

Interfacing the X24C44/45 NOVRAMs to the Motorola 6803 Microcontroller

by Applications Staff, July 1992

The following code demonstrates how the Xicor X24C44/45 serial NOVRAMs can be interfaced to the Motorola 6803 microcontroller when connected as shown in Figure 1. The code uses three pins from port

1 to implement the interface. Additional code can be found on the Xicor web site at <http://www.xicor.com> that will implement interfaces between several other Motorola microcontroller families and most Xicor serial devices.

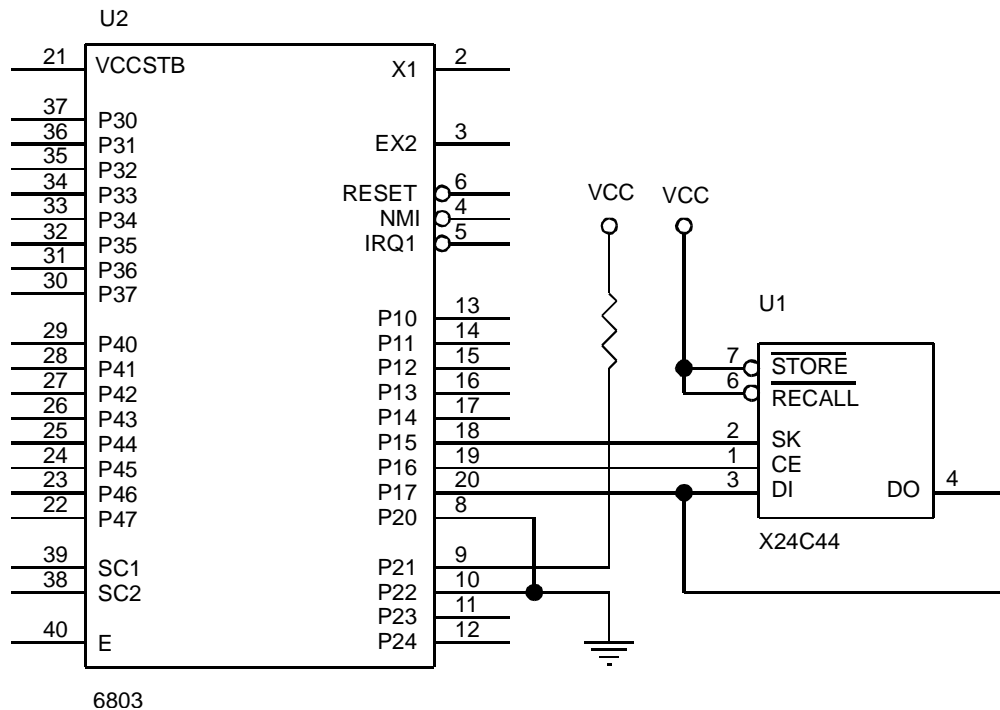


Figure 1. Interfacing an X24C44 to a 6803 microcontroller

```

*****
* THIS CODE WAS DESIGNED TO DEMONSTRATE HOW THE X24C44 COULD BE INTERFACED TO *
* THE 6803 MICROCONTROLLER. THE INTERFACE USES 3 LINES FROM PORT 1 (P17, *
* P16, AND P15) TO COMMUNICATE. THE DI AND DO PINS ON THE X24C44 ARE TIED *
* TOGETHER WHICH ALLOWS 1 LESS PORT LINE TO BE USED. *
* *
* THE CODE SHOWN DEMONSTRATES RCL, WREN, READ, WRITE, AND STORE *
* INSTRUCTIONS. THE REMAINING INSTRUCTIONS (WRDS AND ENAS) CAN BE ISSUED *
* USING THE SAME 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 IS TRANSFERED TO THE EEPROM *
* *
* DATA TRANSFER IS PERFORMED WITH THE MOST SIGNIFICANT BIT FIRST. DURING *
* THE READ AND WRITE INSTRUCTIONS THE DATA SEQUENCE IS INVERTED FROM THAT *
* SHOWN IN THE DATA BOOK (D15 IS SHIFTED FIRST). *
*****
SKHI  EQU  $20          MASK TO GENERATE A 1 ON SK
SKLO  EQU  $DF          MASK TO GENERATE A 0 ON SK
DIHI  EQU  $80          MASK TO GENERATE A 1 ON DI
DILO  EQU  $7F          MASK TO GENERATE A 0 ON DI
CEHI  EQU  $40          MASK TO GENERATE A 1 ON CE
CELO  EQU  $BF          MASK TO GENERATE A 0 ON CE
WRDS  EQU  $80          RESET WRITE ENABLE LATCH
STO   EQU  $81          TRANSFERS FROM RAM TO EEPROM
ENAS  EQU  $82          PLACES PART INTO POWER DOWN MODE
WRITE EQU  $83          RAM WRITE
WREN  EQU  $84          SET WRITE ENABLE LATCH
RCL   EQU  $85          TRANSFERS FROM EEPROM TO RAM, RESETS
*                               WRITE ENABLE LATCH

READ  EQU  $86          RAM READ
DDR1  EQU  $00          DATA DIRECTION REGISTER FOR PORT 1
PORT1 EQU  $02          ADDRESS FOR PORT 1
ADDR  EQU  $80          LOCATION FOR X24C44 ADDRESS TO ACCESS
INST  EQU  $81          INSTRUCTION FOR PART
RWDAT EQU  $82          LOCATION FOR X24C44 DATA TRANSFERED
P1DATA EQU  $84         DATA TO BE SENT TO DUT
DD1DAT EQU  $85         DATA TO BE STORED IN PORT 1 DIRECTION REGISTER

*****
* RESET VECTOR TO BEGINNING OF PROGRAM CODE *
*****

ORG   $FFFE          RESET VECTOR TO PROGRAM ENTRY POINT
FDB   $E000

```

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

```

      ORG      $E000          BEGINNING OF EXECUTABLE CODE

BEGIN: LDS    #$00FF        INITIALIZE STACK POINTER
      LDAA   #$FF          PORT 1 ALL OUTPUTS
      STAA  DDR1          INITIALIZE PORT1 DIRECTION REGISTER
      STAA  DD1DAT        INITIALIZE PORT1 DIRECTION VALUE
      LDAA   #$1F          CE, SK, DI ALL 0S
      STAA  PORT1         INITIALIZE PORT1
      STAA  P1DATA        INITIALIZE PORT1 DATA VALUE
      LDAA   #RCL          PERFORM A RECALL TO SET
      STAA  INST          THE RECALL LATCH
      JSR   CEHIGH
      JSR   OUTBYT
      JSR   CELOW
      LDAA   #WREN          PERFORM A WRITE ENABLE TO SET
      STAA  INST          THE WRITE ENABLE LATCH
      JSR   CEHIGH
      JSR   OUTBYT
      JSR   CELOW
      LDAA   #$05          READ THE CONTENTS OF ADDRESS 5
      STAA  ADDR          THE VALUE READ WILL BE IN STORED
      JSR   RDWRD          IN RWDATA
      LDAA   #$09          WRITE THE DATA JUST READ INTO
      STAA  ADDR          ADDRESS 9
      JSR   WRWRD
      LDAA   #STO          PERFORM A STORE OPERATION
      STAA  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 RWDATA INTO LOCATION
      LDAA  ADDR          SPECIFIED IN ADDR
      LSLA                JUSTIFY ADDRESS IN INSTRUCTION
      LSLA
      LSLA
      ORAA  #WRITE        MASK IN WRITE INSTRUCTION
      STAA  INST
      JSR   OUTBYT        SEND WRITE INSTRUCTION TO DUT
      LDAA  RWDAT
      STAA  INST
      JSR   OUTBYT        SEND IN UPPER BYTE OF DATA
      LDAA  RWDAT+1
      STAA  INST

```

```

JSR   OUTBYT      SEND IN LOWER 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
        LDAA  ADDR
        LSLA                JUSTIFY ADDRESS TO READ
        LSLA
        LSLA
        ORAA  #READ      MASK IN READ INSTRUCTION
        STAA  INST
        JSR   SEND7      SEND IN 7 BITS OF READ INSTRUCTION
        LDAA  DD1DAT     MAKE DATA LINE AN INPUT
        ANDA  #DILO
        STAA  DDR1
        STAA  DD1DAT
        JSR   CLOCK      SEND EIGHTH CLOCK PULSE FOR READ INSTRUCTION
        LDX  #$0010     PREPARE TO SHIFT IN 16 BITS
BITX:  CLC                ASSUME BIT IS GOING TO BE A ZERO (CLEAR CARRY)
        LDAA  PORT1     READ BIT VALUE
        ANDA  #DIHI     MASK BIT OUT OF BYTE READ
        BEQ  NO1        LEAVE CARRY FLAG ALONE IF BIT IS A 0
        SEC                SET CARRY IF BIT IS A 1
NO1:   ROL  RWDAT+1     ROLL CARRY FLAG INTO DATA WORD
        ROL  RWDAT
        JSR   CLOCK      SEND A CLOCK PULSE
        DEX                LOOP UNTIL
        BNE  BITX       16 BITS ARE READ
        LDAA  DD1DAT     MAKE DATA LINE AN OUTPUT
        ORAA  #DIHI
        STAA  DDR1
        STAA  DD1DAT
        JSR   CELOW
RTS

```

```

*****
* SEND DATA OUT TO THE PART. THE DATA TO BE SENT IS *
* LOCATED IN INST.                                     *
*****

```

```

SEND7: LDX  #$0007      SHIFT OUT 7 BITS FOR READ INSTRUCTION
        BRA  LOOPO
OUTBYT: LDX  #$0008      PREPARE TO SHIFT OUT 8 BITS
LOOPO: LDAB  P1DATA
        ANDB #DILO
        ROL  INST
        BCC  IS0        JUMP IF DATA SHOULD BE 0
        ORAB #DIHI     MAKE DATA A 1
IS0:   STAB  PORT1     PUT DATA ON SDA
        STAB  P1DATA

```

```
JSR    CLOCK          SEND CLOCK SIGNAL
DEX
BNE    LOOPO          LOOP UNTIL ALL 8 BITS HAVE BEEN SENT
RTS
```

* BRING CE HIGH *

```
CEHIGH:LDAA  P1DATA          BRING CE HIGH
        ORAA  CEHI
        STAA  PORT1
        STAA  P1DATA
        RTS
```

* BRING CE LOW *

```
CELOW: LDAA  P1DATA          BRING CE LOW
        ANDA  CELO
        STAA  PORT1
        STAA  P1DATA
        RTS
```

* ISSUE A CLOCK PULSE. *

```
CLOCK: LDAA  P1DATA          PROVIDE A CLOCK PULSE ON SK
        ORAA  #SKHI
        STAA  PORT1          BRING SK HIGH
        ANDA  #SKLO
        STAA  PORT1          BRING SK LOW
        STAA  P1DATA
        RTS
```



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.