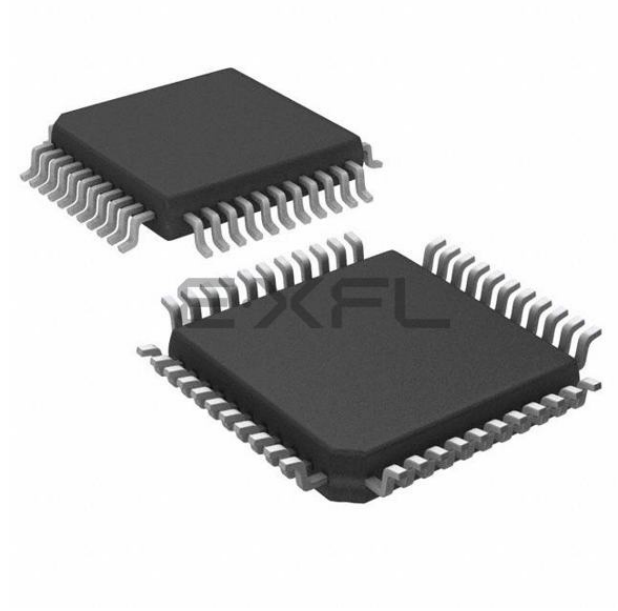


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	HC11
Core Size	8-Bit
Speed	2MHz
Connectivity	SCI, SPI
Peripherals	POR, WDT
Number of I/O	26
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	192 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-QFP
Supplier Device Package	44-QFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc11d0cfbe2

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.

Revision History

Date	Revision Level	Description	Page Number(s)
September, 2003	2	Reformatted to current publications standards	N/A
		Removed references to PROG mode.	Throughout
		Corrected pin assignments for: Figure 1-2. Pin Assignments for 40-Pin Plastic DIP Figure 1-3. Pin Assignments for 44-Pin PLCC Added Figure 1-4. Pin Assignments for 44-Pin QFP	 4 5 6
		1.9 Interrupt Request (IRQ) — Reworked description for clarity.	7
		2.4 Programmable Read-Only Memory (PROM) — Updated with additional data.	13
		Section 10. Ordering Information and Mechanical Specifications — Added mechanical specifications for 44-pin plastic quad flat pack (QFP).	133
		Added the following appendices: Appendix A. MC68HC11D3 and MC68HC11D0 Appendix B. MC68L11D0	 137 143
July, 2005	2.1	Updated to meet Freescale identity guidelines.	Throughout

Table 2-2. Bootstrap Mode Jump Vectors (Continued)

Address	Vector
00EB	Real-time interrupt
00EE	IRQ
00F1	XIRQ
00F4	SWI
00F7	Illegal opcode
00FA	COP fail
00FD	Clock monitor
BF00 (Boot)	Reset

2.2.4 Special Test Mode

This special expanded mode is primarily intended or production testing. The user can access a number of special test control bits in this mode. Reset and interrupt vectors are fetched externally from locations $BFC0–$BFFF. A switch can be made from this mode to other modes under program control.

2.3 Memory Map

Figure 2-1 illustrates the memory map for both normal modes of operation (single-chip and expanded-multiplexed), as well as for both special modes of operation (bootstrap and test).

- In the single-chip mode, the MCU does not generate external addresses. The internal memory locations are shown in the shaded areas, and the contents of these shaded areas are explained on the right side of the diagram.
- In expanded-multiplexed mode, the memory locations are basically the same as in the single-chip mode except that the memory locations between shaded areas are for externally addressed memory and I/O.
- The special bootstrap mode is similar to the single-chip mode, except that the bootstrap program ROM is located at memory locations $BF00–$BFFF, vectors included.
- The special test mode is similar to the expanded-multiplexed mode except the interrupt vectors are at external memory locations.

2.3.1 Control and Status Registers

Figure 2-2 is a representation of all 64 bytes of control and status registers, I/O and data registers, and reserved locations that make up the internal register block. This block may be mapped to any 4-K boundary in memory, but reset locates it at $0000–$003F. This mappability factor and the default starting addresses are indicated by the use of a bold **0** as the starting character of a register's address.

3.2.1 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.

3.2.2 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.

3.2.3 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 3.4 Opcodes and Operands for further information.

3.2.4 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 3-2 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, a 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.

3.2.5 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 3-1.

Table 3-1. Reset Vector Comparison

Mode	POR or $\overline{\text{RESET}}$ Pin	Clock Monitor	COP Watchdog
Normal	$FFFE, $FFFF	$FFFC, $FFFD	$FFFA, $FFFB
Test or boot	$BFFE, $BFFF	$BFFC, $FFFD	$BFFA, $FFFB

3.2.6 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 $\overline{\text{XIRQ}}$)
- One 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 3-2, which shows what condition codes are affected by a particular instruction.

3.2.6.1 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.

3.2.6.2 Overflow (V)

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

3.2.6.3 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.

Table 3-2. Instruction Set (Sheet 2 of 8)

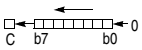
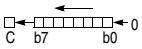
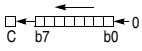
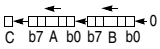
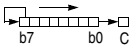
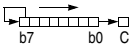
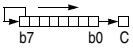
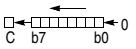
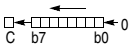
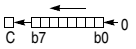
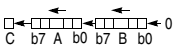
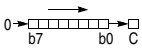
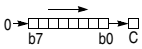
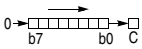
Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
ADDB (opr)	Add Memory to B	B + M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	CB DB FB EB 18 EB	ii dd hh ll ff ff	2 3 4 4 5	—	—	Δ	—	Δ	Δ	Δ	Δ
ADDD (opr)	Add 16-Bit to D	D + (M : M + 1) ⇒ D	IMM DIR EXT IND,X IND,Y	C3 D3 F3 E3 18 E3	jj kk dd hh ll ff ff	4 5 6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
ANDA (opr)	AND A with Memory	A • M ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	84 94 B4 A4 18 A4	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
ANDB (opr)	AND B with Memory	B • M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	C4 D4 F4 E4 18 E4	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	0	—
ASL (opr)	Arithmetic Shift Left		EXT IND,X IND,Y	78 68 18 68	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
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 IND,X IND,Y	77 67 18 67	hh ll ff ff	6 6 7	—	—	—	—	Δ	Δ	Δ	Δ
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 IND,X IND,Y	15 1D 18 1D	dd mm ff mm ff mm	6 7 8	—	—	—	—	Δ	Δ	0	—
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	—	—	—	—	—	—	—	—
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	—

Table 3-2. Instruction Set (Sheet 3 of 8)

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

Table 3-2. Instruction Set (Sheet 5 of 8)

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
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

6.5 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.

6.5.1 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.

6.5.2 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 — 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.

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

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

The SPI configuration determines the characteristics of a transfer in progress. For a master, a transfer begins when data is written to SPDR and ends when SPIF is set. For a slave with CPHA equal to zero, a transfer starts when $\overline{SS}$ goes low and ends when $\overline{SS}$ returns high. In this case, SPIF is set at the middle of the eighth SCK cycle when data is transferred from the shifter to the parallel data register, but the transfer is still in progress until $\overline{SS}$ goes high. For a slave with CPHA equal to one, transfer begins when the SCK line goes to its active level, which is the edge at the beginning of the first SCK cycle. The transfer ends in a slave in which CPHA equals one when SPIF is set. For a slave, after a byte transfer, SCK must be in inactive state for at least 2 E-clock cycles before the next byte transfer begins.

7.7 SPI Registers

The three SPI registers, SPCR, SPSR, and SPDR, provide control, status, and data storage functions. This sub-section provides a description of how these registers are organized.

7.7.1 SPI Control Register

Address: $0028

	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 7-3. SPI Control Register (SPCR)

SPIE — Serial Peripheral Interrupt Enable Bit

0 = SPI interrupt disabled
1 = SPI interrupt enabled

SPE — Serial Peripheral System Enable Bit

0 = SPI off
1 = SPI on

DWOM — Port D Wired-OR Mode Bit

DWOM affects all six port D pins.

0 = Normal CMOS outputs
1 = Open-drain outputs

MSTR — Master Mode Select Bit

0 = Slave mode
1 = Master mode

PR1 and PR0 — Timer Prescaler Select Bits

These bits are used to select the prescaler divide-by ratio. In normal modes, PR1 and PR0 can be written once only, and the write must be within 64 cycles after reset. Refer to Table 8-4 for specific timing values.

Table 8-4. Timer Prescale

PR1 and PR0	Prescaler
0 0	1
0 1	4
1 0	8
1 1	16

8.4.10 Timer Interrupt Flag 2 Register

The timer interrupt flag 2 register (TFLG2) bits 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.

Address: $0025

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TOF	RTIF	PAOVF	PAIF	0	0	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 8-15. 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 8.5 Real-Time Interrupt.

PAOVF — Pulse Accumulator Overflow Interrupt Flag

Refer to 8.7 Pulse Accumulator.

PAIF — Pulse Accumulator Input Edge Interrupt Flag

Refer to 8.7 Pulse Accumulator.

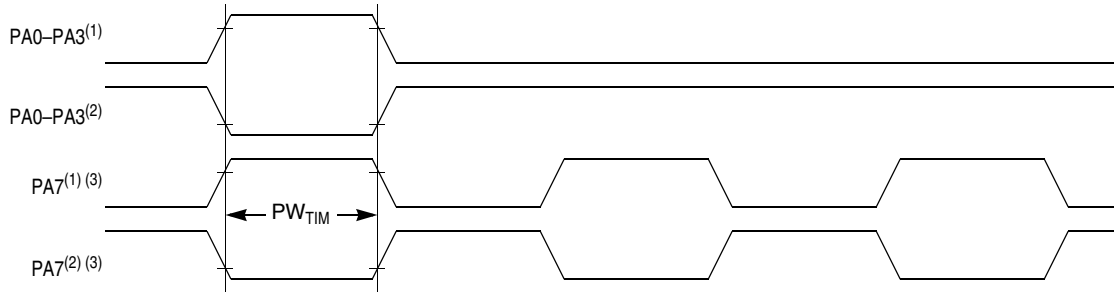
Bits 3–0 — Not implemented

Always read 0.

9.6 Control Timing

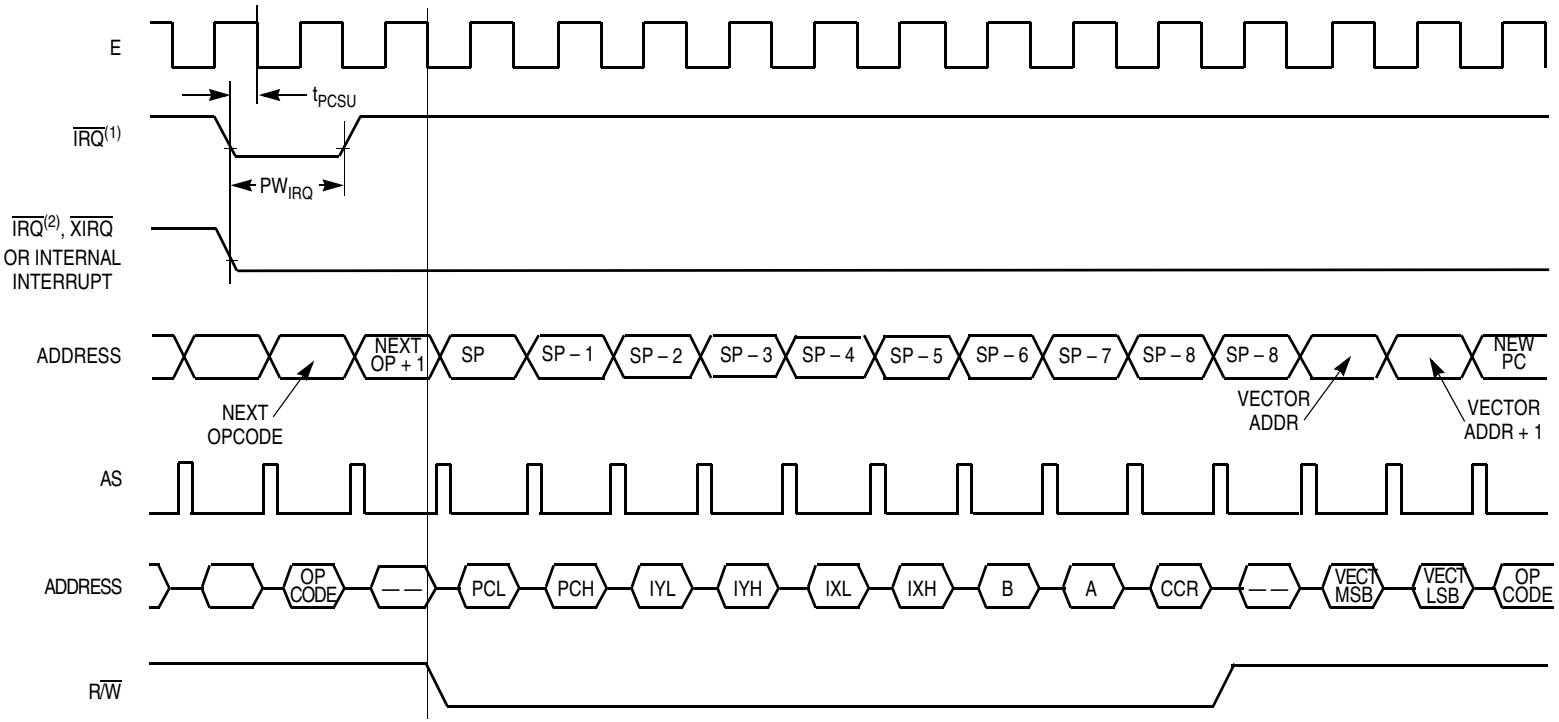
Characteristic[1]	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of operation	f_O	dc	1.0	dc	2.0	dc	3.0	MHz
E-clock period	t_{cyc}	1000	—	500	—	333	—	ns
Crystal frequency	f_{XTAL}	—	4.0	—	8.0	—	12.0	MHz
External oscillator frequency	4 f_O	dc	4.0	dc	8.0	dc	12.0	MHz
Processor control setup time t_{PCSU} = 1/4 t_{cyc} + 50 ns	t_{PCSU}	300	—	175	—	133	—	ns
Reset input pulse width[2] To guarantee external reset vector Minimum input time can be preempted 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, PW_{IRQ} = t_{cyc} + 20 ns $\overline{IRQ}$ edge-sensitive mode	PW_{IRQ}	1020	—	520	—	353	—	ns
Wait recovery startup time	t_{WRS}	—	4	—	4	—	4	t_{cyc}
Timer pulse width PW_{TIM} = t_{cyc} + 20 ns Input capture pulse Accumulator input	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. 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 Input/Output (I/O) Ports for further details.



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 9-3. Timer Inputs



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

Figure 9-7. Interrupt Timing Diagram

9.7 Peripheral Port Timing

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	1.0	1.0	2.0	2.0	3.0	3.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	333	—	ns
Peripheral data setup time[2] MCU read of ports A, B, C, and D	t_{PDSU}	100	—	100	—	100	—	ns
Peripheral data hold time[2] MCU read of ports A, B, C, and D	t_{PDH}	50	—	50	—	50	—	ns
Delay time, peripheral data write MCU write to port A MCU writes to ports B, C, and D t_{PWD} = 1/4 t_{cyc} + 150 ns	t_{PWD}	— —	200 350	— —	200 225	— —	200 183	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. Port C and D timing is valid for active drive (CWOM and DWOM bits not set in PIOC and SPCR registers respectively).

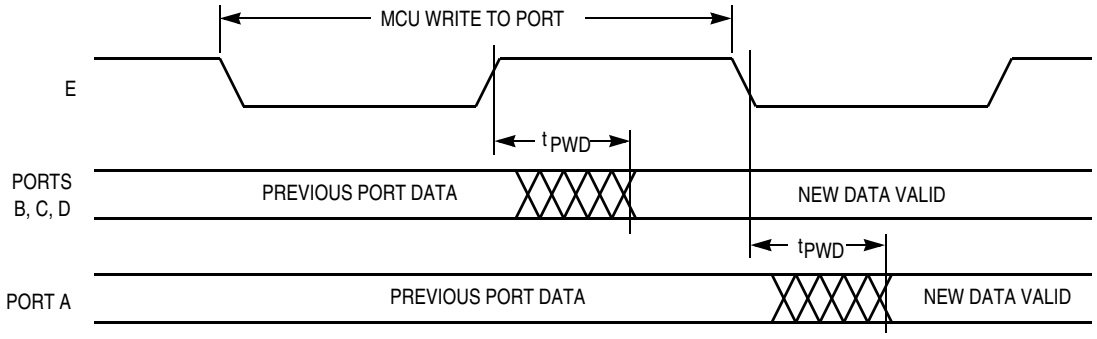


Figure 9-8. Port Write Timing Diagram

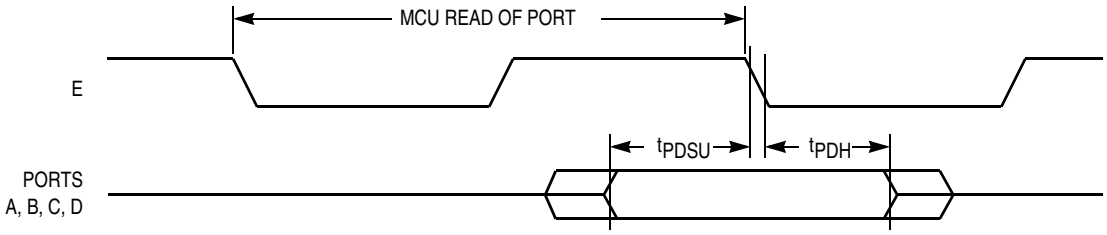
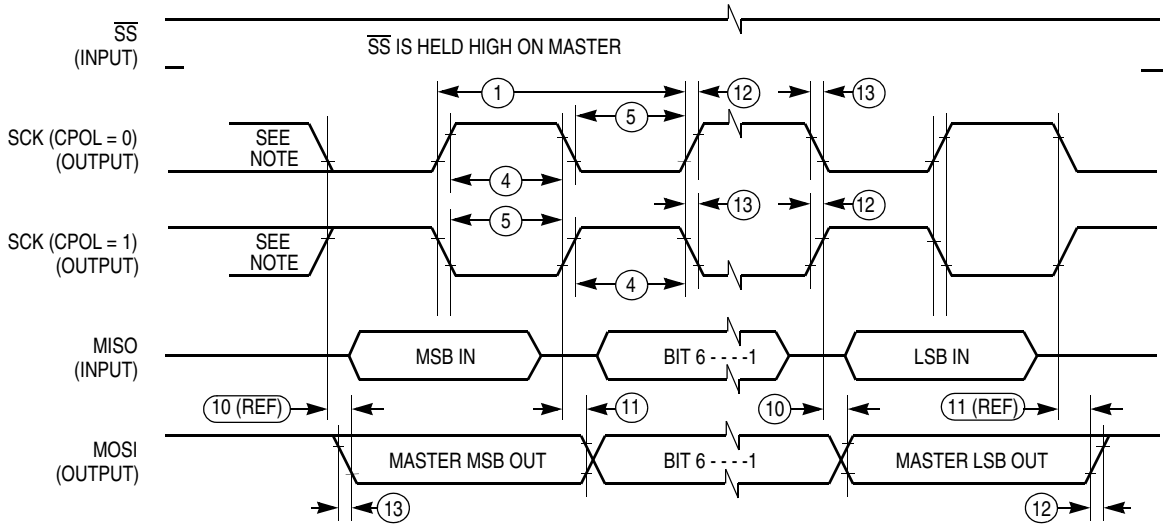
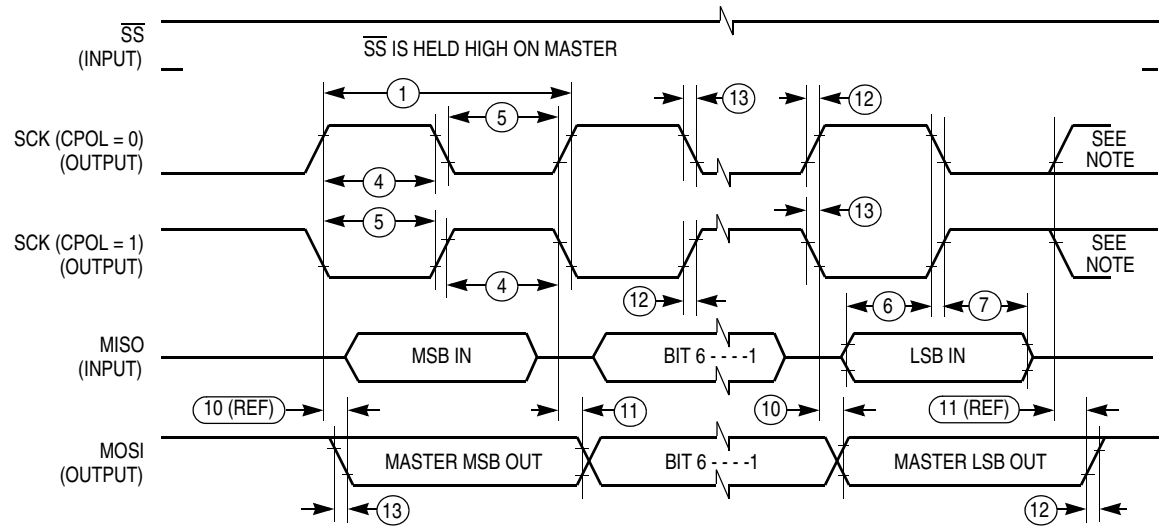


Figure 9-9. Port Read Timing Diagram



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

Figure 9-11. SPI Master Timing (CPHA = 0)



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

Figure 9-12. SPI Master Timing (CPHA = 1)

A.2 Block Diagram

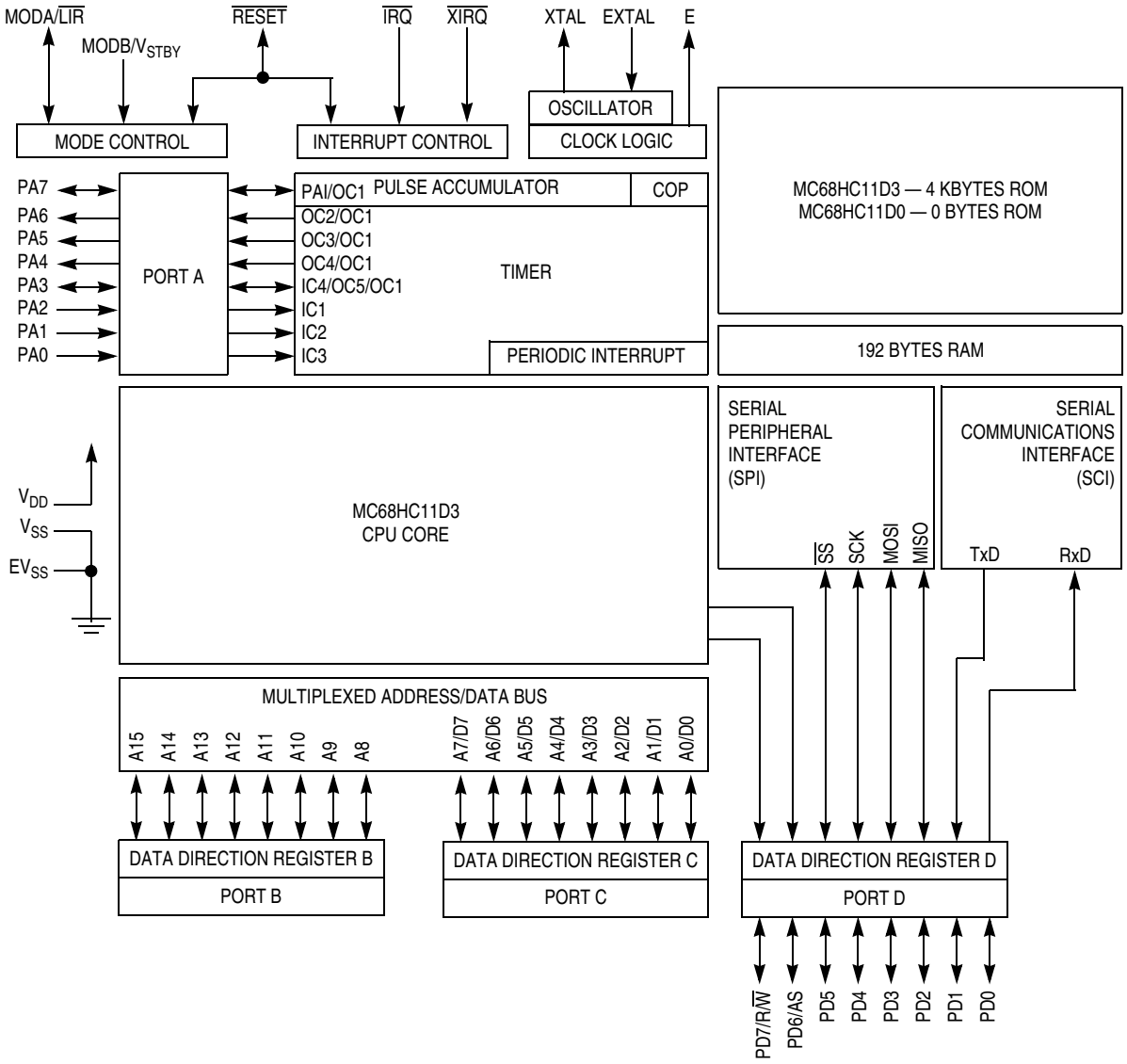


Figure A-1. MC68HC11D3 Block Diagram

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

MC68HC711D3
Rev. 2.1, 07/2005

freescale™
semiconductor