should have PCbug11 verify your code.
- At the PCbug11 command prompt type: VERF C:\MYPROG\ISHERE.S19

Substitute the name of your program into the command above. Use a full path name if your program is not located in the same directory as PCbug11.

If the verify operation fails, a list of addresses which did not program correctly is displayed. Should this occur, you probably need to erase your part more completely. To do so, allow the MC68HC711E9 to sit for at least 45 minutes under an ultraviolet light source. Attempt the programming operation again. If you have purchased devices in plastic packages (one-time programmable parts), you will need to try again with a new, unprogrammed device.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

RoHS-compliant and/or Pb- free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb- free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale.s Environmental Products program, go to http://www.freescale.com/epp.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2005. All rights reserved.

M68HC11E
Rev. 5.1, 07/2005

freescale™
semiconductor