Using the Xicor X5163/X5323/X5643 CPU Supervisors with the 68HC11 Microcontroller

by Applications Staff, September 1998

Description

The following code demonstrates how Xicor's CPU Supervisors with EEPROM can be interfaced to the 68HC11 microcontroller when connected as shown in Figure 1. The circuit uses the 68HC11's built-in SPI port with the PD4/SCK pin connected to the serial clock (SCK), the PD3/MOSI pin connected to serial data in (SI), the PD2/MISO pin connected to serial data out (SO), and the PD5/PCS0/CS pin connected to chip select ($\overline{CS}$). The SPI interface of the CPU Supervisors operate at 2MHz, so the 68HC11 can operate the SPI port at the maximum speed. This code allows the 68HC11 to read and write data from the EEPROM and select EEPROM Block Lock configurations. It also allows the 68HC11 to change the watchdog timer settings (if available) or turn off the watchdog timer.

More Information

Additional code can be found on the Xicor WWW site at http://www.xicor.com.

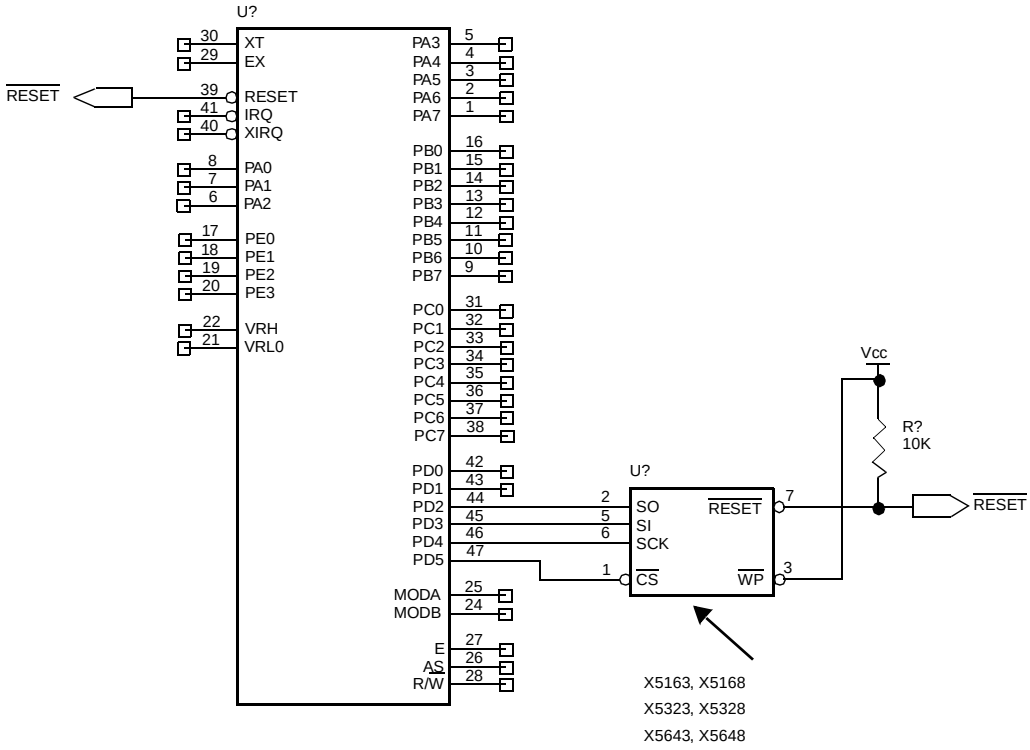

Figure 1. Interfacing the Xicor CPU Supervisors to the 68HC11 Microcontroller using the SPI Port

```
*******************************************************************************
**  DESCRIPTION:
**  The purpose of this program is to show the use of the M68HC11
**  assembly language to program the XICOR CPU Supervisor device. The "WIP"
**  status polling (ACKPOL program) is a unique feature of the SPI memories.
**  The processor interfaces the EEPROM through its Serial Peripheral
**  Interface Port (SPI). The SCK pin is connected to the serial clock (SCK),
**  MOSI to serial data in(SI), MISO to the serial data out (SO), and PCS0/CS
**  to the CS input of the EEPROM. The main section of the code calls up other
**  modules in order to demonstrate the procedure to be followed when
**  reading/writing from/to the device. Routines are provided that write and
**  read data to/from the status register, including BlockLock and Watchdog
**  Timer bits.
**
**  This code supports the following supervisory products:
**
**            X5163, X5165, X5168, X5169*
**            X5323, X5325, X5328, X5329*
**            X5643, X5645, X5648, X5649*
**
**     * These devices have no Watchdog Timer, so write '0' to the Wdx bits.
*******************************************************************************
*******************************************************************************
* INTERNAL RAM LOCATIONS
ADDRL         EQU    $FF           MEMORY ADDRESS LOW BYTE
ADDRH         EQU    ADDRL-1       MEMORY ADDRESS HIGH BYTE
PATTERN       EQU    ADDRH-1       PATTERN REGISTER
STACK         EQU    PATTERN-1     STACK TOP
* CONSTANTS
WREN_CMD      EQU    006           WRITE ENABLE
WRITE_CMD     EQU    002           WRITE DATA TO EEPROM
READ_CMD      EQU    003           READ EEPROM DATA
WRDI_CMD      EQU    006           WRITE DISABLE
RDSR_CMD      EQU    005           READ STATUS REGISTER COMMAND
WRSR_CMD      EQU    001           WRITE STATUS REGISTER COMMAND
DUMMY         EQU    $FF           DUMMY STATUS OF MOSI PIN DURING BYTE READ
SPE_BIT       EQU    $40
SPIF_BIT      EQU    $80           BIT POSITION OF THE SPIF
CE_BIT        EQU    $20           BIT POSITION FOR PCS0/CS
* EQUATES FOR USE WITH INDEX OFFSET = $1000
PORTD         EQU    $08
DDRD          EQU    $09
SPCR          EQU    $28
SPDR          EQU    $2A
SPSR          EQU    $29
BAUD          EQU    $2B
SCDAT         EQU    $2F
SCSR          EQU    $2E
SCCR2         EQU    $2D
STAT_BYTE     EQU    $00           Turn off BlockLock, Set WDT=1.4s
* ASSEMBLER REQUIREMENT- CPU TYPE
              P68H11
              PAGE
*       START OF USER CODE
              ORG    $E000
TEST:
              lds    #STACK        * LOAD STACK POINTER
              ldx    #$1000        * SET REGISTER BASE
* INITIALIZE THE SPI
              ldaa   #$3F
              staa   DDRD,X        * PORT-D PINS ALL SET AS OUTPUTS
              ldaa   #$50
              staa   SPCR,X        * MODE = 0, CLK = 1MHZ
              ldaa   #$00
              staa   PATTERN       * DATA PATTERN TO WRITE
```

```
             ldy    #$100
             sty    ADDRH         * MEMORY ADDRESS TO WRITE
             tab                  * RECALL THE DATA PATTERN
             ldy    ADDRH         * LOAD THE MEMORY ADDRESS
             jsr    wr_byte       * WRITE THE BYTE
             jsr    ACK_POLL      * WAIT TILL DEVICE COMPLETS INTERNAL WRITE
             jsr    rd_byte       * READ THE BYTE
*
             ldab   #STAT_BYTE    * GET THE STATUS BYTE
             jsr    wr_status     * WRITE THE STATUS
             jsr    ACK_POLL      * WAIT TILL DEVICE COMPLETS INTERNAL WRITE

             jmp    *
*******************************************************************************
*** Name: EE_WREN
*** Description: Enable write operation to the EEPROM
*** Function: This program sends out the command to enable the writes and
***          the store operations to the EEPROM
*** Calls: None
*** Input: None
*** Output: None
*** Register Usage: B
*******************************************************************************
EE_wren:
             bclr   PORTD,X,#CE_BIT     * ACTIVATE CE
             ldaa   #WREN_CMD           * WRITE ENABLE COMMAND
             jsr    outbyt              * OUTPUT THE COMMAND
             bset   PORTD,X,#CE_BIT     * DEACTIVATE CE
             rts
*******************************************************************************
*** Name: OUTBYT
*** Description: Sends a byte to the EEPROM
*** Function: This program shifts out a byte, MSB first to the EEPROM.
*** Calls: None
*** Input: A = Byte to be sent
*** Return Value: None
*** Register Usage: None
*******************************************************************************
outbyt:
             staa   SPDR,X
outbyt1:
             brclr  SPSR,X,#SPIF_BIT,outbyt1  *WAIT FOR LAST ONE TO COMPLETE
             rts
*******************************************************************************
*** Name: RD_BYTE
*** Description: Reads content of the EEPROM at a specific location.
*** Function: This program sends out the command to read the content of a memory
***          location specified in the (E) register.
*** Calls: EE_read_cmd,  outbyt
*** Input: Y = Address of the byte
*** Output: B = READ VALUE
*** Register Usage: B
*******************************************************************************
rd_byte:
             bclr   PORTD,X,#CE_BIT     * ACTIVATE CE
             jsr    EE_read_cmd         * ISSUE  READ  COMMAND
             pshy                       * SAVE ADDR
             tsy
             ldaa   0,y                 * RECALL THE MSB OF ADDRESS
             jsr    outbyt              * SEND IT TO EEPROM
             ldaa   1,y                 * RECALL THE LSB OF ADDRESS
             jsr    outbyt              * SEND IT TO EEPROM
             ldaa   #DUMMY              * SHIFT IN THE DATA FROM EEPROM
             jsr    outbyt
             ldaa   SPDR,X              * LOAD RECEIVED DATA FROM SPI
             bset   PORTD,X,#CE_BIT     * DEACTIVATE CE
             puly
```

```
            rts
*******************************************************************************
*** Name: EE_READ_CMD
*** Description: Sends the read command to the EEPROM
*** Function: This program sends a read command to the EEPROM
*** Calls: outbyt
*** Input: E = BYTE ADDRESS
*** Return Value: A = RECEIVED BYTE
*** Register Usage: B, E, IZ
*******************************************************************************
EE_read_cmd:
            ldaa   #READ_CMD           * SEND READ COMMAND TO THE EEPROM
            jmp    outbyt              * SEND THE COMMAND
*******************************************************************************
*** Name: WR_BYTE
*** Description: Writes a byte to the EEPROM at a specific location.
*** Function: This program writes the byte in the (B) register to the EEPROM
***          location specified by the (E) register.
*** Calls: EE_wren, EE_write_cmd, outbyt
*** Input: Y = byte Address, B = Data to write
*** Output: None
*** Register Usage: A,B
*******************************************************************************
wr_byte:
            jsr    EE_wren             * SEND WRITE ENABLE COMMAND
            bclr   PORTD,X,#CE_BIT     * ACTIVATE CE
            jsr    EE_write_cmd        * ISSUE WRITE COMMAND
            pshy                       * SAVE ADDR
            tsy
            ldaa   0,y                 * RECALL THE MSB OF ADDRESS
            jsr    outbyt              * SEND IT TO EEPROM
            ldaa   1,y                 * RECALL THE LSB OF ADDRESS
            jsr    outbyt              * SEND IT TO EEPROM
            tba
            jsr    outbyt              * SEND IT TO EEPROM
            bset   PORTD,X,#CE_BIT     * DEACTIVATE CE
            puly                       * RECALL ADDR
            rts
*******************************************************************************
*** Name: EE_write_cmd
*** Description: Sends the write command to the EEPROM
*** Function: This program creats the write command sequence and transmits
***          it to the EEPROM.
*** Calls: outbyt
*** Input: Y = BYTE   ADDRESS
*** Return Value: None
*** Register Usage: A
*******************************************************************************
EE_write_cmd:
            ldaa   #WRITE_CMD      * SEND WRITE COMMAND TO THE EEPROM
            jmp    outbyt
*******************************************************************************
*** Name: WR_STATUS
*** Description: Writes a byte to the CPU Supervisor status register.
*** Function: This program writes the byte in the (B) register to the status
*** register. Since the command determines the address, no address is sent.
*** The data sent controls the operation of the BlockLock, InCircuit
*** Programmable ROM and Watchdog Timer Functions (if available). The content
*** of the byte sent is:
***
***
***          MSB                              LSB
***          WPEN  X  WD1  WD0 BL1  BL0  WEL  WIP
***
*** Calls: EE_wren, EE_wrsr_cmd, outbyt
*** Input: Y = byte Address, B = Data to write
*** Output: None
```

```
*** Register Usage: A,B
*******************************************************************************
wr_status:
             jsr    EE_wren             * SEND WRITE ENABLE COMMAND
             bclr   PORTD,X,#CE_BIT     * ACTIVATE CE
             jsr    EE_wrsr_cmd         * ISSUE WRITE STATUS COMMAND
             tba                        * GET THE STATUS BYTE
             jsr    outbyt              * SEND IT TO EEPROM
             bset   PORTD,X,#CE_BIT     * DEACTIVATE CE
             rts
*******************************************************************************
*** Name: EE_wrsr_cmd
*** Description: Sends the write command to the EEPROM
*** Function: This program creats the write command sequence and transmits
***          it to the EEPROM.
*** Calls: outbyt
*** Input: Y = BYTE   ADDRESS
*** Return Value: None
*** Register Usage: A
*******************************************************************************
EE_wrsr_cmd:
             ldaa   #WRSR_CMD      * SEND WRITE COMMAND TO THE EEPROM
             jmp    outbyt
*******************************************************************************
*** Name: ACK_POLL
*** Description: Verifies if the EEPROM is ready and accepting commands
*** Function: This program sends the status register read command to the
***             EEPROM and returns to the caller when the WIP bit in the status
***             byte is cleared or the  maximum number of retries is reached.
***             This routine can also be used to read the status of the WDx and
***             BLx bits.
*** Calls: outbyt
*** Input: None
*** Return Value: None
*** Register Usage: A,B
*******************************************************************************
ACK_POLL:
ackpol1:
             bclr   PORTD,X,#CE_BIT     * ACTIVATE CE
             ldaa   #RDSR_CMD           * READ STATUS COMMAND
             jsr    outbyt              * SEND THE COMMAND
             ldaa   #DUMMY              * DUMMY COMMAND
             jsr    outbyt              * SEND THE COMMAND
             bset   PORTD,X,#CE_BIT     * DEACTIVATE CE
             ldaa   SPDR,X              * LOAD RECEIVED DATA FROM SPI
             asra
             bcc    ackpol1
             rts
                    ORG $FFFE
             FDB TEST
                    END
```

littlediode
COMPONENTS FOR ALL

Littlediode supplies new, hard to find or obsolete electronic components and semiconductors all over the world.

With over two million different components listed you are sure to find the part you need.

Feel free to visit us today at our online store:

LittleDiode.com

Looking forward to providing you with the best possible service.