# Interfacing Xicor SPI Serial Memories to Hitachi H8/3000 Microcontrollers

*by Applications Staff*

This code shows how the X25650 family of advanced SPI serial EEPROMs can be interfaced to the Hitachi H8/3000 microcontroller family when connected as shown in Figure 1. The interface uses four of the port pins available on H8 family devices to implement the interface. This interface was tested with the X25650, however by using only part of the memory array, this code can be easily adapted for lower density memories.

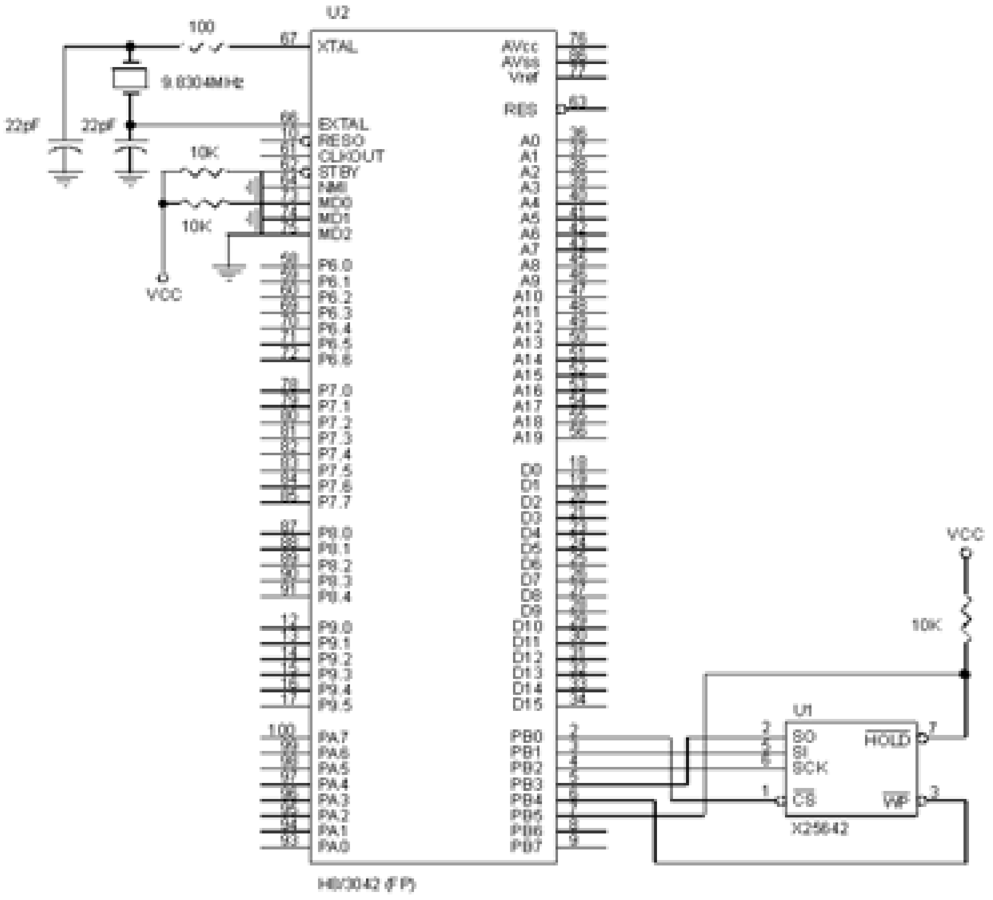

**Figure 1. Typical hardware connection for interfacing a X25650 to the H8/3042 microcontroller**

```
        .h8300h
        .file "x25650.asm"

;  x25650.asm

;  Test code for interfacing Xicor X25080/160/320/650/138 Serial EEPROMS
;  to the Hitachi H8/3042

;  Revision: 1.04  02/04/96

;  Expects the H8/3042 hard-wired in Mode 1: 8-bit bus-width, Expanded address
;  space (0x00000-0xFFFFF), Internal ROM disabled, Internal RAM enabled
;  (0xFF710-0xFFF0F), 9.8304 MHx xtal

      .include "h8stddef.inc"       ; H8/3042 standard register definitions

;  H8/3042 specific equates

      .equ  H8RAMTOP,   0xFFF0F     ; Highest onboard RAM address (2048 bytes)
      .equ  H8RAMBOT,   0xFF710     ; Lowest onboard RAM address
      .equ  RXD0,       bit_2       ; Receive data pin of port/SCI
      .equ  INTMASK,    0b11000000  ; CCR Interrupt Mask (I and UI bits)

;  Xicor EEPROM device-specific equates

      .equ  CS,         bit_0       ; Port bit for Chip Select (H8 out)
      .equ  SI,         bit_1       ; Port bit for Serial data Input (H8 out)
      .equ  SCK,        bit_2       ; Port bit for Serial ClocK (H8 out)
      .equ  SO,         bit_3       ; Port bit for Serial data Output (H8 in)
      .equ  WP,         bit_4       ; Port bit for Write Protect input (H8 out)
      .equ  HOLD,       bit_5       ; Port bit for HOLD input (H8 out)

      .equ  XICOR,      PBDR        ; H8/3042 data port assignment (PB)
      .equ  XICORDDR,   PBDDR       ; H8/3042 data direction port assignment
      .equ  DDRSETUP,   0b11110111  ; Port Data Direction Register setup

      .equ  WREN,       0x06        ; WRite ENable latch instruction
      .equ  WRDI,       0x04        ; WRite DIsable latch instruction
      .equ  WRSR,       0x01        ; WRite Status Register instruction
      .equ  RDSR,       0x05        ; ReaD Status Register instruction
      .equ  WRITE,      0x02        ; WRITE memory instruction
      .equ  READ,       0x03        ; READ memory instruction

      .equ  WIPBIT,     bit_0       ; Write In Progress status bit
      .equ  WPEN,       bit_7       ; Write Protect ENable status bit
      .equ  PAGESIZE,   32          ; Bytes per page
      .equ  NUMPAGES,   512         ; Number of pages in EEPROM
      .equ  PAGEMASK,   PAGESIZE-1  ; Mask out non-page bits
      .equ  PAGEBNDRY,  ~ PAGEMASK  ; Mask out page bits (NOT PAGEMASK)
      .equ  NUMBYTES,   NUMPAGES*PAGESIZE ; Number of bytes in EEPROM
      .equ  ADDRMASK,   NUMBYTES-1  ; Mask out non-address bits
      .equ  MAXPOLLS,   100         ; Maximum number of poll attempts for WIP

;  Main equates
```

```
      .equ  STACKTOP,   H8RAMTOP-4  ; Stack initializes here and builds down
      .equ  ALLONES,    0b11111111  ; All bits set
      .equ  BIGWRITE,   170         ; Test number of bytes to write (>5 pages)
      .equ  RAMBFR0,    H8RAMBOT    ; Start of read/write buffer in on-board RAM
      .equ  RAMBFR1,    H8RAMBOT+PAGESIZE ; Second read/write buffer
      .equ  RAMBFR2,    RAMBFR1+BIGWRITE ; Third read/write buffer
      .equ  RAMBFR3,    RAMBFR2+BIGWRITE ; Fourth read/write buffer
      .equ  DATABYTE0,  0xF0        ; Test byte 0
      .equ  DATABYTE1,  0x0F        ;           1
      .equ  DATABYTE2,  0x55        ;           2
      .equ  DATABYTE3,  0xAA        ;           3
      .equ  XADDRB,     155         ; Test byte write/read address
      .equ  TESTPAGE,   12          ; Test page number
      .equ  XADDRP,     TESTPAGE*PAGESIZE ; Test page write/read address
      .equ  XADDRBP,    NUMBYTES-10 ; Test block protect read/write address
      .equ  XADDRBW,    14*PAGESIZE ; EEPROM BigWrite destination

;     Start of code

;     .org  0x00000                 ; H8/3042 Interrupt vector jump table
                                    ; Must reside at address 0x00000

      .long 0x00000100              ;  0 RESET
      .long _bogus_int              ;  1 reserved
      .long _bogus_int              ;  2 reserved
      .long _bogus_int              ;  3 reserved
      .long _bogus_int              ;  4 reserved
      .long _bogus_int              ;  5 reserved
      .long _bogus_int              ;  6 reserved
      .long _bogus_int              ;  7 external interrupt (NMI)
      .long _bogus_int              ;  8 Trap Instruction (4 sources)
      .long _bogus_int              ;  9 Trap Instruction (4 sources)
      .long _bogus_int              ; 10 Trap Instruction (4 sources)
      .long _bogus_int              ; 11 Trap Instruction (4 sources)
      .long _bogus_int              ; 12 External Interrupt IRQ0
      .long _bogus_int              ; 13 External Interrupt IRQ1
      .long _bogus_int              ; 14 External Interrupt IRQ2
      .long _bogus_int              ; 15 External Interrupt IRQ3
      .long _bogus_int              ; 16 External Interrupt IRQ4
      .long _bogus_int              ; 17 External Interrupt IRQ5
      .long _bogus_int              ; 18 reserved
      .long _bogus_int              ; 19 reserved
      .long _bogus_int              ; 20 WOVI Watchdog timer
      .long _bogus_int              ; 21 CMI  Refresh controller
      .long _bogus_int              ; 22 reserved
      .long _bogus_int              ; 23 reserved
      .long _bogus_int              ; 24 IMIA0 GRA0 compare match/input capture
      .long _bogus_int              ; 25 IMIB0 GRB0 compare match/input capture
      .long _bogus_int              ; 26 OVI0  overflow 0
      .long _bogus_int              ; 27 reserved
      .long _bogus_int              ; 28 IMIA1 GRA1 compare match/input capture
      .long _bogus_int              ; 29 IMIB1 GRB1 compare match/input capture
      .long _bogus_int              ; 30 OVI1  overflow 1
      .long _bogus_int              ; 31 reserved
      .long _bogus_int              ; 32 IMIA2 GRA2 compare match/input capture
      .long _bogus_int            ; 33 IMIB2 GRB2 compare match/input capture
      .long _bogus_int            ; 34 OVI2  overflow 2
      .long _bogus_int            ; 35 reserved
      .long _bogus_int            ; 36 IMIA3 GRA3 compare match/input capture
      .long _bogus_int            ; 37 IMIB3 GRB3 compare match/input capture
      .long _bogus_int            ; 38 OVI3  overflow 3
      .long _bogus_int            ; 39 reserved
      .long _bogus_int            ; 40 IMIA4 GRA4 compare match/input capture
      .long _bogus_int            ; 41 IMIB4 GRB4 compare match/input capture
      .long _bogus_int            ; 42 OVI4 overflow 4
      .long _bogus_int            ; 43 reserved
      .long _bogus_int            ; 44 DEND0A  DMAC group 0
      .long _bogus_int            ; 45 DEND0B  DMAC group 0
      .long _bogus_int            ; 46 DEND1A  DMAC group 0
      .long _bogus_int            ; 47 DEND1B  DMAC group 0
      .long _bogus_int            ; 48 reserved
      .long _bogus_int            ; 49 reserved
      .long _bogus_int            ; 50 reserved
      .long _bogus_int            ; 51 reserved
      .long _bogus_int            ; 52 ERI0 receive error       SCI chan 0
      .long _bogus_int            ; 53 RXI0 receive data full   SCI chan 0
      .long _bogus_int            ; 54 TXI0 transmit data empty SCI chan 0
      .long _bogus_int            ; 55 TEI0 transmit end        SCI chan 0
      .long _bogus_int            ; 56 ERI1 receive error       SCI chan 1
      .long _bogus_int            ; 57 RXI1 receive data full   SCI chan 1
      .long _bogus_int            ; 58 TXI1 transmit data empty SCI chan 1
      .long _bogus_int            ; 59 TEI1 transmit end        SCI chan 1
      .long _bogus_int            ; 60 ADI  A/D end

_bogus_int:
      orc         #0b11000000,ccr   ; Disable interrupts (I and UI)
      rte

;     .org        0x000100

_powerup:                           ; Initialization
      orc         #INTMASK,ccr      ; Disable interrupts (I and UI)
      mov.l       #STACKTOP,er7     ; Initialize the Stack pointer
      bsr         _init_xiport:16   ; Initialize the EEPROM I/O port
      bset        #WP,@XICOR        ; Write Protect line high to enable EEPROM

_test_start:
      bsr         _wrdi_cmnd:16     ; Disable writing to the EEPROM
      bsr         _rdsr_cmnd:16     ; Read status register
      bsr         _wren_cmnd:16     ; Enable writing to the EEPROM
      bsr         _rdsr_cmnd:16     ; Read status register

;  Write a single byte to the EEPROM, then read it back

      bsr         _wren_cmnd:16     ; Enable writing to the EEPROM
      mov.b       #DATABYTE3,r0l
      mov.w       #XADDRB,r1        ; Setup target test address
      bsr         _byte_write:16    ; Write byte

      mov.w       #XADDRB,r1        ; Setup the same address just written
      bsr         _byte_read:16     ; Read byte

;  Write a single page to the EEPROM, then read it back

      bsr         _wren_cmnd:16     ; Enable writing to the EEPROM
      mov.w       #XADDRP,r1        ; Setup EEPROM target page address
      mov.l       #-datase0,er2     ; Point to source data
      bsr         _page_write:16    ; ... and write the page

      mov.w       #XADDRP,r1        ; Again, setup EEPROM source page address
      mov.l       #RAMBFR1,er2      ; Point to destination buffer in onboard RAM
      mov.w       #PAGESIZE,e1      ; Setup byte read count
      bsr         _read_seq:16      ; ... and read the page

;  Write a sequence of bytes to the EEPROM (multiple pages), then read them back

;  First, setup a byte sequence in RAM

      xor.w       r1,r1             ; Initialize byte sequence/count
      mov.l       #RAMBFR1,er2      ; ... and point to the destination
_bigwr_setup:
      mov.b       r1l,r0l           ; Fetch byte to store
      xor.b       r6h,r0l           ; Complement if required
      mov.b       r0l,@er2          ; Store a byte in the RAM buffer
      inc.l       #1,er2            ; Bump pointer to next RAM location
      inc.w       #1,r1             ; Count bytes stored/generate next byte
      cmp.w       #BIGWRITE,r1      ; Have we done 'em all?
      blt         _bigwr_setup      ;  No, continue

;  Write the byte sequence

      mov.w       #BIGWRITE,e1      ; Setup byte count
      mov.w       #XADDRBW,r1       ; Setup EEPROM starting address
      mov.l       #RAMBFR1,er2      ; Setup EEPROM 'big write' (count is in e1)
      bsr         _wren_cmnd:16     ; Enable writing to the EEPROM
      bsr         _write_seq:16     ; ... and write 'em

; Read the byte sequence

      mov.w       #BIGWRITE,e1      ; Setup to read 'em back!
      mov.w       #XADDRBW,r1
      mov.l       #RAMBFR2,er2
      bsr         _read_seq:16

; Spin a tight loop ... wait for a reset to do it over

_endloop:
      bra         _endloop

; EEPROM Interface Subroutines

; Name:       _init_xiport
; Function:   Initializes the H8 I/O port bit directions
; Calls:      None
; Expects:    Nothing
; Returns:   Nothing
; Registers: r0l
; Remarks    Must be called once initially to setup the I/O port

_init_xiport:
      mov.b       #DDRSETUP,r0l     ; Setup Port data directions
      mov.b       r0l,@XICORDDR     ; ... and load 'em
      bset        #CS,@XICOR        ; Chip Select high (disable EEPROM)
      bclr        #SI,@XICOR        ; Drop Serial Input to EEPROM
      bclr        #SCK,@XICOR       ; Drop the clock SCK
      bclr        #WP,@XICOR        ; Write Protect line low, disable writes
      bset        #HOLD,@XICOR      ; HOLD line high, enable writes
      rts

 Name:       _wren_cmnd
; Function:  Sends the command to enable writing to the Xicor EEPROM
; Calls:     _send_byte
; Expects:   Nothing
; Returns:   Nothing
; Registers: r0l
; Remarks:   Note that this command must preceed each write sequence

_wren_cmnd:
      bclr        #SCK,@XICOR       ; SCK low
      bclr        #CS,@XICOR        ; CS low
      mov.b       #WREN,r0l         ; Setup Write Enable instruction
      bsr         _send_byte:16
      bclr        #SCK,@XICOR       ; SCK low
      bset        #CS,@XICOR        ; CS high
      rts

; Name:      _wrdi_cmnd
; Function:  Sends the command to disable writing to the Xicor EEPROM
; Calls:     _send_byte
; Expects:   Nothing
; Returns:   Nothing
; Registers: r0l
; Remarks:

_wrdi_cmnd:
      bclr        #SCK,@XICOR       ; SCK low
      bclr        #CS,@XICOR        ; CS low
      mov.b       #WRDI,r0l         ; Setup Write Disable instruction
      bsr         _send_byte:16
      bclr        #SCK,@XICOR       ; SCK low
      bset        #CS,@XICOR        ; CS high
      rts

; Name:      _wrsr_cmnd
; Function:  Sends the command which enables writing to the BP0 and
;            BP1 bits of the Xicor EEPROM status register
; Calls:     _send_byte, _poll_write
; Expects:   Block Protect bits in r0l
; Returns:   Nothing
; Registers: r0l, e0
; Remarks:

_wrsr_cmnd:
      mov.w       r0,e0              ; Temporarily preserve Block Protect bits
      bclr        #SCK,@XICOR        ; SCK low
      bclr        #CS,@XICOR         ; CS low
      mov.b       #WRSR,r0l          ; Setup Write Status Register instruction
      bsr         _send_byte:16
      mov.w       e0,r0              ; Recover Block Protect bits
      bsr         _send_byte:16
      bclr        #SCK,@XICOR        ; SCK low
      bset        #CS,@XICOR         ; CS high
      bsr         _poll_write:16
      rts


; Name:      _rdsr_cmnd
; Function:  Sends the command which reads the contents of the Xicor EEPROM
;            EEPROM status register
; Calls:     _send _byte, _recv_byte
; Expects:   Nothing
; Returns:   Status in r0l
; Registers: r0l
; Remarks:

_rdsr_cmnd:
      bclr        #SCK,@XICOR        ; SCK low
      bclr        #CS,@XICOR         ; CS low
      mov.b       #RDSR,r0l          ; Setup Read Status Register instruction
      bsr         _send_byte:16
      bsr         _recv_byte:16
      bclr        #SCK,@XICOR        ; SCK low
      bset        #CS,@XICOR         ; CS high
      rts

; Name:      _byte_write
; Function:  Writes a single byte to the Xicor EEPROM memory array
; Calls:     _send_byte, _poll_write, _send_word
; Expects:   Byte to be sent in r0l, address in r1
; Returns:   Nothing
; Registers: r0l, r1, e0
; Remarks:

_byte_write:
      bclr        #SCK,@XICOR        ; SCK low
      bclr        #CS,@XICOR         ; CS low
      mov.w       r0,e0              ; Temporarily preserve the byte
      mov.b       #WRITE,r0l         ; Setup Write instruction
      bsr         _send_byte:16      ; Send it
      bsr         _send_word:16      ; Send address
      mov.w       e0,r0              ; Recover the byte
      bsr         _send_byte:16
      bclr        #SCK,@XICOR        ; SCK low
      bset        #CS,@XICOR         ; CS high
      bsr         _poll_write:16
      rts

; Name:      _byte_read
; Function:  Reads a single byte from the Xicor EEPROM memory array
; Calls:     _send_byte, _recv_byte _send_word
; Expects:   Serial EEPROM byte address in r1 (0x0000-0x01FF)
; Returns:   Byte read in r0l
; Registers: r0l, r1, e0
; Remarks:

_byte_read:
      bclr        #SCK,@XICOR       ; SCK low
      bclr        #CS,@XICOR        ; CS low
      mov.b       #READ,r0l         ; Setup Read instruction
      bsr         _send_byte:16     ; Send it
      bsr         _send_word:16     ; Send address
      bsr         _recv_byte:16     ; Receive byte from EEPROM
      bclr        #SCK,@XICOR       ; SCK low
      bset        #CS,@XICOR        ; CS high
      rts
; Name:      _page_write
; Function:  Sends a full page (32 bytes) to the Xicor EEPROM
; Calls:     _send_byte, _poll_write _send_word
; Expects:   Serial EEPROM destination starting address in r1, pointer to
;            source bytes (32) in er2
; Returns:   Nothing
; Registers: r0l, r0h, r1, er2, r3l
; Remarks:

_page_write:
      bclr        #SCK,@XICOR       ; SCK low
      bclr        #CS,@XICOR        ; CS low
      mov.b       #WRITE,r0l        ; Setup Write instruction
      bsr         _send_byte:16     ; Send it
      bsr         _send_word:16     ; Send address
      mov.b       #PAGESIZE,r3l
_page_wrloop:
      mov.b       @er2+,r0l         ; Fetch byte to send, point to next one
      bsr         _send_byte:16
      dec.b       r3l               ; Click off another sent byte
      bne         _page_wrloop
      bclr        #SCK,@XICOR       ; SCK low
      bset        #CS,@XICOR        ; CS high
      bsr         _poll_write:16
      rts

; Name:      _read_seq
; Function:  Reads a sequence of bytes from the Xicor EEPROM Serial EEPROM
; Calls:     _send_byte, _recv_byte _send_word
; Expects:   EEPROM source starting address in r1, count of
;            bytes to read in e1, pointer to start of destination storage in er2
; Returns:   Byte array read in memory
; Registers: r0l, r0h, r1, e1, er2
; Remarks:
```

```
_read_seq:
      bclr        #SCK,@XICOR       ; SCK low
      bclr        #CS,@XICOR        ; CS low
      mov.b       #READ,r0l         ; Setup Read instruction
      bsr         _send_byte:16     ; ... and send it
      bsr         _send_word:16     ; Send address
_read_seqloop:
      bsr         _recv_byte:16     ; Fetch byte from EEPROM
      mov.b       r0l,@er2          ; Store EEPROM byte into memory
      inc.l       #1,er2            ; Point to next storage location
      dec.w       #1,e1             ; Click off another byte read
      bne         _read_seqloop     ; Continue if not zero ...
      bclr        #SCK,@XICOR       ; SCK low
      bset        #CS,@XICOR        ; CS high
      rts
 Name:      _write_seq
; Function:  Writes a sequence of bytes to the Xicor EEPROM Serial EEPROM
; Calls:     _send_byte, _send_word, _poll_write
; Expects:   EEPROM destination starting address in r1, count of bytes to write
;            in e1, pointer to start of source storage in er2
; Returns:   Nothing
; Registers: r0l, r0h, r1, e1, er2, r3
; Remarks:   Takes advantage of page write mode to minimize write times

_write_seq:
      bclr        #SCK,@XICOR       ; SCK low
      bclr        #CS,@XICOR        ; CS low
      mov.b       #WRITE,r0l        ; Setup Write instruction
      bsr         _send_byte:16     ; Send it
      mov.w       r1,e3             ; preserve EEPROM destination address
      bsr         _send_word:16     ; Send address
_write_seqloop:
      mov.b       @er2+,r0l         ; Fetch byte to write & point to next one
      bsr         _send_byte:16
      dec.w       #1,e1             ; Click off another byte written
      inc.w       #1,e3             ; Point to next destination address
      mov.w       e3,r3             ; ... and scratchpad it
      and.b       #PAGEMASK,r3l     ; Keep destination address page bits
      beq         _wrseq_pagend     ; Zero means prior address was page end
      or.w        e1,e1             ; Is the byte count zero?
      bne         _write_seqloop    ;   No, continue with this page
      bclr        #SCK,@XICOR       ;   Yes, set SCK low
      bset        #CS,@XICOR        ; CS high
      bsr         _poll_write:16    ; Is the write still in progress in EEPROM?
      rts
_wrseq_pagend:
      bclr        #SCK,@XICOR       ; SCK low
      bset        #CS,@XICOR        ; CS high
      bsr         _poll_write:16    ; Is the write still in progress in EEPROM?
      or.w        e1,e1             ; Is the byte count zero?
      beq         _wrseq_done       ;   Yes, we're done
      bsr         _wren_cmnd:16     ;   No, more to do so re-enable writing
      mov.w       e3,r1             ; Recover next EEPROM address as expected
      bra         _write_seq        ; Back to the EEPROM
_wrseq_done:
      rts

; Name:      _send_byte
; Function:  Sends a byte to the Xicor EEPROM, serially shifting MSB first
; Calls:     None
; Expects:   Byte to be sent in r0l
; Returns:   Nothing
; Registers: r0l, r0h
; Remarks:

_send_byte:
      mov.b       #8,r0h             ; Setup bit count
_send_loop:
      bclr        #SCK,@XICOR        ; SCK low
      rotxl.b     r0l                ; Slip next MSB into Carry
      bst         #SI,@XICOR         ; Copy Carry to I/O port bit
      bset        #SCK,@XICOR        ; SCK high
      dec.b       r0h                ; Click off a bit
      bne         _send_loop         ; Continue if not done
      bclr        #SI,@XICOR         ; SI low
      rts

; Name:      _send_word
; Function:  Sends a word to the Xicor EEPROM, serially shifting MSB first
; Calls:     None
; Expects:   Word to be sent in r1
; Returns:   Nothing
; Registers: r0h, r1
; Remarks:

_send_word:
      mov.b       #16,r0h            ; Setup bit count
_word_loop:
      bclr        #SCK,@XICOR        ; SCK low
      rotxl.w     r1                 ; Slip next MSB into Carry
      bst         #SI,@XICOR         ; Copy Carry to I/O port bit
      bset        #SCK,@XICOR        ; SCK high
      dec.b       r0h                ; Click off a bit
      bne         _word_loop         ; Continue if not done
      bclr        #SI,@XICOR         ; SI low
      rts

; Name:      _recv_byte
; Function:  Receives a byte from the Xicor EEPROM, serially shifting MSB first
; Calls:     None
; Expects:   Nothing
; Returns:   Received byte in r0l
; Registers: r0l, r0h
; Remarks:   Clock rate limited for H8/3042 16MHz xtal

_recv_byte:
      mov.b       #8,r0h             ; Setup bit count
_recv_loop:
      bset        #SCK,@XICOR        ; SCK high
      nop
      nop
      nop                                 ; Guarantee limited clock rate
      bclr        #SCK,@XICOR             ; SCK low
      bld         #SO,@XICOR              ; Copy input port bit to Carry
      rotxl.b     r0l                     ; Slip Carry into LSB
      dec.b       r0h                     ; Click off a bit
      bne         _recv_loop              ; Continue if not done
      rts

; Name:       _poll_write
; Function:   Polls for the completion of the non-volatile write cycle by
;             examining the Write-In-Progress bit of the status register
; Calls:      _rdsr_cmnd
; Expects:    Nothing
; Returns:    Nothing
; Registers:  r1l
; Remarks:    Polling delay count setup for H8/3042 16MHx xtal

_poll_write:
      mov.b       #MAXPOLLS,r1l           ; Setup maximum number of poll attempts
_poll_loop:
      bsr         _rdsr_cmnd:16           ; Fetch the EEPROM Status Register content
      btst        #WIPBIT,r0l             ; Is the Write-In-Progress bit zero?
      beq         _poll_loop1             ;  Yes, write  is complete
      dec.b       r1l                     ;  No, click off another poll attempt
      bne         _poll_loop              ; If we haven't exceeded maximum polls ...
_poll_loop1:
      rts

;  Data tables

      .align      0
_dataset0:
      .byte       0xAA,0x55,0xAA,0x55,0xAA,0x55,0xAA,0x55
      .byte       0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80
      .byte       0xAA,0x55,0xAA,0x55,0xAA,0x55,0xAA,0x55
      .byte       0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F
```