Interfacing the X25650 to 8051 Microcontrollers

by Applications Staff, August 1994

The following code demonstrates how Xicor's X25650 family of SPI serial E^2PROMs can be interfaced to the 8051 microcontroller family when connected as shown in Figure 1. The interface uses 4 lines from port 1, where CS is on P1.0, SI on P1.1, SCK on P1.2, and SO on P1.3. Additional code that will implement interfaces between the 8051 microcontroller family and other Xicor serial devices can be found on the Xicor web site at http://www.xicor.com.

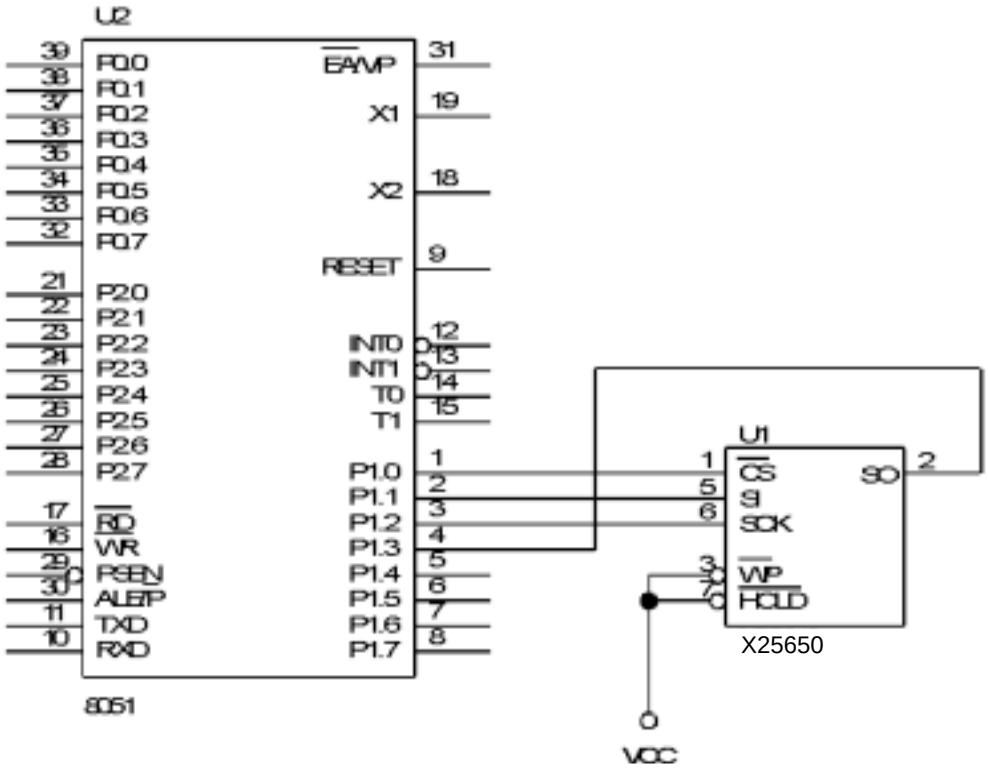

Figure 1. Typical hardware connection for interfacing an X25650 to 8051 microcontrollers

```
;*******************************************************************************
;* THIS CODE IS DESIGNED TO SHOW HOW THE XICOR X25650 SPI SERIAL EEPROM CAN
;* BE INTERFACED WITH THE 8051 MICROCONTROLLER. THE INTERFACE USES THE 8051 GENERAL
;* PURPOSE PARALLEL PORT 1 AND CONNECTS P1.0 TO THE CHIP SELECT LINE (/CS), P1.1 TO
;* THE SERIAL  INPUT DATA LINE (SI), P1.2 TO THE SERIAL CLOCK LINE (SCK) AND P1.3 TO
;* THE SERIAL OUTPUT DATA LINE (SO).
;*
;* THE COMMAND ROUTINES PROVIDED DEMONSTRATE THE PROTOCOL FOR ALL X25650 OPERATIONS.
;* THESE ARE:
;*
;*   1.  SET WRITE ENABLE LATCH
;*   2.  RESET WRITE ENABLE LATCH
;*   3.  WRITE STATUS REGISTER
;*   4.  READ STATUS REGISTER
;*   5.  SINGLE BYTE WRITE
;*   6.  SINGLE BYTE READ
;*   7.  PAGE WRITE
;*   8.  SEQUENTIAL READ
;*
;* THE PROGRAM WRITES 00H TO THE STATUS REGISTER; READS THE STATUS REGISTER; WRITES
;* 11H TO ADDRESS 555H IN BYTE MODE; PERFORMS A SINGLE BYTE READ FROM ADDRESS 555H;
;* WRITES 22H, 33H, 44H TO ADDRESSES 1F00H, 1F01H, 1F02H IN PAGE MODE; AND PERFORMS
;* A SEQUENTIAL READ FROM ADDRESSES 1F00H, 1F01H, 1F02H.
;*
;*******************************************************************************
*

CS              BIT   P1.0  ; port 1 bit 0 used for chip select (/CS)
SI              BIT   P1.1  ; port 1 bit 1 used for serial input (SI)
SCK             BIT   P1.2  ; port 1 bit 2 used for serial clock (SCK)
SO              BIT   P1.3  ; port 1 bit 3 used for serial output (SO)
WREN_INST       EQU   06H   ; write enable latch instruction (WREN)
WRDI_INST       EQU   04H   ; write disable latch instruction (WRDI)
WRSR_INST       EQU   01H   ; write status register instruction (WRSR)
RDSR_INST       EQU   05H   ; read status register instruction (RDSR)
WRITE_INST      EQU   02H   ; write memory instruction (WRITE)
READ_INST       EQU   03H   ; read  memory instruction (READ)
BYTE_ADDR       EQU   0555H ; memory address for byte mode operations
BYTE_DATA       EQU   11H   ; data byte for byte write operation
PAGE_ADDR       EQU   1F00H ; memory address for page mode operations
PAGE_DATA1      EQU   22H   ; 1st data byte for page write operation
PAGE_DATA2      EQU   33H   ; 2nd data byte for page write operation
PAGE_DATA3      EQU   44H   ; 3rd data byte for page write operation
STATUS_REG      EQU   00H   ; status register
MAX_POLL        EQU   99H   ; maximum number of  polls
INIT_STATE      EQU   01H   ; initialization value for control ports
SLIC            EQU   030H  ; address location of SLIC (rest of system code)
;***********************************
;* RESET VECTOR TO BEGINNING OF CODE
;***********************************

            ORG   0000H       ; reset vectors to this location
            LJMP  BEGIN
```

```
;*******************************
;* START OF PROGRAM EXECUTION
;*******************************

            ORG   0120H
BEGIN:      MOV   SP, #60H          ; initialize stack pointer
            CLR   EA                ; disable interupts
            MOV   P1, #INIT_STATE   ; init. control lines (CS high, SCK/SI/SO low)
            LCALL WREN_CMD          ; set write enable latch
            LCALL WRSR_CMD          ; write 00h to status register
            LCALL WREN_CMD          ; set write enable latch
            LCALL BYTE_WRITE        ; write 11h to address 555h (byte write)
            LCALL BYTE_READ         ; read from address location 555h (byte read)
            LCALL WREN_CMD          ; set write enable latch
            LCALL PAGE_WRITE        ; page write 22h/33h/44h to addresses 1F00/1/2h
            LCALL SEQU_READ         ; sequential read from addresses 1F00/1/2h
            JMP   SLIC

;***************************
;*  1. SET WRITE ENABLE LATCH
;***************************

WREN_CMD:   CLR   SCK               ; bring SCK low
            CLR   CS                ; bring CS low
            MOV   A, #WREN_INST
            LCALL OUTBYT            ; send WREN instruction
            CLR   SCK               ; bring SCK low
            SETB  CS                ; bring CS high
            RET

;*****************************
;*  2. RESET WRITE ENABLE LATCH
;*****************************

WRDI_CMD:   CLR   SCK               ; bring SCK low
            CLR   CS                ; bring CS low
            MOV   A,#WRDI_INST
            LCALL OUTBYT            ; send WRDI instruction
            CLR   SCK               ; bring SCK low
            SETB  CS                ; bring CS high
            RET

;*************************
;*  3. WRITE STATUS REGISTER
;*************************

WRSR_CMD:   CLR   SCK               ; bring SCK low
            CLR   CS                ; bring CS low
            MOV   A, #WRSR_INST
            LCALL OUTBYT            ; send WRSR instruction
            MOV   A, #STATUS_REG
            LCALL OUTBYT            ; send status register byte
            CLR   SCK               ; bring SCK low
```

```
            SETB  CS                ; bring CS high
            LCALL WIP_POLL          ; poll for completion of write cycle
            RET

;*************************
;*  4. READ STATUS REGISTER
;*************************

RDSR_CMD:   CLR   SCK               ; bring SCK low
            CLR   CS                ; bring CS low
            MOV   A, #RDSR_INST
            LCALL OUTBYT            ; send RDSR instruction
            LCALL INBYT             ; read status register byte
            CLR   SCK               ; bring SCK low
            SETB  CS                ; bring CS high
            RET

;**********************
;*  5. SINGLE BYTE WRITE
;**********************

BYTE_WRITE: MOV   DPTR, #BYTE_ADDR  ; set address of  byte to be written
            CLR   SCK               ; bring SCK low
            CLR   CS                ; bring CS low
            MOV   A, #WRITE_INST
            LCALL OUTBYT            ; send WRITE instruction
            MOV   A, DPH
            LCALL OUTBYT            ; send high order address byte
            MOV   A, DPL
            LCALL OUTBYT            ; send low order address byte
            MOV   A, #BYTE_DATA
            LCALL OUTBYT            ; send data byte
            CLR   SCK               ; bring SCK low
            SETB  CS                ; bring CS high
            LCALL WIP_POLL          ; poll for completion of write cycle
            RET

;********************
;*  6. SINGLE BYTE READ
;********************

BYTE_READ:  MOV   DPTR, #BYTE_ADDR  ; set address of byte to be read
            CLR   SCK               ; bring SCK low
            CLR   CS                ; bring CS low
            MOV   A, #READ_INST
            LCALL OUTBYT            ; send READ instruction
            MOV   A, DPH
            LCALL OUTBYT            ; send high order address byte
            MOV   A, DPL
            LCALL OUTBYT            ; send low order address byte
            LCALL INBYT             ; read data byte
            CLR   SCK               ; bring SCK low
            SETB  CS                ; bring CS high
```

```
            RET

;***************
;*  7. PAGE WRITE
;***************

PAGE_WRITE: MOV   DPTR, #PAGE_ADDR  ; set address of 1st byte to be written
            CLR   SCK               ; bring SCK low
            CLR   CS                ; bring CS low
            MOV   A, #WRITE_INST
            LCALL OUTBYT            ; send WRITE instruction
            MOV   A, DPH
            LCALL OUTBYT            ; send high order address byte
            MOV   A, DPL
            LCALL OUTBYT            ; send low order address byte
            MOV   A, #PAGE_DATA1
            LCALL OUTBYT            ; send 1st data byte
            MOV   A, #PAGE_DATA2
            LCALL OUTBYT            ; send 2nd data byte
            MOV   A, #PAGE_DATA3
            LCALL OUTBYT            ; send 3rd data byte
            CLR   SCK               ; bring SCK low
            SETB  CS                ; bring CS high
            LCALL WIP_POLL          ; poll for completion of write cycle
            RET

;********************
;*  8. SEQUENTIAL READ
;********************

SEQU_READ:  MOV   DPTR, #PAGE_ADDR  ; set address of 1st byte to be read
            CLR   SCK               ; bring SCK low
            CLR   CS                ; bring CS low
            MOV   A, #READ_INST
            LCALL OUTBYT            ; send READ instruction
            MOV   A, DPH
            LCALL OUTBYT            ; send high order address byte
            MOV   A, DPL
            LCALL OUTBYT            ; send low order address byte
            LCALL INBYT             ; read 1st data byte
            LCALL INBYT             ; read 2nd data byte
            LCALL INBYT             ; read 3rd data byte
            CLR   SCK               ; bring SCK low
            SETB  CS                ; bring CS high
            RET

;*****************
;*  WIP POLLING
;*****************

WIP_POLL:   MOV   R1, #MAX_POLL     ; set maximum number of polls
WIP_POLL1:  LCALL RDSR_CMD          ; read status register
            JNB   ACC.0, WIP_POLL2  ; if WIP bit '0' write cycle completed
```

```
            DJNZ  R1, WIP_POLL1     ; if  WIP bit '1' continue polling
WIP_POLL2:  RET

;***************
;* SEND A BYTE
;***************

OUTBYT:     MOV   R0, #08           ; set bit counter to eight
OUTBYT1:    CLR   SCK               ; bring SCK low
            RLC   A                 ; shift byte left through carry
            MOV   SI, C             ; send data bit in carry
            SETB  SCK               ; bring SCK high
            DJNZ  R0, OUTBYT1       ; finish if  last data bit
            CLR   SI                ; place SI in known condition
            RET

;****************
; RECIEVE A BYTE
;****************

INBYT:      MOV   R0, #08           ; set bit counter to eight
INBYT1:     CLR   SCK               ; bring SCK low
            MOV   C, SO             ; receive data bit and store in carry
            RLC   A                 ; shift byte left through carry
            SETB  SCK               ; bring SCK high
            DJNZ  R0, INBYT1        ; finish if last data bit
            RET
```