Interfacing the X24C44/45 NOVRAMs to the Motorola 68HC11 Microcontroller

by Applications Staff, July 1992

The following code demonstrates how the Xicor X24C44/45 serial NOVRAMs can be interfaced to the Motorola 68HC11 microcontroller family when connected as shown in Figure 1. The code uses three pins from port D 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.

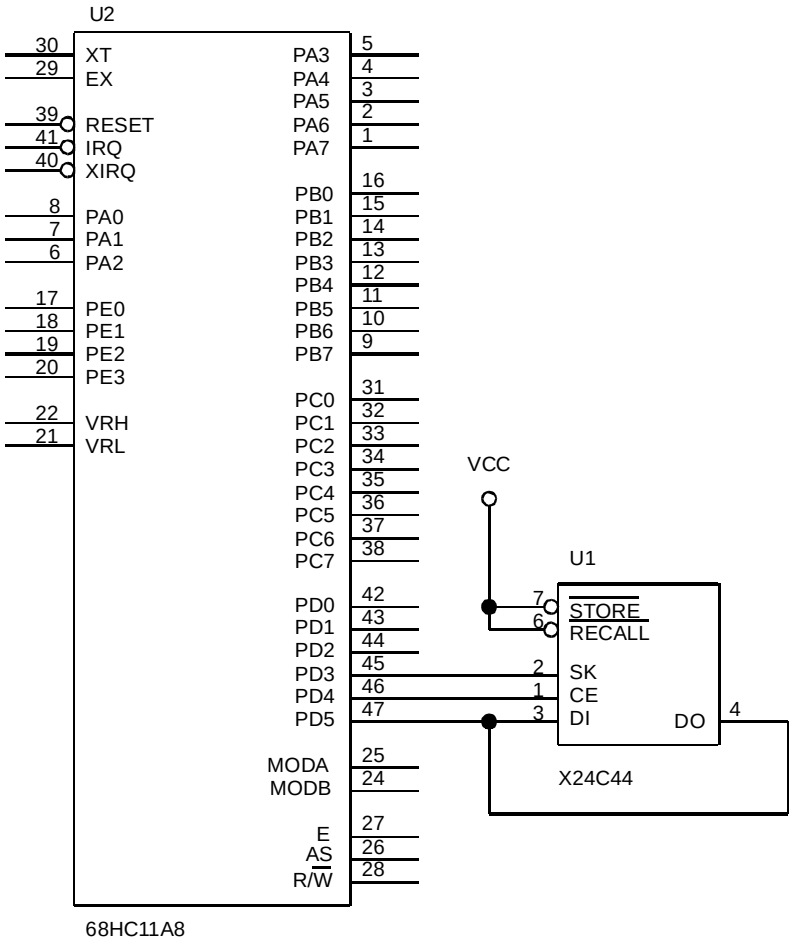

Figure 1. Interfacing an X24C44 to a 68HC11 microcontroller using Port D

```
*****************************************************************************
* THIS CODE WAS DESIGNED TO DEMONSTRATE HOW THE X24C44 COULD BE INTERFACED TO *
* THE 68HC11 MICROCONTROLLER. THE INTERFACE USES 3 LINES FROM PORT 1 (PD3,    *
* PD4, AND PD5) 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).                              *
*****************************************************************************

SKBIT       EQU   $08          MASK INDICATING PORTD SK POSITION
CEBIT       EQU   $10          MASK INDICATING PORTD CE POSITION
DIOBIT      EQU   $20          MASK INDICATING PORTD DATA POSITION
DOUT        EQU   $38          MASK TO MAKE DI/O AN OUTPUT
DIN         EQU   $18          MASK TO MAKE DI/O AN INPUT
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
DDRD        EQU   $09          DATA DIRECTION REGISTER FOR PORT D
PORTD       EQU   $08          ADDRESS FOR PORT D
ADDR        EQU   $80          LOCATION FOR X24C44 ADDRESS TO ACCESS
INST        EQU   $81          INSTRUCTION FOR PART
RWDAT       EQU   $82          LOCATION FOR X24C44 DATA TRANSFERED
COUNT       EQU   $84          COUNTER VARIABLE

*****************************************
* 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
            LDX         #$1000      INITIALIZE PAGE OFFSET LOCATION
            LDAA        #DOUT
            STAA        DDRD,X      MAKE CE, SK, DI/O OUTPUTS
            LDAA        #$00
            STAA        PORTD,X     INITIALIZE CE, SK, DI/O TO ZEROS
            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  #DIN          MAKE DATA LINE AN INPUT
            STAA  DDRD,X
            JSR   CLOCK         SEND EIGHTH CLOCK PULSE FOR READ INSTRUCTION
            LDAA  #$10          PREPARE TO SHIFT IN 16 BITS
            STAA  COUNT
BITX:       CLC               ASSUME BIT IS GOING TO BE A ZERO (CLEAR CARRY)
            LDAA  PORTD,X       READ BIT VALUE
            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
            DEC   COUNT         LOOP UNTIL
            BNE   BITX          16 BITS ARE READ
            LDAA  #DOUT         MAKE DATA LINE AN OUTPUT
            STAA  DDRD,X
            JSR   CELOW         BRING CE LOW
    RTS

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

SEND7:      LDAA  #$07              SHIFT OUT 7 BITS FOR READ INSTRUCTION
            STAA  COUNT
            BRA   LOOPO
OUTBYT:     LDAA  #$08              PREPARE TO SHIFT OUT 8 BITS
            STAA  COUNT
LOOPO:      ROL   INST
            BCC   IS0               JUMP IF DATA SHOULD BE 0
            BSET  PORTD,X DIOBIT    SEND 1 TO DI/O
            BRA   IS1
IS0:        BCLR  PORTD,X DIOBIT    SEND 0 TO DI/O
IS1:        JSR   CLOCK             SEND CLOCK SIGNAL
```

```
            DEC    COUNT
            BNE    LOOPO              LOOP UNTIL ALL 8 BITS HAVE BEEN SENT
            RTS

*****************
* BRING CE HIGH *
*****************

CEHIGH:     BSET   PORTD,X #CEBIT     BRING CE HIGH
            RTS

****************
* BRING CE LOW *
****************

CELOW:      BCLR   PORTD,X DIOBIT     BRING DATA LINE LOW
            BCLR   PORTD,X #CEBIT     BRING CE LOW
            RTS

**********************
* ISSUE A CLOCK PULSE *
****************

CLOCK:      BSET   PORTD,X SKBIT      BRING SK HIGH
            BCLR   PORTD,X SKBIT      BRING SK LOW
            RTS
```