Interfacing the X24C01A/02/04/08/16 to the Motorola 68HC11 Microcontroller

by Applications Staff, August 1992

The following code demonstrates how the Xicor X24C04 family of serial E^2PROMs could be interfaced to the Motorola 68HC11 microcontroller family when connected as shown in Fig. 1. The code uses two pins from port D to implement the interface. Additional code can be found on the Xicor BBS (or through the Xicor FaxBack system) that will implement interfaces between several other Motorola microcontroller families and most Xicor serial devices. The Xicor BBS can be reached toll free at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 19.2K baud rate modem (no parity, 8 bit words, 1 stop bit, and no local echo). These listings can be found in the MOTOROLA SIG (Special Interest Group). Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627.

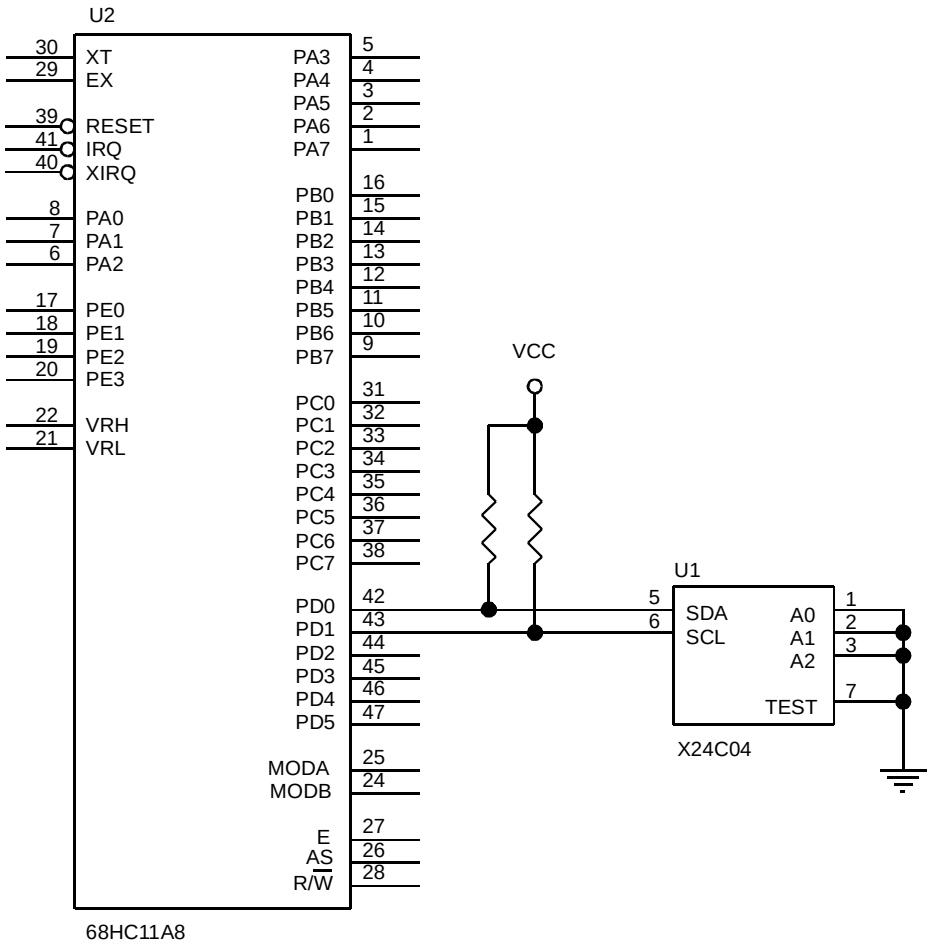

Figure 1 - Interfacing an X24C04 to a 68HC11 microcontroller using Port D

```
*******************************************************************************
*  This code was designed to demonstrate how the X24C04 family of parts could  *
*  be interfaced to the 68HC11 microcontroller. The interface uses 2 lines     *
*  from Port D (PD0 and PD1) to communicate.  Other I2C compatible             *
*  parts can be added to the bus as long as they do not have $A as their       *
*  device identifier.  The routines RDBYT and WRBYT are tailored specifically  *
*  to the X24C04 family.  The routines START, STOP, ACK, NACK, OUTBYT, and     *
*  INBYT can be considered generic I2C routines.                               *
*                                                                             *
*  The code shown demonstrates a 'random read' and 'byte write'.  The other    *
*  modes of operation can be created by expanding upon these routines.         *
*  Acknowledge polling is used to determine when the write cycle finishes.     *
*                                                                             *
*  This code will work with all Xicor I2C compatible EEPROMs reguardless of    *
*  their size.  As long as the address pins are configured correctly this      *
*  code will not know the difference between a bus with a single X24C16 and a  *
*  bus with eight X24C02 devices.                                             *
*                                                                             *
*  The mainline of this program reads the data located at address 002DH and    *
*  then writes that data back to address 0041H.                               *
*  REVISED: MAY 1996
*******************************************************************************

SCLBIT      EQU   $02         MASK INDICATING PORTD SCL POSITION
SDABIT      EQU   $01         MASK INDICATING PORTD SDA POSITION
SDAOUT      EQU   $03         MAKES SDA AN OUTPUT IF STORED IN DDRD
SDAIN       EQU   $02         MAKES SDA AN INPUT IF STORED IN DDRD
DMASK       EQU   $80         USED TO MASK BIT TO SEND TO DUT
PORTD       EQU   $08         PORT D OFFSET IN 'PAGE' $1000
DDRD        EQU   $09         PORT D DIRECTION REGISTER OFFSET
ADDR        EQU   $80         LOCATION FOR X24C04 ADDRESS TO ACCESS
DATA        EQU   $82         LOCATION FOR X24C04 DATA TRANSFERED
COUNT       EQU   $83         COUNTER LOCATION FOR LOOPING
PDDATA      EQU   $84         TEMP REGISTER FOR DATA STORAGE
COUNT2      EQU   $85         COUNTER LOCATION FOR ACK POLLING

****************************
* RESET VECTOR ENTRY POINT *
****************************

            ORG   $FFFE       RESET VECTOR ADDRESS TO PROGRAM ENTRY
            FDB   $E000       JUMP TO BEGINNING OF EXECUTABLE CODE

***********************
* PROGRAM ENTRY POINT *
***********************

            ORG   $E000             BEGINNING OF EXECUTABLE CODE

BEGIN:      LDS   #$00FF            INITIALIZE STACK POINTER
            LDX   #$1000            INITIALIZE PAGE OFFSET LOCATION
            LDAA  #$FF              MAKE PORTD ALL ONES
            STAA  PORTD,X
```

```
            LDAA  #$03        MAKE SDA AND SCL OUTPUTS
            STAA  DDRD,X
            LDD   #$002D
            STD   ADDR
            JSR   RDBYT       READ DATA FROM ADDRESS 002DH
            LDD   #$0041
            STD   ADDR
            JSR   WRBYT       WRITE DATA BACK TO ADDRESS 0041H
            JSR   ACKPOL      PERFORM ACK POLLING
            BRA   *           LOOP UNTIL RESET


*********************************************************************
* READ A BYTE "RANDOM READ SEQUENCE".  THE ADDRESS TO READ IS STORED *
*  IN ADDR.  THE DATA FROM THE DUT IS STORED IN DATA.                *
*********************************************************************

RDBYT:      JSR   START       READ A BYTE FROM THE ADDRESS INDICATED
            LDAA  ADDR        IN 'ADDR'
            ASLA
            ORAA  #$A0        BUILD SLAVE ADDRESS
            PSHA
            STAA  DATA
            JSR   OUTBYT      SEND SLAVE ADDRESS
            JSR   NACK        SET SDA HIGH TO RECEIVE ACKNOWLEDGE
            LDAA  ADDR+1
            STAA  DATA
            JSR   OUTBYT      SEND WORD ADDRESS
            JSR   NACK        SET SDA HIGH TO RECEIVE ACKNOWLEDGE
            JSR   START       SEND START COMMAND
            PULA
            ORAA  #$01
            STAA  DATA
            JSR   OUTBYT      SEND SLAVE ADDRESS
            JSR   NACK        SET SDA HIGH TO RECEIVE ACKNOWLEDGE
            JSR   INBYT       READ DATA FROM X24C04
            JSR   ACK         SEND ACKNOWLEDGE
            JSR   STOP        SEND STOP COMMAND
            RTS

*********************************************************************
* WRITE A BYTE "BYTE WRITE SEQUENCE".  THE ADDRESS TO WRITE IS STORED *
*  IN ADDR.  THE DATA TO WRITE IS STORED IN DATA.                     *
*********************************************************************

WRBYT:      LDAA  DATA        WRITE TO BYTE POINTED TO BY ADDR THE
            PSHA              VALUE IN LOCATION 'DATA'
            JSR   START       SEND START COMMAND
            LDAA  ADDR
            ASLA
            ORAA  #$A0
            STAA  DATA
            JSR   OUTBYT      SEND SLAVE ADDRESS
```

```
            JSR   NACK        SET SDA HIGH TO RECEIVE ACKNOWLEDGE
            LDAA  ADDR+1
            STAA  DATA
            JSR   OUTBYT      SEND WORD ADDRESS
            JSR   NACK        SET SDA HIGH TO RECEIVE ACKNOWLEDGE
            PULA
            STAA  DATA
            JSR   OUTBYT      SEND WRITE DATA
            JSR   NACK        SET SDA HIGH TO RECEIVE ACKNOWLEDGE
            JSR   STOP        SEND STOP
            RTS

*****************************************************************
* READ 8 BITS FROM THE DUT.  THE RESULTS ARE RETURNED IN DATA.  *
*****************************************************************

INBYT:      LDAA  #SDAIN      MAKE SDA AN INPUT
            STAA  DDRD,X
            JSR   CLOCK       GET ACK
            LDAA  #$08        PREPARE TO SHIFT IN 8 BITS
            STAA  COUNT
            LDAB  #$00
LOOPI:      JSR   CLOCK       CLOCK DATA
            LSRA
            ROLB
            DEC   COUNT
            BNE   LOOPI       LOOP UNTIL 8 BITS ARE READ
            STAB  DATA        STORE VALUE READ INTO DATA
            LDAA  #SDAOUT           MAKE SDA AN OUTPUT
            STAA  DDRD,X
            RTS

*********************************************************************
* WRITE 8 BITS TO THE DUT.  THE DATA TO SEND IS IN DATA.  IF THE LAST *
* BIT TO SEND IS A ONE THE SDA LINE IS MADE AN INPUT BEFORE THE EIGHTH*
* CLOCK PULSE TO AVOID BUS CONTENTION WHEN THE DUT ACKNOWLEDGES.  THE *
* ROUTINE FINISHES WITH SDA IN AN INPUT STATE.                        *
*********************************************************************

OUTBYT:     LDAA  #$08                  PREPARE TO SHIFT OUT 8 BITS
            STAA  COUNT
            BCLR  PORTD,X #SDABIT       MAKE SDA A 0
            LDAA  DATA
LOOPO:      LDAB  PDDATA
            ANDB  #$01
            ANDA  #DMASK                IS THE DATA TO BE SHIFTED A 1 OR A 0
            BEQ   IS0                   JUMP IF DATA SHOULD BE 0
            BSET  PORTD,X #SDABIT       MAKE SDA A 1
            LDAA  COUNT                 CHECK TO SEE IF LAST BIT TO SEND
            CMPA  #$01
            BNE   IS1
            LDAA  #SDAIN                MAKE SDA AN INPUT IF A 1 IS THE LAST BIT
            STAA  DDRD,X
```

```
            BRA   IS1
IS0:        BCLR  PORTD,X #SDABIT         MAKE SDA A 0
IS1:        JSR   CLOCK                   SEND CLOCK SIGNAL
            LDAA  DATA
            ASLA
            STAA  DATA
            DEC   COUNT
            BNE   LOOPO                   LOOP UNTIL ALL 8 BITS HAVE BEEN SENT
            LDAA  #SDAIN                  MAKE SDA AN INPUT
            STAA  DDRD,X
            RTS

*************************************************************************
* PERFORM ACKNOWLEDGE POLLING TO DETERMINE WHEN THE WRITE CYCLE        *
* COMPLETES.  UPON RETURN FROM THIS ROUTINE THE A REGISTER INDICATES   *
* WHETHER THE DUT EVER ACKNOWLEDGED THE WRITE.  A=0 PART ACKNOWLEDGED, *
* A=1 NO ACKNOWLEDGE RECEIVED.                                         *
*************************************************************************

ACKPOL:     LDAA  #$07F             MAX NUMBER OF TIMES TO CHECK THE PART
            STAA  COUNT2
AKLOOP:     DEC   COUNT2            RETURN IF THE PART
            BEQ   OUTACK            NEVER ISSUES AN ACKNOWLEDGE
            JSR   START             SEND START COMMAND
            LDAA  #$0A0
            STAA  DATA
            JSR   OUTBYT            SEND SLAVE ADDRESS
            JSR   NACK              SEE IF PART ACKNOWLEDGES
            CMPA  #$00
            BNE   AKLOOP            LOOP IF NO ACKNOWLEDGE
OUTACK:     PSHA
            JSR   STOP              SEND STOP
            PULA
            RTS

************************
* ISSUE A STOP COMMAND *
************************

STOP:       BCLR  PORTD,X #SDABIT         MAKE SURE SDA IS LOW
            BSET  PORTD,X #SCLBIT         BRING SCL HIGH
            NOP                           PROVIDE SET-UP TIME
            NOP
            NOP
            NOP
            BSET  PORTD,X #SDABIT         BRING SDA HIGH
            RTS

************************
* ISSUE A START COMMAND *
************************

START:      BSET  PORTD,X #SDABIT         MAKE SURE THAT SDA IS HIGH
```

```
            BSET  PORTD,X #SCLBIT         MAKE SURE THAT SCL IS HIGH
            BCLR  PORTD,X #SDABIT         FORCE SDA LOW
            NOP                           PROVIDE SET-UP TIME
            NOP
            NOP
            NOP
            BCLR  PORTD,X #SCLBIT   FORCE SCL LOW
            RTS

*********************************************************
* ISSUE AN ACKNOWLEDGE.                          *
*********************************************************

ACK:        LDAA  #SDAOUT           MAKE SDA AN OUTPUT
            STAA  DDRD,X
            BCLR  PORTD,X #SDABIT   PERFORM AN 'ACKNOWLEDGE' WITH SDA LOW
            JSR   CLOCK             GENERATE A CLOCK PULSE
            RTS

*****************************************************************
* SDA IS SET HIGH IN THE OUTBYTE ROUTINE. THE ACK ROUTINE     *
* DOES NOT CHECK TO SEE IF THE DUT ACTUALLY ISSUES AN ACKNOWLEDGE.*
*****************************************************************

NACK:       JSR   CLOCK             GENERATE A CLOCK PULSE
            PSHA
            LDAA  #SDAOUT           MAKE SDA AN OUTPUT
            STAA  DDRD,X
            PULA
            RTS

*****************************************************************
* ISSUE A CLOCK PULSE.  WHILE THE CLOCK IS HIGH THE VALUE ON THE *
*  SDA LINE IS PLACED IN THE CARRY FLAG.  WHEN A READ IS TAKING  *
*  PLACE THE CARRY FLAG WILL INDICATE THE VALUE FROM THE DUT.    *
*****************************************************************

CLOCK:      BSET  PORTD,X #SCLBIT         PROVIDE A CLOCK ON SCL, START HIGH
            LDAA  PORTD,X                 READ SDA WHILE SCL IS HIGH
            BCLR  PORTD,X #SCLBIT
            ANDA  #$01                    SDA VALUE IS IN LOWER BIT OF A REG
            RTS
```