Simple Interface for 68HC11 to MICROWIRE™ "Chip Security" EEPROMs

Fairchild
Application Note 909

FAIRCHILD SEMICONDUCTOR™

ABSTRACT

Fairchild's NM93CSxx Family of serial EEPROMs offers sophisticated protection against accidental overwrites from power surges, controller crashes, and other potential noise sources. Utilizing the data protection features in these devices require a few, simple command sequences which may not be familiar to users of standard MICROWIRE commands. This application note presents and explains assembly code for Motorola's 68HC11 microcontroller which implements all of the interface command sequences required to control the NM93CSxx Family of EEPROMs.

WHY SHOULD DESIGN ENGINEERS USE Fairchild SEMICONDUCTOR'S NM93CSxx FAMILY OF EEPROMs?

There has never been a more "data secure" EEPROM available on the market than the NM93CSxx devices. These "Chip Security" EEPROMs enable Design Engineers to bring new products to market quicker than ever before. The Chip Security features prevent the slightest risk of inadvertent writes caused by noise sources, and unplanned jumps in program sequencing that are difficult to "track down." Trial-and-error tactics to determine the cause of accidental overwrites often takes several hours which are not expendable in todays dynamic and fast paced market, so to help prevent data corruption problems that often delay product introduction dates, NM93CSxx devices have software and hardware features that eliminate those potential problems.

Another benefit realized by using the NM93CSxx Family of EEPROMs is preventing field failure returns of the end product. The hardware and software data protection features described in this application note prevent all types of accidental overwrites that could cause a field failure of the end product, such as losing radio stations stored in EEPROM or losing valuable calibration data points that are considered vital to the accuracy of test equipment. As the "Rule of Tens" states, if it cost $10 to find an error at the board level, then it will cost $100 at the system level, and finally $1000 at the field level. Why is the cost incurred exponential in nature? Simple, failures in the field can lose customers and ruin reputations if the product doesn't work.

BACKGROUND INFORMATION ON Fairchild SEMICONDUCTOR's EEPROMs

Fairchild Semiconductor is the market leader in low density serial EEPROMs, and defined the ubiquitous MICROWIRE interface commonly copied by our competitors. We offer standard MICROWIRE EEPROMs from 256 bits up to 16 Kbits (16K available 1st quarter of 1994). And for upscale products that can't afford the slightest risk of data corruption, we invented and brought to market the "Chip Security" MICROWIRE EEPROM Family. With over ten years of EEPROM design and process experience, we have also developed low voltage (1.8V to 6.0V) and Zero Standby (< 1 μA standby current) EEPROMs to meet the battery supported market needs.

BRIEF OVERVIEW OF THE NM93CSxx FEATURES

The "Chip Security" EEPROM Family has data protection features added to the standard MICROWIRE Family that give the ultimate data protection solution in non-volatile, serial memory devices. The assembly code required to read and write the NM93CSxx (Chip Security devices) is the same as that required by the NM93Cxx EEPROMs (standard devices). The difference lays in the added commands and hardwire pins that bring about the data protection.

The first available method of data protection with the NM93CSxx EEPROMs is by using the Program Enable (PE) pin. The PE pin can have a delay signal placed on it via software control, or an external switch for hardware control. Either way reduces the odds of accidental overwrites during voltage transients caused by power outages, unplugging equipment, system noise, unregulated batteries, or when changing batteries.

The second method of data protection with the NM93CSxx EEPROMs is to use the Protect Register which is controlled by the Protect Register Enable (PRE) pin and software commands. This is the best method to prevent overwrites caused by disturbed reset operations and crashing controllers that often write "garbage" to EEPROMs during these periods. "Garbage" gets written to the EEPROM because of the "skipping" binary counter in the controller. Just by having the PRE pin externally controlled, or having extra commands to execute a write command, reduces the odds of overwrites to near zero for the NM93CSxx Family.

The final method of protecting information in the EEPROM is to use the Protect Register Disable (PRDS) command. Once this command is executed after using the Protect Register commands, the user selected portion of the EEPROM array becomes permanently disabled from all future writes. This is the best method for protecting security codes, calibration information and any other information that needs to become ROM once written into the EEPROM.

BRIEF OVERVIEW OF THE 68HC11 MICROCONTROLLER

The MC68HC11 is a high density CMOS microcontroller which contain a microcontroller unit (MCU) and highly sophisticated integrated peripheral capabilities. An eight-channel A/D converter is included on chip. An 8-bit pulse accumulator subsystem on chip can count external events or measure external periods. The main 16-bit, free-running timer system has three input capture lines, five output compare lines, and a real-time interrupt function. An asynchronous serial communications interface (SCI) and a synchronous serial peripheral interface (SPI) are also included.

The SPI is an independent serial communications subsystem which allows the MCU to communicate synchronously with peripheral devices. This application note describes assembly code which interfaces the MC68HC11 to the NM93CSxx Family of EEPROMs through the SPI without any intermediate logic.

TABLE 1. Instruction Set for the NM93CSxx Family of EEPROMs

Command	Op Cd	Address NM93CS06 NM93CS46	Address NM93CS56 NM93CS66	Data	Pre	Pe	Comments
READ	10	A5–A0	A7–A0		0	X	Reads data stored in memory, starting at specified address.
WEN	00	11XXXX	11XXXXXX		0	1	Write enable must precede all programming modes.
WRITE	01	A5–A0	A7–A0	D15–D0	0	1	Writes register if address is unprotected.
WRALL	00	01XXXX	01XXXXXX	D15–D0	0	1	Writes all registers. Valid only when Protect Register is cleared.
WRDS	00	00XXXX	00XXXXXX		0	X	Disables all programming instructions.
PRREAD	10	XXXXXX	XXXXXXXX		1	X	Reads address stored in Protect Register.
PREN	00	11XXXX	11XXXXXX		1	1	Must immediately precede PRCLEAR, PRWRITE, and PRDS instructions.
PRCLEAR	11	111111	11111111		1	1	Clears the Protect Register, so that no registers are protected from WRITE. Protect register equals 0000.
PRWRITE	01	A5–A0	A7–A0		1	1	Programs address into Protect Register. Thereafter, memory addresses greater than or equal to the address in Protect Register are protected from WRITE.
PRDS	00	000000	00000000		1	1	One time only instruction after which the address in the Protect Register cannot be altered.

MC68HC11/NM93CSxx INTERFACE HARDWARE DESCRIPTION

A block diagram of the interface between the MC68HC11 microcontroller and the NM93CSxx family of serial EEPROMs is given in Figure 1. Straps are inserted in the PRE and PE lines in the schematic (attached) to permit hardware protection from accidental data corruption. By removing the straps, data cannot be written to the EEPROM, but the EEPROM can still be read.

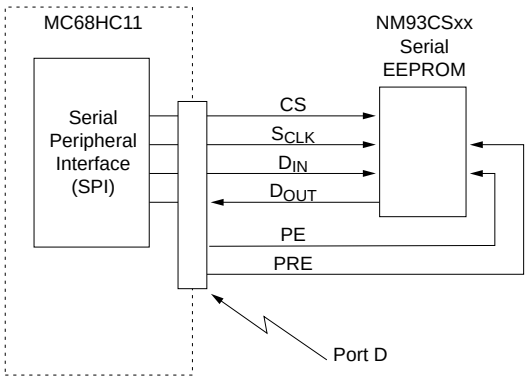


FIGURE 1. 68HC11 to 93CSxx Block Diagram

MC68HC11/NM93CSxx INTERFACE CODE DESCRIPTION

The assembly code which interfaces the MC68HC11 microcontroller to the NM93CSxx serial EEPROMs is structured as 10 subroutines: one subroutine for each command type. A data flow diagram of the code is shown in Figure 2. Each routine pushes the values of the A, B, X, and Y accumulators on a subroutine call and restore their values on return. The routines expect the address of the command being sent (if required) to reside at XADDR, and the data to be sent (if required) to reside at location $XDAT_{HI}$ and $XDAT_{LO}$.

Reads of the serial EEPROM are implemented as polled reads ; that is, the subroutine which implements the reads goes into a polling mode which checks that the read is complete before control is returned to the calling program. Writes to the EEPROM are implemented as posted writes; that is, the first byte of the command message is sent out, then control is returned to the calling program. The rest of the command message is sent out by an interrupt service routine which sends each additional byte as the SPI interrupts the main program when the previous byte is complete. This necessitates that each subroutine check that the previous command write is complete before it allows the next read or write to take place (implemented in subroutine WRPEN). Posted writes are used to improve real-time performance, particularly when the SPI clock rate is low. It also provides an example of implementing the MC68HC11 to NM93CSxx as a polled routine and as an interrupt driven routine.

Two commands: READ and PRRD, read data from the EEPROM; these commands are implemented such that the subroutine does not release control of the microcontroller until the SPI has completed sending and receiving the entire command message. The subroutines pack the READ or PRRD commands to be sent in the transmit buffer (XMESSx), set the count of the number of bytes in the command message (MESSCT), set the PRE and PE bits appropriately, and jump to the POLLRD routine. This routine applies the PRE, PE, and CS signals to the EEPROM and sends the first packet in the transmit buffer. The routine then waits for the SPI to acknowledge that it has completed sending that byte of the command message. The received data is then unloaded into the receive buffer (RDATHI or RDATLO , whichever is appropriate). The next byte in the transmit buffer is then sent out, the routine waits for the SPI to acknowledge the byte was sent, then the received data is unloaded if necessary. This process is repeated until the entire command message is sent out and all the data is received from the EEPROM, then the subroutine relinquishes control to the main code.

The remainder of the commands (WRITE, PRWRITE, WREN, PREN, WRALL, PRWRITE, WRDIS, and PRD5) only write data to the EEPROM; these commands are implemented such that the subroutine releases control back to the main program immediately after the first byte of the command message is sent. These subroutines pack the commands to be sent in the transmit buffer (XMESSx), set the count of the number of bytes in the command message (MESSGT), set the PRE and PE bits appropriately, and jump to the PSTWR routine.

This routine applies the PRE, PE, and CS signals to the EEPROM and sends the first packet in the transmit buffer and control is returned to the main program. When the byte transfer is complete, the SPI interrupts the microcontroller, and the interrupt service routine sends the next byte of the command message if applicable.

A summary of the command messages, the data structure, and the bytes transmitted for each is given in Table 2 for NM93CS06 and NM93CS46, and Table 3 for NM93CS56 and NM93CS66.

TABLE 2. SPI Messages for NM93CS06 and NM93CS46

Command	Pre	Pe	XMESS3	XMESS2	XMESS1	XMESS0	$RDAT_{HI}$	$RDAT_{LO}$
READ	0	0	0000 0011	0 a_5 a_4 a_3 a_2 a_1 a_0 0	0000 0000	0000 0000	D_{15} –D_8	D_7 –D_0
WEN	0	1			0000 0001	0011 0000		
WRITE	0	1	0000 0001	0 1 a_5 a_4 a_3 a_2 a_1 a_0	D_{15} –D_8	D_7 –D_0		
WRALL	0	1	0000 0001	0100 0000	D_{15} –D_8	D_7 –D_0		
WRDS	0	1			0000 0001	0000 0000		
PRRD	1	0		0000 0011	0 a_5 a_4 a_3 a_2 a_1 a_0 0	0000 0000	xxxx xxx0	a_5 a_4 a_3 a_2 a_1 a_0 x x
PREN	1	1			0000 0001	0011 0000		
PRCLR	1	1			0000 0001	1111 1111		
PRWRT	1	1			0000 0001	0 1 a_5 a_4 a_3 a_2 a_1 a_0		
PRDS	1	1			0000 0001	0000 0000		

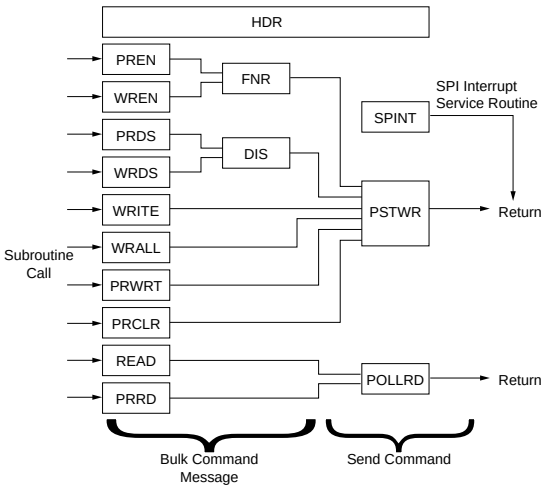
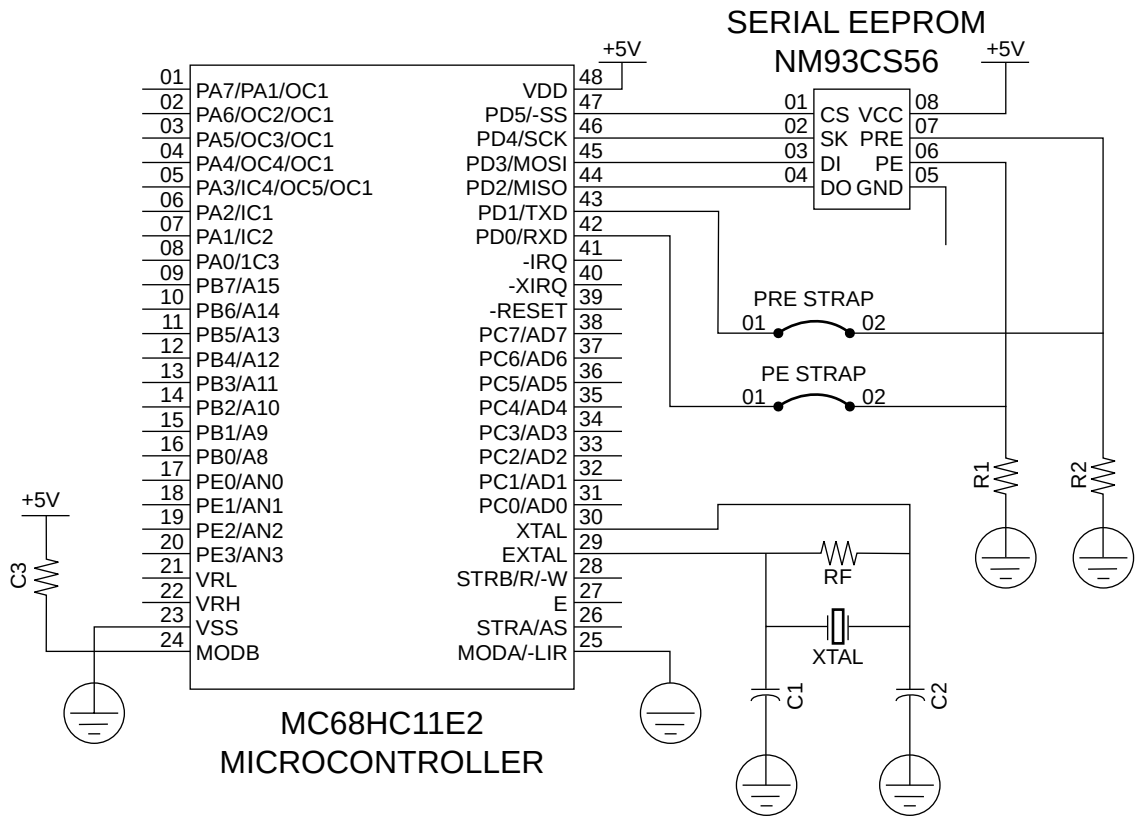


FIGURE 2. 93CSxx Interface Code Flow Diagram

TABLE 3. SPI Messages for NM93CS56 and NM93CS66

Command Pre Pe XMESS3 XMESS2 XMESS1 XMESS0 RDATHI RDATLO

Command	Pre	Pe	XMESS3	XMESS2	XMESS1	XMESS0	$RDAT_{HI}$	$RDAT_{LO}$
WEN	0	1			0000 0100	1100 0000		
WRITE	0	1	0000 0101	$a_7\ a_6\ a_5\ a_4$ $a_3\ a_2\ a_1\ a_0$	D_{15} –D_8	D_7 –D_0		
WRALL	0	1	0000 0101	0000 0000	D_{15} –D_8	D_7 –D_0		
WRDS	0	1			0000 0100	0000 0000		
PRRD	1	0		0000 110a_7	$a_6\ a_5\ a_4\ a_3$ $a_2\ a_1\ a_0\ 0$	0000 0000	xxxx xxx0	$a_5\ a_4\ a_3\ a_2$ $a_1\ a_0$ x x
PREN	1	1			0000 0100	1100 0000		
PRCLR	1	1			0000 0111	1111 1111		
PRWRT	1	1			0000 0101	$a_7\ a_6\ a_5\ a_4$ $a_3\ a_2\ a_1\ a_0$		
PRDS	1	1			0000 0100	0000 0000		



Assembly Code for Interfacing the NM93CS06 and NM93CS46 EEPROMs to the MC68HC11

```
                          1    ;MC68HC11 to NM93CS06/46 interface code
                          2    ;Revision A                 7/28/93
                          3    ;This assembly code implements the serial interface
                          4    ;to the NM93CS06/46 serial EEPROM with Chip Security
                          5    ;through the SPI interface and the rest of Port D.
                          6    ;There are 10 commands:
                          7    ;   WRITE                   Protect Reg WRite (PRWR)
                          8    ;   READ                    Protect Reg ReaD (PRRD)
                          9    ;   WRite ENable (WREN)     Protect Reg ENable (PREN)
                         10    ;   WRite DiSable (WRDS)    Protect Reg DiSable (PRDS)
                         11    ;   WRite ALL (WRALL)       Protect Reg CLeaR (PRCLR)
                         12    ;Each EEPROM command type is implemented as a
                         13    ;subroutine call. Address is passed to the routine
                         14    ;in XADDR, data in XDATLO and XDATHI. Data is
                         15    ;returned from read operations in RDATLO and RDATHI.
                         16
0000                     17    $ include “header.asm”
0000                     18    PORTD     equ $1008    ;Port D data Register
0000                     19    DDRD      equ $1009    ;D Data Direction Register
0000                     20    SPCR      equ $1028    ;SPI Control Register
0000                     21    SPSR      equ $1029    ;SPI Status Register
0000                     22    SPDR      equ $102a    ;SPI Data Register
0000                     23    HPRIO     equ $103c    ;Interrupt Priority
                         24
0000                     25    XMESS0    equ $00      ;Transmit Message Buffer 0
0000                     26    XMESS1    equ $01      ;Transmit Message Buffer 1
0000                     27    XMESS2    equ $02      ;Transmit Message Buffer 2
0000                     28    XMESS3    equ $03      ;Transmit Message Buffer 3
0000                     29    MESSCT    equ $04      ;Message Byte Count
0000                     30    PREPE     equ $05      ;CS, Pre, Pe Bit settings
0000                     31    EXTWR     equ $0b      ;Set if write takes Extra Time
0000                     32    WRACTV    equ $0c      ;Set if write still active after
                         33                           ;message count goes to 00
                         34
0000                     35    XADDR     equ $06      ;Transmit Address
0000                     36    XDATLO    equ $07      ;Transmit Data low byte
0000                     37    XDATHI    equ $08      ;Transmit Data high bype
0000                     38    RDATLO    equ $09      ;Receive Data low byte
0000                     39    RDATHI    equ $0a      ;Receive Data high byte
                         40
0000                     41    $ include “init.asm”
                         42              ;Init intializes registers critical to SPI
                                                                    operation
                         43
C000                     44              org $c000          ;Start code at C000
C000  8E00FF             45    init1     lds #$00ff         ;Locate Stack at 00FF
C003  863B               46              ldaa #$3b          ;Initialize DDRD Register
C005  B71009             47              staa DDRD
C008  8653               48              ldaa #$53          ;Initialize SPCR Register
C00A  B71028             49              staa SPCR
C00D  8600               50              ldaa #$00          ;Zero out Port D Outputs
C00F  B71008             51              staa PORTD
C012  8603               52              ldaa #$03          ;Set SPI Interrupt to highest
                                                                    priority
C014  B7103C             53              staa HPRIO
C017  C6C0               54              ldab #$c0          ;Turn off global interrupt disable
C019  06                 55              tap
C01A  8600               56              ldaa #$00
```

```
C01C  9704              57              staa MESSCT       ;Message count is zero
C01E  970C              58              staa WRACTV       ;Write not active
                        59
C020  86AA              60    main      ldaa #$aa         ;Main give examples of using
                        61                                ;a few of the command rountines
C022  9706              62              staa XADDR        ;Load in transmit address
C024  8633              63              ldaa #$33
C026  9707              64              staa XDATLO       ;Load in low byte xmit dat
C028  86CC              65              ldaa #$cc
C02A  9708              66              staa XDATHI       ;Load in hi byte xmit data
C02C  BDC051            67              jsr WREN          ;Call Write Enable
C02F  BDC03E            68              jsr PREN          ;Call Protect Register Enable
C032  BDC16D            69              jsr PRCLR         ;Call Protect Register Clear
C035  BDC0F0            70              jsr WRITE         ;Call Write
C038  BDC0A2            71              jsr READ          ;Call Read
                        72                                ;(read back memory)
C03B  7EC020            73              jmp main
                        74
C03E  36                75    PrEn      psha
C03F  37                76              pshb
C040  3C                77              pshx
C041  183C              78              pshy
C043  BDC18C            79              jsr WrPen         ;Subroutine Write Pending checks
                        80                                ;whether a previous write is still
                        81                                ;pending, and waits until it is
                        82                                ;done if there is.
C046  8623              83              ldaa #$23
C048  9705              84              staa PREPE        ;Set CS, PRE, and PE bits
C04A  8600              85              ldaa #$00
C04C  970B              86              staa EXTWR        ;No extra write time required
C04E  7EC061            87              jmp Enb
                        88
C051  36                89    WrEn      psha
C052  37                90              pshb
C053  3C                91              pshx
C054  183C              92              pshy
C056  BDC18C            93              jsr WrPen
C059  8621              94              ldaa #$21
C05B  9705              95              staa PREPE        ;Set CS, PRE, and PE bits
C05D  8600              96              ldaa #$00
C05F  970B              97              staa EXTWR        ;No extra write time required
                        98
C061  8602              99    Enb       ldaa #$02
C063  9704             100              staa MESSCT       ;Load message byte count
C065  8630             101              ldaa #$30         ;Load last byte, op code=00
C067  9700             102              staa XMESS0       ;Data is 110000
C069  8601             103              ldaa #$01
C06B  9701             104              staa XMESS1       ;Load start bit
C06D  7EC195           105              jmp PstWr         ;Jump to Posted Write Routine
                       106
C070  36               107    PrDs      psha
C071  37               108              pshb
C072  3C               109              pshx
C073  183C             110              pshy
C075  BDC18C           111              jsr WrPen         ;Check Write still Pending?
C078  8623             112              ldaa #$23
C07A  9705             113              staa PREPE        ;Set CS, PRE, and PE bits
C07C  8601             114              ldaa #$01
```

```
C07E  970B            115             staa EXTWR        ;Extra write time required
C080  7EC093          116             jmp Dis
                      117
C083  36              118   WrDs      psha
C084  37              119             pshb
C085  3C              120             pshx
C086  183C            121             pshy
C088  BDC18C          122             jsr WrPen         ;Check Write still Pending?
C08B  8621            123             ldaa #$21
C08D  9705            124             staa PREPE        ;Set CS, PRE, and PE bits
C08F  8600            125             ldaa #$00
C091  970B            126             staa EXTWR        ;No extra write time required
                      127
C093  8602            128   Dis       ldaa #$02
C095  9704            129             staa MESSCT       ;Load message byte count
C097  8600            130             ldaa #$00         ;Load last byte, op code=00
C099  9700            131             staa XMESS0       ;Data = 000000
C09B  8601            132             ldaa #$01
C09D  9701            133             staa XMESS1       ;Load start bit
C09F  7EC195          134             jmp PstWr         ;jump to Posted Write Routine
                      135
C0A2  36              136   Read      psha
C0A3  37              137             pshb
C0A4  3C              138             pshx
C0A5  183C            139             pshy
C0A7  BDC18C          140             jsr WrPen         ;Check Write still Pending?
C0AA  8620            141             ldaa #$20
C0AC  9705            142             staa PREPE        ;Set CS, PRE, and PE bits
C0AE  8600            143             ldaa #$00
C0B0  970B            144             staa EXTWR        ;No extra write time required
C0B2  8604            145             ldaa #$04
C0B4  9704            146             staa MESSCT       ;Load message byte count
C0B6  8600            147             ldaa #$00
C0B8  9700            148             staa XMESS0       ;Load low byte data (don't care)
C0BA  9701            149             staa XMESS1       ;Load hi byte data (don't care)
C0BC  9606            150             ldaa XADDR
C0BE  48              151             asla
C0BF  847E            152             anda #$7e         ;Mask off bits 0 & 7 of address
C0C1  9702            153             staa XMESS2       ;Load address byte
C0C3  8603            154             ldaa #$03
C0C5  9703            155             staa XMESS3       ;Load start bit and MSB of op code
C0C7  7EC215          156             jmp PollRd        ;Jump to Polled Read Routine
                      157
C0CA  36              158   PrRd      psha
C0CB  37              159             pshb
C0CC  3C              160             pshx
C0CD  183C            161             pshy
C0CF  BDC18C          162             jsr WrPen         ;Check Write still Pending?
C0D2  8622            163             ldaa #$22
C0D4  9705            164             staa PREPE        ;Set CS, PRE, and PE bits
C0D6  8600            165             ldaa #$00
C0D8  970B            166             staa EXTWR        ;No extra write time required
C0DA  8603            167             ldaa #$03
C0DC  9704            168             staa MESSCT       ;Load message byte count
C0DE  8600            169             ldaa #$00
C0E0  9700            170             staa XMESS0       ;Load low byte (don't care)
C0E2  9606            171             ldaa XADDR
C0E4  48              172             asla
```

```
C0E5  847E            173             anda #$7e         ;Mask off bits 0 & 7 of address
C0E7  9701            174             staa XMESS1       ;Load address byte
C0E9  8603            175             ldaa #$03
C0EB  9702            176             staa XMESS2       ;Load start bit and MSB of op code
C0ED  7EC215          177             jmp PollRd        ;Jump to Polled Read Routine
                      178
C0F0  36              179   Write     psha
C0F1  37              180             pshb
C0F2  3C              181             pshx
C0F3  183C            182             pshy
C0F5  BDC18C          183             jsr WrPen         ;Check Write still Pending?
C0F8  8621            184             ldaa #$21
C0FA  9705            185             staa PREPE        ;Set CS, PRE, and PE bits
C0FC  8601            186             ldaa #$01
C0FE  970B            187             staa EXTWR        ;Extra write time required
C100  8604            188             ldaa #$04
C102  9704            189             staa MESSCT       ;Load message byte count
C104  9607            190             ldaa XDATLO
C106  9700            191             staa XMESS0       ;Load low xmit data byte
C108  9608            192             ldaa XDATHI
C10A  9701            193             staa XMESS1       ;Load high xmit data byte
C10C  863F            194             ldaa #$3f
C10E  C640            195             ldab #$40
C110  9406            196             anda XADDR        ;Mask off bits 6 & 7 of address
C112  9702            197             staa XMESS2
C114  DA02            198             orab XMESS2       ;Add op code=01 to address
C116  D702            199             stab XMESS2       ;Load address byte
C118  8601            200             ldaa #$01
C11A  9703            201             staa XMESS3       ;Load start bit
C11C  7EC195          202             jmp PstWr         ;Jump to Posted Write Routine
                      203
C11F  36              204   WrAll     psha
C120  37              205             pshb
C121  3C              206             pshx
C122  183C            207             pshy
C124  BDC18C          208             jsr WrPen         ;Check if Write still Pending?
C127  8621            209             ldaa #$21
C129  9705            210             staa PREPE        ;Set CS, PRE, and PE bits
C12B  8601            211             ldaa #$01
C12D  970B            212             staa EXTWR        ;Extra write time required
C12F  8604            213             ldaa #$04
C131  9704            214             staa MESSCT       ;Load message byte count
C133  9607            215             ldaa XDATLO
C135  9700            216             staa XMESS0       ;Load low xmit data byte
C137  9608            217             ldaa XDATHI
C139  9701            218             staa XMESS1       ;Load high xmit data byte
C13B  8610            219             ldaa #$10
C13D  9702            220             staa XMESS2       ;Load op code=00, rest 010000
C13F  8601            221             ldaa #$01
C141  9703            222             staa XMESS3       ;Load start bit
C143  7EC195          223             jmp PstWr         ;Jump to Posted Write Routine
                      224
C146  36              225   PrWrt     psha
C147  37              226             pshb
C148  3C              227             pshx
C149  183C            228             pshy
C14B  BDC18C          229             jsr WrPen         ;Check if Write still Pending?
C14E  8623            230             ldaa #$23
```

```
C150  9705            231             staa PREPE        ;Set CS, PRE, and PE bits
C152  8601            232             ldaa #$01
C154  970B            233             staa EXTWR        ;Extra write time required
C156  8602            234             ldaa #$02
C158  9704            235             staa MESSCT       ;Load message byte count
C15A  863F            236             ldaa #$3f
C15C  C640            237             ldab #$40
C15E  9406            238             anda XADDR        ;Mask off bits 6 & 7 of address
C160  9700            239             staa XMESS0
C162  DA00            240             orab XMESS0       ;Add op code=01 to address
C164  D700            241             stab XMESS0       ;Load address byte
C166  8601            242             ldaa #$01
C168  9701            243             staa XMESS1       ;Load start bit
C16A  7EC195          244             jmp PstWr         ;Jump to Posted Write Routine
                      245
C16D  36              246   PrClr     psha
C16E  37              247             pshb
C16F  3C              248             pshx
C170  183C            249             pshy
C172  BDC18C          250             jsr WrPen         ;Check if Write still Pending?
C175  8623            251             ldaa #$23
C177  9705            252             staa PREPE        ;Set CS, PRE, and PE bits
C179  8601            253             ldaa #$01
C17B  970B            254             staa EXTWR        ;Extra write time required
C17D  8602            255             ldaa #$02
C17F  9704            256             staa MESSCT       ;Load message byte count
C181  86FF            257             ldaa #$0ff
C183  9700            258             staa XMESS0       ;Load address byte of all one's
C185  8601            259             ldaa #$01
C187  9701            260             staa XMESS1       ;Load start bit
C189  7EC195          261             jmp PstWr         ;Jump to Posted Write Routine
                      262
C18C  9604            263   WrPen     ldaa MESSCT       ;Write Pending Subroutine:
C18E  26FC            264             bne WrPen         ;Check if previous message byte
                      265                               ;count is 0 before preceding
C190  960C            266             ldaa WRACTV       ;Check write still active even
C192  26F9            267             bne WrPen         ;though last message sent
C194  39              268             rts               ;Note that a bad write (ie. WREN
                      269                               ;has not been sent) will not return
                      270                               ;a ready so this loop will never
                      271                               ;stop... a timer (approx 10ms)
                      272                               ;should be implemented to monitor
                      273                               ;for this situation.
                      274
C195  8680            275   PstWr     ldaa #$80
C197  BA1028          276             oraa SPCR
C19A  B71028          277             staa SPCR         ;Enable SPI Interrupt
C19D  D604            278             ldab MESSCT       ;Message byte count in B
C19F  CE0000          279             ldx #XMESS0       ;Address of last message byte
C1A2  B61008          280             ldaa PORTD        ;Read Port D
C1A5  9A05            281             oraa PREPE
C1A7  B71008          282             staa PORTD        ;Set CS, PRE, and PE lines of PORT D
C1AA  5A              283             decb              ;decrement message indexing B
C1AB  3A              284             abx               ;index address in X to mess byte
C1AC  A600            285             ldaa 0, x         ;Load A with message byte data
C1AE  B7102A          286             staa SPDR         ;send data packet (start bit)
C1B1  1838            287             puly
C1B3  38              288             pulx
```

```
C1B4  33              289             pulb
C1B5  32              290             pula
C1B6  39              291             rts             ;Return to main program...the
                      292                             ;rest of the message is sent
                      293                             ;by interrupt routine SpInt
                      294
C1B7  F61029          295   SpInt     ldab SPSR       ;Check that interrupt cause by
C1BA  sA58            296             bpl done        ;SPIF (not mode error)
C1BC  B6102A          297             ldaa SPDR       ;Clear SPIF interrupt
C1BF  D604            298             ldab MESSCT     ;Load message byte count in B
C1C1  272D            299             beq Twp         ;Jump to post write poller
C1C3  5A              300             decb            ;Decrement message count
C1C4  D704            301             stab MESSCT     ;Store that 1 byte was sent
C1C6  270D            302             beq SsOff       ;Branch if last byte was sent
C1C8  CE0000          303             ldx #XMESS0     ;Load address of last byte in X
C1CB  5A              304             decb            ;Decrement message index B
C1CC  3A              305             abx             ;Index address in X to next byte
C1CD  A600            306             ldaa 0,x        ;Load next message byte
C1CF  B7102A          307             staa SPDR       ;Send message byte
C1D2  7EC214          308             jmp Done        ;Return to Main Program
C1D5  B61008          309   SsOff     ldaa PORTD
C1D8  84DF            310             anda #$df
C1DA  B71008          311             staa PORTD      ;Turn off CS
C1DD  84FC            312             anda #$fc
C1DF  B71008          313             staa PORTD      ;Turn off PRE, PE
C1E2  060B            314             ldaa EXTWR
C1E4  C716            315             beq Off         ;Jump to turn off interrupts
C1E6  B61008          316             ldaa PORTD
C1E9  8A20            317             oraa #$20
C1EB  B71008          318             staa PORTD      ;Turn CS on
C1EE  8600            319             ldaa #$00
C1F0  81FF            320   Twp       cmpa #$ff       ;Check if write complete
C1F2  2617            321             bne Send        ;No, send another idle byte
C1F4  B61008          322             ldaa PORTD
C1F7  84DF            323             anda #$df
C1F9  B71008          324             staa PORTD      ;Turn off CS
C1FC  867F            325   Off       ldaa #$7f
C1FE  B41028          326             anda SPCR
C201  B71028          327             staa SPCR       ;Disable SPI interrupt
C204  8600            328             ldaa #$00
C206  970C            329             staa WRACTV     ;Write complete
C208  7EC214          330             jmp Done
C20B  8601            331   Send      ldaa #$01
C20D  970C            332             staa WRACTV     ;Set write active
C20F  8600            333             ldaa #$00
C211  B7102A          334             staa SPDR       ;Send data
C214  3B              335   Done      rti             ;Return to main program
                      336
C215  D604            337   PollRd    ldab MESSCT     ;Load message byte count in B
C217  CE0000          338             ldx #MESS0      ;Load address of last byte in X
C21A  B61008          339             ldaa PORTD
C21D  9A05            340             oraa PREPE
C21F  B71008          341             staa PORTD      ;Set CS, PRE, and PE
C222  3A              342             abx             ;Index into address past byte
C223  09              343   SendX     dex             ;Decrement back to byte
C224  5A              344             decb            ;Decrement index B
C225  A600            345             ldaa 0,x        ;Load next message byte
C227  B7102A          346             staa SPDR       ;Send message byte
```

```
C22A  B61029           347   Wait     ldaa SPSR         ;Load SPI Status register
C22D  2AFB             348            bpl Wait          ;Wait unit SPIF is set
C22F  B6102A           349            ldaa SPDR         ;Load in data to A
C232  C100             350            cmpb #$00         ;Last message byte?
C234  2705             351            beq StrLo
C236  970A             352            staa RDATHI       ;Save byte as hi receive byte
C238  7EC223           353            jmp SendX         ;Send/receive next byte
C23B  9709             354   StrLo    staa RDATLO       ;Store lo receive byte
C23D  D704             355            stab MESSCT       ;Zero out message byte count
C23F  86DC             356            ldaa #$dc
C241  B41008           357            anda PORTD
C244  B71008           358            staa PORTD        ;Turn off CS, PRE, PE
C247  1838             359            puly
C249  38               360            pulx
C24A  33               361            pulb
C24B  32               362            pula
C24C  39               363            rts               ;Return to Main Program
                       364
FFD8                   365            org $ffd8
FFD8  C1B7             366            fdb SpInt         ;Set SPI interrupt vector to
                       367                              ;point to SPI interrupt service
                       368                              ;routine
                       369
                       370

Symbol Table
DDRD                1009
DIS                 C093
DONE                C214
ENB                 C061
EXTWR               000B
HPRIO               103C
INIT1               C000
MAIN                C020
MESSCT              0004
OFF                 C1FC
POLLRD              C215
PORTD               1008
PRCLR               C16D
PRDS                C070
PREN                C03E
PREPE               0005
PRRD                C0CA
PRWRT               C146
PSTWR               C195
RDATHI              000A
RDATLO              0009
READ                C0A2
SEND                C20B
SENDX               C223
SPCR                1028
SPDR                102A
SPINT               C1B7
SPSR                1029
SSOFF               C1D5
STRLO               C23B
TWP                 C1F0
WAIT                C22A
WRACTV              000C
WRALL               C11F
WRDS                C083
WREN                C051
WRITE               C0F0
WRPEN               C18C
XADDR               0006
XDATHI              0008
XDATLO              0007
XMESS0              0000
XMESS1              0001
XMESS2              0002
XMESS3              0003
```

Assembly Code for Interfacing the NM93CS56 and NM93CS66 EEPROMs to the MC68HC11

```
                          1    ;MC68HC11 to NM93CS56/66 interface code
                          2    ;Revision A                 7/28/93
                          3    ;This assembly code implements the serial interface
                          4    ;to the NM93CS06/46 serial EEPROM with Chip Security
                          5    ;through the SPI interface and the rest of Port D.
                          6    ;There are 10 commands:
                          7    ;   WRITE                   Protect Reg WRite (PRWR)
                          8    ;   READ                    Protect Reg ReaD (PRRD)
                          9    ;   WRite ENable (WREN)     Protect Reg ENable (PREN)
                         10    ;   WRite DiSable (WRDS)    Protect Reg DiSable (PRDS)
                         11    ;   WRite ALL (WRALL)       Protect Reg CLeaR (PRCLR)
                         12    ;Each EEPROM command type is implemented as a
                         13    ;subroutine call. Address is passed to the routine
                         14    ;in XADDR, data in XDATLO and XDATHI. Data is
                         15    ;returned from read operations in RDATLO and RDATHI.
                         16
0000                     17    $ include “header.asm”
0000                     18    PORTD     equ $1008    ;Port D data Register
0000                     19    DDRD      equ $1009    ;D Data Direction Register
0000                     20    SPCR      equ $1028    ;SPI Control Register
0000                     21    SPSR      equ $1029    ;SPI Status Register
0000                     22    SPDR      equ $102a    ;SPI Data Register
0000                     23    HPRIO     equ $103c    ;Interrupt Priority
                         24
0000                     25    XMESS0    equ $00      ;Transmit Message Buffer 0
0000                     26    XMESS1    equ $01      ;Transmit Message Buffer 1
0000                     27    XMESS2    equ $02      ;Transmit Message Buffer 2
0000                     28    XMESS3    equ $03      ;Transmit Message Buffer 3
0000                     29    MESSCT    equ $04      ;Message Byte Count
0000                     30    PREPE     equ $05      ;CS, Pre, Pe Bit settings
0000                     31    EXTWR     equ $0b      ;Set if write takes Extra Time
0000                     32    WRACTV    equ $0c      ;Set if write still active after
                         33                           ;message count goes to 00
                         34
0000                     35    XADDR     equ $06      ;Transmit Address
0000                     36    XDATLO    equ $07      ;Transmit Data low byte
0000                     37    XDATHI    equ $08      ;Transmit Data high bype
0000                     38    RDATLO    equ $09      ;Receive Data low byte
0000                     39    RDATHI    equ $0a      ;Receive Data high byte
                         40
0000                     41    $ include “init.asm”
                         42              ;Init intializes registers critical to SPI
                                                                    operation
                         43
C000                     44              org $c000          ;Start code at C000
C000  8E00FF             45    init1     lds #$00ff         ;Locate Stack at 00FF
C003  863B               46              ldaa #$3b          ;Initialize DDRD Register
C005  B71009             47              staa DDRD
C008  8653               48              ldaa #$53          ;Initialize SPCR Register
C00A  B71028             49              staa SPCR
C00D  8600               50              ldaa #$00          ;Zero out Port D Outputs
C00F  B71008             51              staa PORTD
C012  8603               52              ldaa #$03          ;Set SPI Interrupt to highest
                                                                    priority
C014  B7103C             53              staa HPRIO
C017  C6C0               54              ldab #$c0          ;Turn off global interrupt disable
C019  06                 55              tap
C01A  8600               56              ldaa #$00
```

```
C01C  9704              57              staa MESSCT       ;Message count is zero
C01E  970C              58              staa WRACTV       ;Write not active
                        59
C020  8656              60    main      ldaa #$56         ;Main give examples of using
                        61                                ;a few of the command rountines
C022  9706              62              staa XADDR        ;Load in transmit address
C024  8633              63              ldaa #$33
C026  9707              64              staa XDATLO       ;Load in low byte xmit dat
C028  86CC              65              ldaa #$cc
C02A  9708              66              staa XDATHI       ;Load in hi byte xmit data
C02C  BDC054            67              jsr WREN          ;Call Write Enable
C02F  BDC041            68              jsr PREN          ;Call Protect Register Enable
C032  BDC15C            69              jsr PRCLR         ;Call Protect Register Clear
C035  BDC041            70              jsr PREN          ;Call Protect Register Enable
C038  BDC13D            71              jsr PRWRT         ;Call Protect Register Write
C03B  BDC0CB            72              jsr PRRD          ;Call Protect Register Read
                        73                                ;(read back Protect Register)
C03E  7EC020            74              jmp main
                        75
C041  36                76    PrEn      psha
C042  37                77              pshb
C043  3C                78              pshx
C044  970B              79              pshy
C046  BDC17B            80              jsr WrPen         ;Subroutine Write Pending checks
                        81                                ;whether a previous write is still
                        82                                ;pending, and waits until it is
                        83                                ;done if there is.
C049  8623              84              ldaa #$23
C04B  9705              85              staa PREPE        ;Set CS, PRE, and PE bits
C04D  8600              86              ldaa #$00
C04F  970B              87              staa EXTWR        ;No extra time after command
C051  7EC061            88              jmp Enb
                        89
C054  36                90    WrEn      psha
C055  37                91              pshb
C056  3C                92              pshx
C057  183C              93              pshy
C059  BDC17B            94              jsr WrPen
C05C  8621              95              ldaa #$21
C05E  9705              96              staa PREPE        ;Set CS, PRE, and PE bits
C060  8600              97              ldaa #$00
C062  970B              98              staa EXTWR        ;No extra time after command
                        99
C064  8602             100    Enb       ldaa #$02
C066  9704             101              staa MESSCT       ;Load message byte count
C068  86C0             102              ldaa #$c0         ;Load last byte
C06A  9700             103              staa XMESS0       ;Data is 11000000
C06C  8604             104              ldaa #$04
C06E  9701             105              staa XMESS1       ;Load start bit & op code=00
C070  7EC184           106              jmp PstWr         ;Jump to Posted Write Routine
                       107
C073  36               108    PrDs      psha
C074  37               109              pshb
C075  3C               110              pshx
C076  183C             111              pshy
C078  BDC17B           112              jsr WrPen         ;Check Write still Pending?
C07B  8623             113              ldaa #$23
C07D  9705             114              staa PREPE        ;Set CS, PRE, and PE bits
```

```
C07F  8601             115              ldaa #$01
C081  970B             116              staa EXTWR        ;Extra write time required
C083  7EC096           117              jmp Dis
                       118
C086  36               119    WrDs      psha
C087  37               120              pshb
C088  3C               121              pshx
C089  183C             122              pshy
C08B  BDC17B           123              jsr WrPen         ;Check Write still Pending?
C08E  8621             124              ldaa #$21
C090  9705             125              staa PREPE        ;Set CS, PRE, and PE bits
C092  8600             126              ldaa #$00
C094  970B             127              staa EXTWR        ;No extra time after command
                       128
C096  8602             129    Dis       ldaa #$02
C097  9704             130              staa MESSCT       ;Load message byte count
C09A  8600             131              ldaa #$00         ;Load last byte
C09C  9700             132              staa XMESS0       ;Data = 0000 0000
C09E  8604             133              ldaa #$04
C0A0  9701             134              staa XMESS1       ;Load start bit & op code=00
C0A2  7EC184           135              jmp PstWr         ;jump to Posted Write Routine
                       136
C0A5  36               137    Read      psha
C0A6  37               138              pshb
C0A7  3C               139              pshx
C0A8  183C             140              pshy
C0AA  BDC17B           141              jsr WrPen         ;Check Write still Pending?
C0AD  8620             142              ldaa #$20
C0AF  9705             143              staa PREPE        ;Set CS, PRE, and PE bits
C0B1  8600             144              ldaa #$00
C0B3  970B             145              staa EXTWR        ;No extra time after command
C0B5  8604             146              ldaa #$04
C0B7  9704             147              staa MESSCT       ;Load message byte count
C0B9  8600             148              ldaa #$00
C0BB  9700             149              staa XMESS0       ;Load low byte data (don't care)
C0BD  9701             150              staa XMESS1       ;Load hi byte data (don't care)
C0BF  D606             151              ldab XADDR
C0C1  8606             152              ldaa #$06
C0C3  06               153              asld
C0C4  D702             154              staa XMESS2       ;Load address
C0C6  9703             155              staa XMESS3       ;Load start bit and op code=10
C0C8  7EC204           156              jmp PollRd        ;Jump to Polled Read Routine
                       157
C0CB  36               158    PrRd      psha
C0CC  37               159              pshb
C0CD  3C               160              pshx
C0CE  183C             161              pshy
C0D0  BDC17B           162              jsr WrPen         ;Check Write still Pending?
C0D3  8622             163              ldaa #$22
C0D5  9705             164              staa PREPE        ;Set CS, PRE, and PE bits
C0D7  8600             165              ldaa #$00
C0D9  970B             166              staa EXTWR        ;No extra time after command
C0DB  8603             167              ldaa #$03
C0DD  9704             168              staa MESSCT       ;Load message byte count
C0DF  8600             169              ldaa #$00
C0E1  9700             170              staa XMESS0       ;Load low byte (don't care)
C0E3  D606             171              ldaa XADDR
C0E5  8606             172              ldaa #$06
```

```
C0E7  05              173             asld
C0E8  D701            174             stab XMESS1       ;Load address byte
C0EA  9702            175             stab XMESS2       ;Load start bit op code=10
C0EC  7EC204          176             jmp PollRd        ;Jump to Polled Read Routine
                      177
C0EF  36              178   Write     psha
C0F0  37              179             pshb
C0F1  3C              180             pshx
C0F2  183C            181             pshy
C0F4  BDC17B          182             jsr WrPen         ;Check Write still Pending?
C0F7  8621            183             ldaa #$21
C0F9  9705            184             staa PREPE        ;Set CS, PRE, and PE bits
C0FB  8601            185             ldaa #$01
C0FD  970B            186             staa EXTWR        ;Extra time after command
C0FF  8604            187             ldaa #$04
C101  9704            188             staa MESSCT       ;Load message byte count
C103  8607            189             ldaa XDATLO
C105  9700            190             staa XMESS0       ;Load low xmit data byte
C107  9608            191             ldaa XDATHI
C109  9701            192             staa XMESS1       ;Load high xmit data byte
C10B  9606            193             ldaa XADDR
C10D  9702            194             staa XMESS2       ;Load address byte
C10F  8605            195             ldaa #$05
C111  9703            196             staa XMESS3       ;Load start bit & op code=01
C113  7EC184          197             jmp PstWr         ;Jump to Posted Write Routine
                      198
C114  36              199   WrAll     psha
C117  37              200             pshb
C118  3C              201             pshx
C119  183C            202             pshy
C11B  BDC17B          203             jsr WrPen         ;Check if Write still Pending?
C11E  8621            204             ldaa #$21
C120  9705            205             staa PREPE        ;Set CS, PRE, and PE bits
C122  8601            206             ldaa #$01
C124  970B            207             staa EXTWR        ;Extra time after command
C126  8604            208             ldaa #$04
C128  9704            209             staa MESSCT       ;Load message byte count
C12A  8607            210             ldaa XDATLO
C12C  9700            211             staa XMESS0       ;Load low xmit data byte
C12E  9608            212             ldaa XDATHI
C130  9701            213             staa XMESS1       ;Load high xmit data byte
C132  8640            214             ldaa #$40
C134  9702            215             staa XMESS2       ;Load address 0100 0000
C136  8604            216             ldaa #$04
C138  9703            217             staa XMESS3       ;Load start bit & op code=00
C13A  7EC184          218             jmp PstWr         ;Jump to Posted Write Routine
                      219
C13D  36              220   PrWrt     psha
C13E  37              221             pshb
C13F  3C              222             pshx
C140  183C            223             pshy
C142  BDC17B          224             jsr WrPen         ;Check if Write still Pending?
C145  8623            225             ldaa #$23
C147  9705            226             staa PREPE        ;Set CS, PRE, and PE bits
C149  8601            227             ldaa #$01
C14B  970B            228             staa EXTWR        ;Extra time after command
C14D  8602            229             ldaa #$02
C14F  9704            230             staa MESSCT       ;Load message byte count
```

```
C151  9606            231             ldaa XADDR
C153  9700            232             staa XMESS0       ;Load address byte
C155  8605            233             ldaa #$05
C157  9701            234             staa XMESS1       ;Load start bit & op code=01
C159  7EC184          235             jmp PstWr         ;Jump to Posted Write Routine
                      236
C15C  36              237   PrClr     psha
C15D  37              238             pshb
C15E  3C              239             pshx
C15F  183C            240             pshy
C161  BDC17B          241             jsr WrPen         ;Check if Write still Pending?
C164  8623            242             ldaa #$23
C166  9705            243             staa PREPE        ;Set CS, PRE, and PE bits
C168  8601            244             ldaa #$01
C16A  970B            245             staa EXTWR        ;Extra time after command
C16C  8602            246             ldaa #$02
C16E  9704            247             staa MESSCT       ;Load message byte count
C170  86FF            248             ldaa #$ff
C172  9700            249             staa XMESS0       ;Load address byte of all one's
C174  8607            250             ldaa #$07
C176  9701            251             staa XMESS1       ;Load start bit & op code=11
C178  7EC184          252             jmp PstWr         ;Jump to Posted Write Routine
                      253
C17B  9604            254   WrPen     ldaa MESSCT       ;Write Pending Subroutine:
C17D  26FC            255             bne WrPen         ;Check if previous message byte
                      256                               ;count is 0 before preceding
C17F  960C            257             ldaa WRACTV       ;Check write still active even
C181  26F8            258             bne WrPen         ;tho last message sent
C183  39              259             rts               ;Note that a bad write (ie. WREN
                      260                               ;has not been sent) will not return
                      261                               ;a ready so this loop will never
                      262                               ;end... a timer (approx 10ms) would
                      263                               ;normally be implemented to
                      264                               ;monitor this.
                      265
C184  8680            266   PstWr     ldaa #$80
C186  BA1028          267             oraa SPCR
C189  B71028          268             staa SPCR         ;Enable SPI Interrupt
C18C  D604            269             ldab MESSCT       ;Message byte count in B
C18E  CE0000          270             ldx #XMESS0       ;Address of last message byte
C191  B61008          271             ldaa PORTD        ;Read Port D
C194  9A05            272             oraa PREPE
C196  B71008          273             staa PORTD        ;Set CS, PRE, and PE lines of PORT D
C199  5A              274             decb              ;decrement message indexing B
C19A  3A              275             abx               ;index address in X to mess byte
C19B  A600            276             ldaa 0,x          ;Load A with message byte data-
C19D  B7102A          277             staa SPDR         ;Send data packet (start bit)
C1A0  1838            278             puly
C1A2  38              279             pulx
C1A3  33              280             pulb
C1A4  32              281             pula
C1A5  39              282             rts               ;Return to main program...the
                      283                               ;rest of the message is sent
                      284                               ;by interrupt routine SpInt
                      285
C1A6  F61029          286   SpInt     ldab SPSR         ;Check that interrupt cause by
C1A9  2A58            287             bpl done          ;SPIF (not mode error)
C1AB  B6102A          288             ldaa SPDR         ;Clear SPIT interrupt
```

```
C1AE  D604            289             ldab MESSCT       ;Load message byte count in B
C1B0  272D            290             beq Twp           ;Jump to post write poller
C1B2  5A              291             decb              ;Decrement message count
C1B3  D704            292             stab MESSCT       ;Store that 1 byte was sent
C1B5  270D            293             beq SsOff         ;Branch if last byte was sent
C1B7  CE0000          294             ldx #XMESS0       ;Load address of last byte in X
C1BA  5A              295             decb              ;Decrement message index B
C1BB  3A              296             abx               ;Index address in X to next byte
C1BC  A600            297             ldaa 0,x          ;Load next message byte
C1BE  B7102A          298             staa SPDR         ;Send message byte
C1C1  7EC203          299             jmp Done          ;Return to Main Program
C1C4  B61008          300   SsOff     ldaa PORTD
C1C7  84DF            301             anda #$dF
C1C9  B71008          302             staa PORTD        ;Turn off CS
C1CC  84FC            303             anda #$fc
C1CE  B71008          304             staa PORTD        ;Turn off PRE, PE
C1D1  960B            305             ldaa EXTWR        ;Zero out Extra Write time
C1D3  2716            306             beq Off           ;Jump to turn off interrupt
C1D5  B61008          307             ldaa PORTD
C1D8  8A20            308             oraa #$20
C1DA  B71008          309             staa PORTD        ;Turn on CS
C1DD  8600            310             ldaa #$00
C1DF  81FF            311   Twp       cmpa #$ff         ;Check if write complete
C1E1  2617            312             bne Send          ;No, send another idle byte
C1E3  B61008          313             ldaa PORTD
C1E6  84DF            314             anda #$df
C1E8  B71008          315             staa PORTD        ;Turn off CS
C1EB  867F            316   Off       ldaa #$7f
C1ED  B41028          317             anda SPCR
C1F0  B71028          318             staa SPCR         ;Disable SPI interrupt
C1F3  8600            319             ldaa #$00
C1F5  970C            320             staa WRACTV       ;Write complete
C1F7  7EC203          321             jmp Done
C1FA  8601            322   Send      ldaa #$01
C1FC  970C            323             staa WRACTV
C1FE  8600            324             ldaa #$00
C200  B7102A          325             staa SPDR
C203  3B              326   Done      rti               ;Return to main program
                      327
C204  D604            328   PollRd    ldab MESSCT       ;Load message byte count in B
C206  CE0000          329             ldx #MESS0        ;Load address of last byte in X
C209  B61008          330             ldaa PORTD
C20C  9A05            331             oraa PREPE
C20E  B71008          332             staa PORTD        ;Set CS, PRE, and PE
C211  3A              333             abx               ;Index into address past byte
C212  09              334   SendX     dex               ;Decrement back to byte
C213  5A              335             decb              ;Decrement index B
C214  A600            336             ldaa 0,x          ;Load next message byte
C216  B7102A          337             staa SPDR         ;Send message byte
C219  B61029          338   Wait      ldaa SPSR         ;Load SPI Status register
C21C  2AFB            339             bpl Wait          ;Wait unit SPIF is set
C21E  B6102A          340             ldaa SPDR         ;Load in data to A
C221  C100            341             cmpb #$00         ;Last message byte?
C223  2705            342             beq StrLo
C225  970A            343             staa RDATHI       ;Save byte as hi receive byte
C227  7EC212          344             jmp SendX         ;Send/receive next byte
C22A  9709            345   StrLo     staa RDATLO       ;Store lo receive byte
C22C  D704            346             stab MESSCT       ;Zero out message byte count
```

```
C22E  86DC            347           ldaa #$dc
C230  B41008          348           anda PORTD
C233  B71008          349           staa PORTD        ;turn off CS, PRE, PE
C236  1838            350           puly
C238  38              351           pulx
C239  33              352           pulb
C23A  32              353           pula
C23B  39              354           rts               ;Return to Main Program
                      355
FFD8                  356           org $ffd8
FFD8  C1A6            357           fdb SpInt         ;Set SPI interrupt vector to
                      358                             ;point to SPI interrupt service
                      359                             ;routine
                      360
                      361

Symbol Table
DDRD                1009
DIS                 C096
DONE                C203
ENB                 C064
EXTWR               000B
HPRIO               103C
INIT1               C000
MAIN                C020
MESSCT              0004
OFF                 C1EB
POLLRD              C204
PORTD               1008
PRCLR               C15C
PRDS                C073
PREN                C041
PREPE               0005
PRRD                C0CB
PRWRT               C13D
PSTWR               C184
RDATHI              000A
RDATLO              0009
READ                C0A5
SEND                C1FA
SENDX               C212
SPCR                1028
SPDR                102A
SPINT               C1A6
SPSR                1029
SSOFF               C1C4
STRLO               C22A
TWP                 C1DF
WAIT                C219
WRACTV              000C
WRALL               C116
WRDS                C086
WREN                C054
WRITE               C0EF
WRPEN               C17B
XADDR               0006
XDATHI              0008
XDATLO              0007
XMESS0              0000
XMESS1              0001
XMESS2              0002
XMESS3              0003
```

Life Support Policy

Fairchild's products are not authorized for use as critical components in life support devices or systems without the express written approval of the President of Fairchild Semiconductor Corporation. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Fairchild Semiconductor
Americas
Customer Response Center
Tel. 1-888-522-5372

Fairchild Semiconductor
Europe
Fax: +44 (0) 1793-856858
Deutsch Tel: +49 (0) 8141-6102-0
English Tel: +44 (0) 1793-856856
Français Tel: +33 (0) 1-6930-3696
Italiano Tel: +39 (0) 2-249111-1

Fairchild Semiconductor
Hong Kong
8/F, Room 808, Empire Centre
68 Mody Road, Tsimshatsui East
Kowloon. Hong Kong
Tel; +852-2722-8338
Fax: +852-2722-8383

Fairchild Semiconductor
Japan Ltd.
4F, Natsume Bldg.
2-18-6, Yushima, Bunkyo-ku
Tokyo, 113-0034 Japan
Tel: 81-3-3818-8840
Fax: 81-3-3818-8841