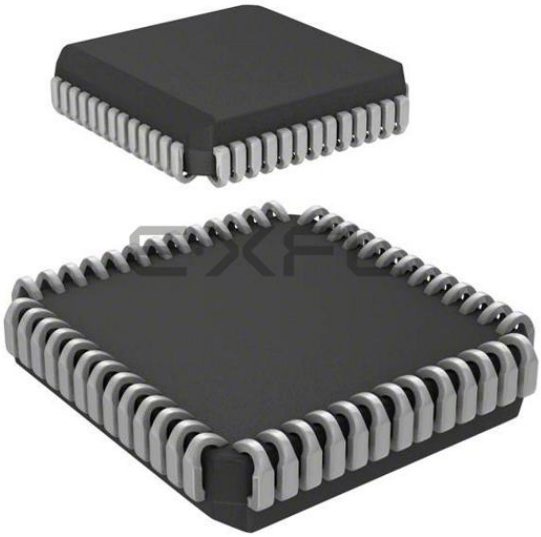


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	HC11
Core Size	8-Bit
Speed	3MHz
Connectivity	SCI, SPI
Peripherals	POR, WDT
Number of I/O	38
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LCC (J-Lead)
Supplier Device Package	52-PLCC (19.1x19.1)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc11e1cfne2

List of Chapters

Chapter 11
Ordering Information and Mechanical Specifications

Appendix A
Development Support

Appendix B
EVBU Schematic

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)

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.

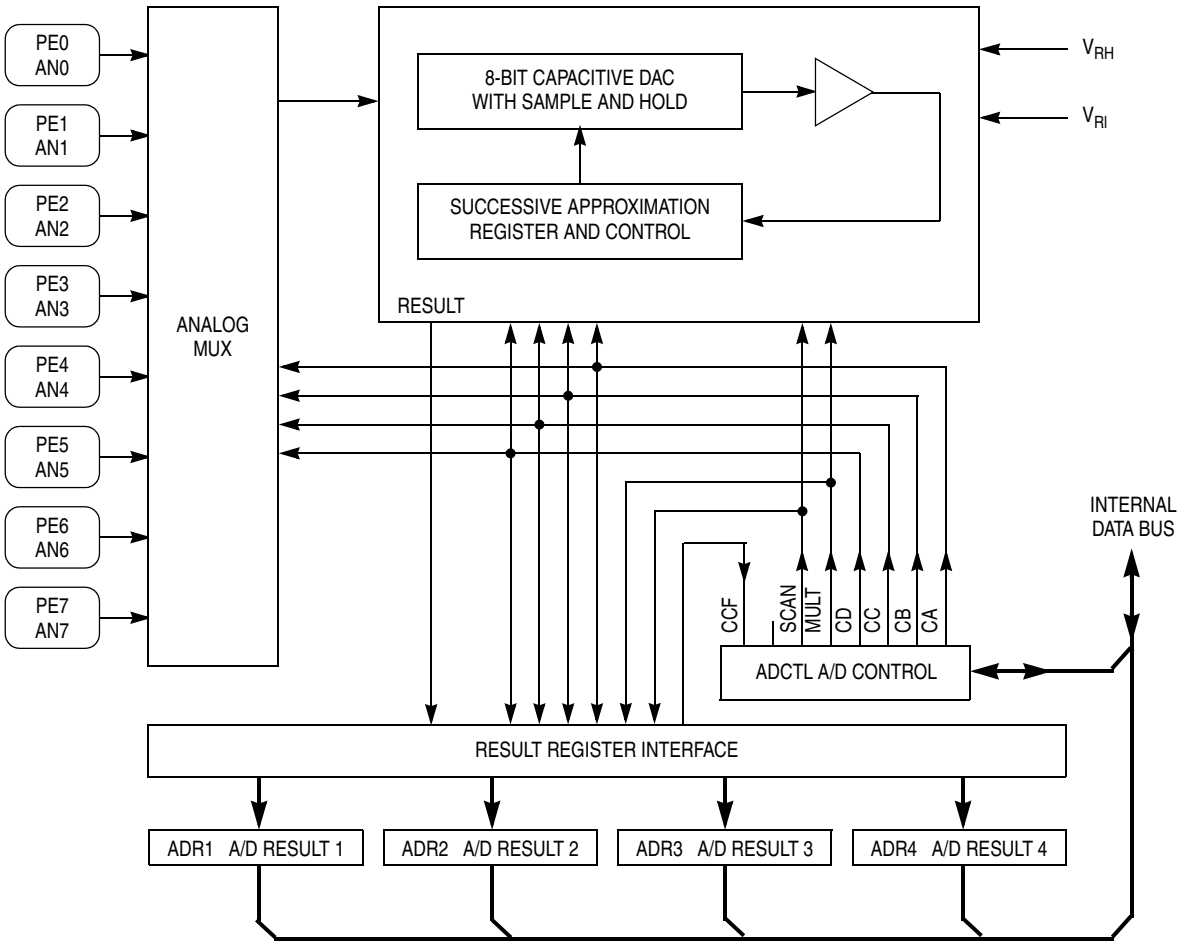
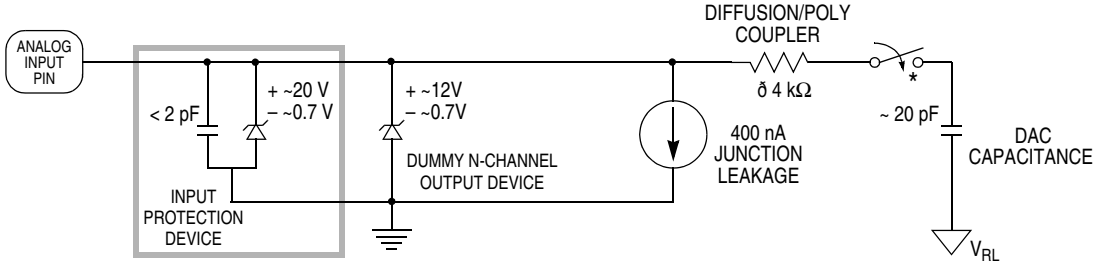


Figure 3-1. A/D Converter Block Diagram



* 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.

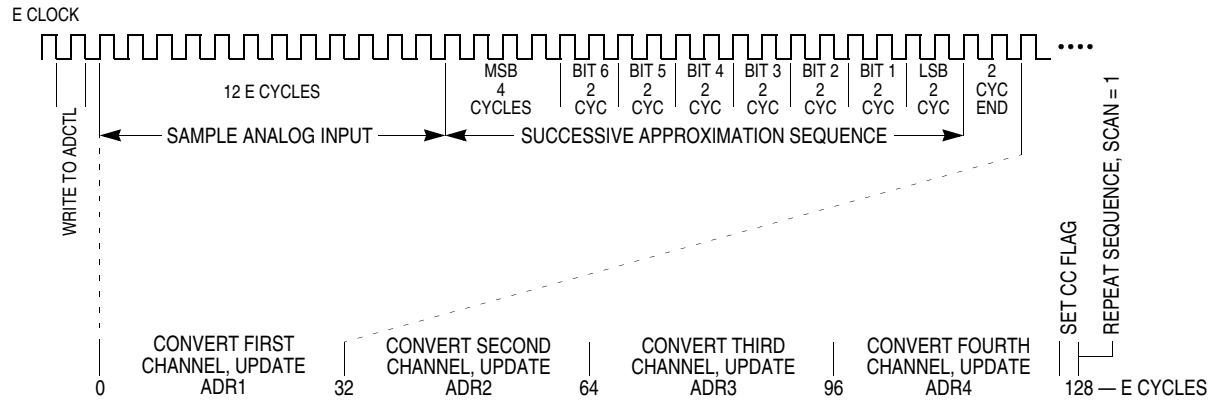


Figure 3-3. A/D Conversion Sequence

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

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.

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

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

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.

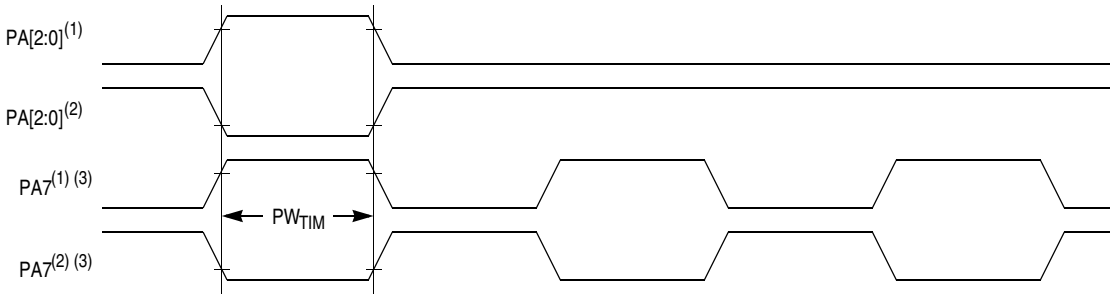
8.3 SPI Transfer Formats

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.

10.10 MC68L11E9/E20 Control Timing

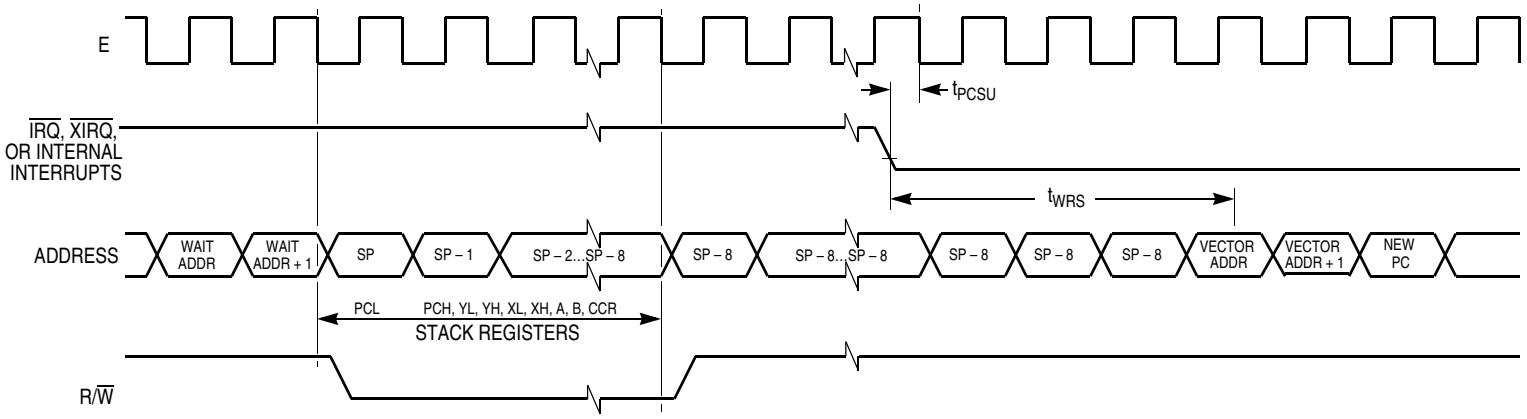
Characteristic(1) (2)	Symbol	1.0 MHz		2.0 MHz		Unit
		Min	Max	Min	Max	
Frequency of operation	f_o	dc	1.0	dc	2.0	MHz
E-clock period	t_{CYC}	1000	—	500	—	ns
Crystal frequency	f_{XTAL}	—	4.0	—	8.0	MHz
External oscillator frequency	4 f_o	dc	4.0	dc	8.0	MHz
Processor control setup time t_{PCSU} = 1/4 t_{CYC}+ 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



Note: RESET also causes recovery from WAIT.

Figure 10-5. WAIT Recovery from Interrupt Timing Diagram

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)

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.
```

```
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
```