

Interfacing the X24C44/45 NOVRAMs to the Motorola 6805 Microcontroller using the SPI Port

by Applications Staff, March 1991

The following code demonstrates how the Xicor X24C44/45 serial NOVRAMs could be interfaced to the 6805 microcontroller family when connected as shown in Figure 1. The interface uses the SPI port, with the MISO pin connected to DO, MOSI connected to DI, and

SCK connected to SK. CE is generated from another port pin. Additional code can be found on the Xicor web site at <http://www.xicor.com> that will implement interfaces between the 6805 microcontroller family and other Xicor serial devices.

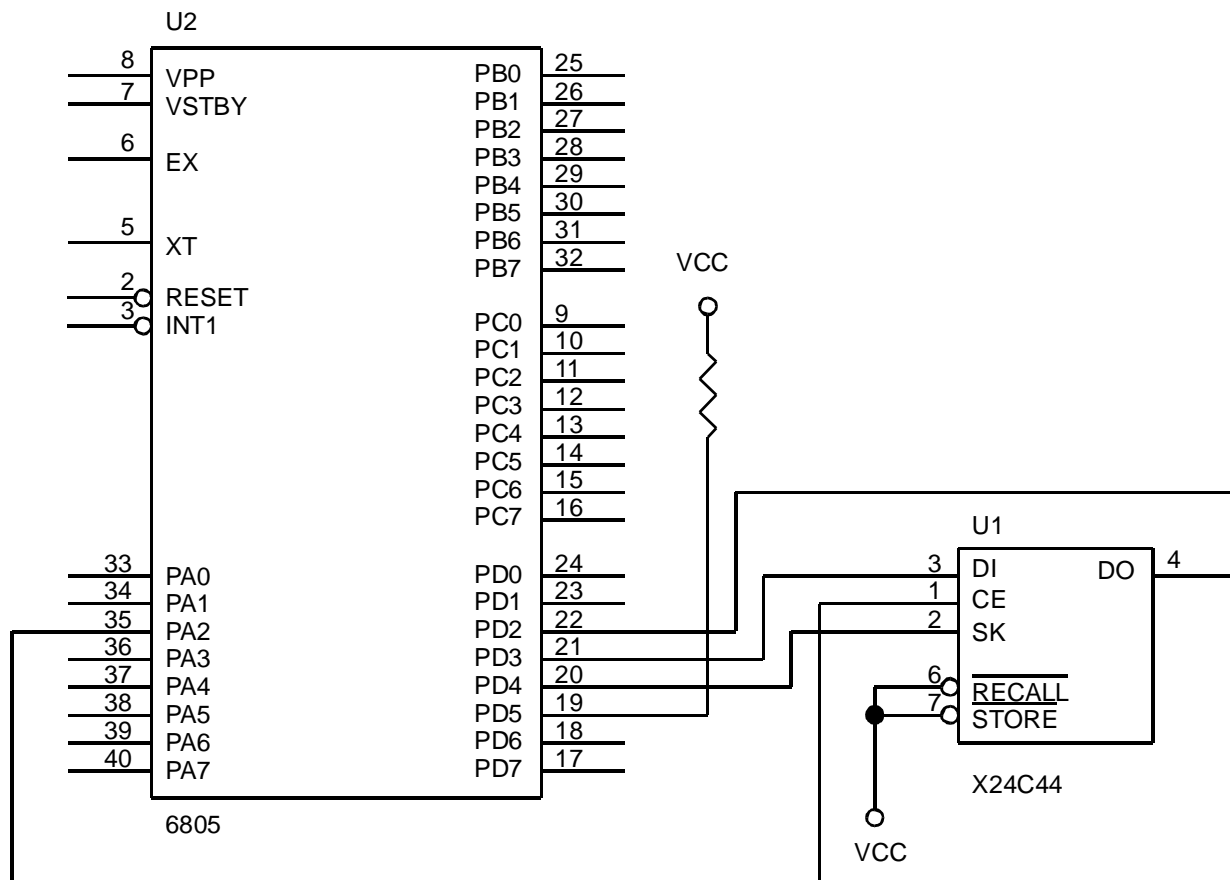


Figure 1. Typical hardware connection for interfacing an X24C44 to 6805 microcontrollers

```

*****
* This code was designed to demonstrate how the X24C44 could be interfaced *
* to the 68HC05 microcontroller. The interface uses the serial SPI port to *
* communicate with the X24C44. The 68HC05 is configured as a master and *
* the X24C44 is the slave. The X24C44 CE signal is generated with a port *
* line. The SPI port provides an interface that greatly minimizes the *
* software required to communicate with the X24C44. *
*
* The code shown demonstrates RCL, WREN, READ, WRITE, and STORE *
* instructions. The remaining instructions (WRDS and for the X24C45, ENAS) *
* can be issued using the routine as other non-data instructions. *
*
* The program issues a sequence of instructions to read the contents of *
* address 5 and stores the same value in address 9. The sequence of *
* instructions is as follows: *
*
* 1) RCL sets the previous recall latch *
* 2) WREN sets the write enable latch *
* 3) READ data from address 5 is read *
* 4) WRITE the data read during step 3 is written to address 9 *
* 5) STO the RAM's contents are transferred to the EEPROM *
*
* Data transfer is performed with the most significant bit first. *
*****

```

```

CEBIT EQU 2 bit number indicating port D CE position
SPIF EQU 7 bit indicating end of transmission
DDRA EQU $04 port A data direction register address
PORTA EQU $00 port A address
WRDS EQU $80 reset write enable latch
STO EQU $81 transfer data from RAM to EEPROM
ENAS EQU $82 enable AUTOSTORE (X24C45)
WRITE EQU $83 RAM write
WREN EQU $84 set write enable latch
RCL EQU $85 transfer data from EEPROM to RAM and does WRDS
READ EQU $86 RAM read
SPCR EQU $0A SPI control register
SPSR EQU $0B SPI status register
SPDR EQU $0C PI data register
ADDR EQU $80 location for X24C44 address to access
INST EQU $81 instruction for part
RWDAT EQU $82 location for X24C44 data transferred

```

```

*****
* Reset vector to beginning of program *
*****

```

```

ORG $FFE reset vector to program entry point
FDB $0100

```

```

*****
* Start of program execution *
*****

```

```

ORG $0100 beginning of executable code
BEGIN: LDA #$04
STA DDRA
BCLR #CEBIT,PORTA bring CE low
LDA #$50 initialize SPI port
STA SPCR
LDA SPSR make sure SPI SPIF bit is reset
LDA #RCL perform a recall to set the
STA INST recall latch
JSR CEHIGH
JSR OUTBYT
JSR CELOW

```

```

LDA    #WREN          perform a write enable to set
STA    INST          the write enable latch
JSR    CEHIGH
JSR    OUTBYT
JSR    CELOW
LDA    #$05          read the contents of address 5
STA    ADDR          the value read will be stored
JSR    RDWRD         in RWDATA
LDA    #$09          write the data just read into
STA    ADDR          address 9
JSR    WRWRD         perform a store operation
LDA    #STO
STA    INST
JSR    CEHIGH
JSR    OUTBYT
JSR    CELOW
BRA    *             loop until reset

```

```

*****
* Write the word specified in RWDAT. The address to be written *
* is specified in ADDR. *
*****

```

```

WRWRD: JSR    CEHIGH      write value in RWDAT into location
LDA    ADDR              justify address
LSLA
LSLA
LSLA
ORA    #WRITE           mask in write instruction
JSR    OUTBYT           send write instruction
LDA    RWDAT
JSR    OUTBYT           send first byte of data
LDA    RWDAT+1
JSR    OUTBYT           send second byte of data
JSR    CELOW
RTS

```

```

*****
* Read the word at the location specified in ADDR. The data *
* read will be placed in RWDAT. *
*****

```

```

RDWRD: JSR    CEHIGH      read the address specified in ADDR
LDA    ADDR              justify address
LSLA
LSLA
LSLA
ORA    #READ            mask in read instruction
JSR    OUTBYT           send read instruction
JSR    OUTBYT           'read' first byte
STA    RWDAT            save first byte sent back from X24C44
JSR    OUTBYT           'read' second byte
STA    RWDAT+1          save second byte sent back from X24C44
JSR    CELOW
RTS

```

```

*****
* Send a byte out to the X24C44 and read what is sent back on the DO *
* pin. Data is shifted out to the X24C44 on the MOSI pin. While the *
* shifting is taking place the level from the X24C44 DO pin is being *
* read by the MISO input. Clocking for the X24C44 is generated by the *
* SPI SCK output. The full duplex method of data transfer means that *
* the same routine that is used to write to the X24C44 can be used *
* for the read operation. The routine waits until the transfer is *
* completed by polling the SPIF bit in the SPI control register. Once *
* the transfer has completed the data sent back from the X24C44 is *

```

```
* read from the SPDR. *  
*****
```

```
OUTBYT:STA    SPDR          send byte out  
WAIT2: BRCLR  #SPIF,SPSR,WAIT2  wait for transfer  
          LDA    SPDR          read byte sent back  
          RTS
```

```
*****  
* bring CE high *  
*****
```

```
CEHIGH:BSET   #CEBIT,PORTA  bring CE high  
          RTS
```

```
*****  
* bring CE low *  
*****
```

```
CELOW: BCLR   #CEBIT,PORTA  bring CE low  
          RTS
```