Interfacing the X24C01 to the Motorola 68HC11 Microcontroller

by Applications Staff, July 1992

The following code demonstrates how the Xicor X24C01 serial E^2PROM can be interfaced to the Motorola 68HC11 microcontroller family when connected as shown in Figure 1. The code uses two 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

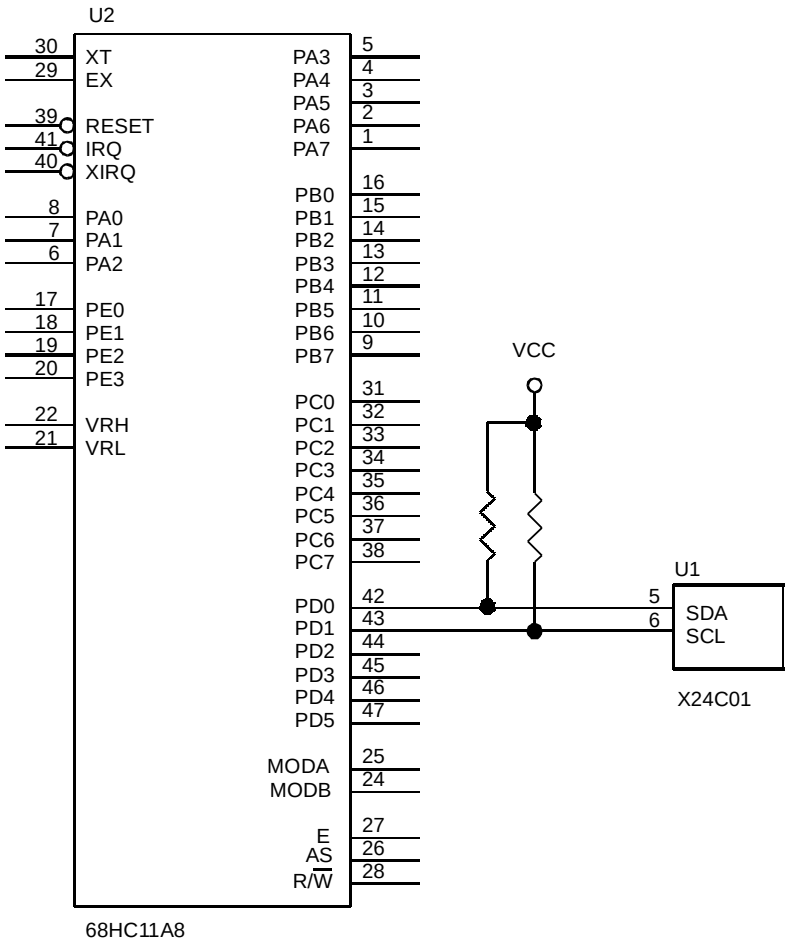

Figure 1. Interfacing an X24C01 to a 68HC11 microcontroller using Port D

devices.

```
*****************************************************************************
*  THIS CODE WAS DESIGNED TO DEMONSTRATE HOW THE XICOR X24C01 COULD          *
*  BE INTERFACED TO THE 68HC11 MICROCONTROLLER.  THE INTERFACE USES 2 LINES  *
*  FROM PORT D (PD0 AND PD1) TO COMMUNICATE.                                 *
*                                                                            *
*  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.   *
*                                                                            *
*  THE MAINLINE OF THIS PROGRAM READS THE DATA LOCATED AT ADDRESS 002DH AND  *
*  THEN WRITES THAT DATA BACK TO ADDRESS 0041H. THIS PROGRAM HAS BEEN TESTED *
*  USING THE X24C01.                                                     *
*  REVISED: JANUARY 1997                                                     *
*****************************************************************************

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 X24C01 ADDRESS TO ACCESS
DATA         EQU    $82          LOCATION FOR X24C01 DATA TRANSFERED
COUNT        EQU    $83          COUNTER LOCATION FOR LOOPING
PDDATA       EQU    $84          TEMP REGISTER FOR DATA STORAGE
COUNT2       EQU    $85          COUNTER 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
             LDAA   #$2D
             STAA   ADDR
             JSR    RDBYT        READ DATA FROM ADDRESS 002DH
             LDAA   #$41
```

```
            STAA   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   #$01         BUILD WORD ADDRESS
            STAA   DATA
            JSR    OUTBYT       SEND WORD ADDRESS
            JSR    NACK         SET SDA HIGH TO RECEIVE ACKNOWLEDGE
            JSR    INBYT        READ DATA FROM X24C01
            JSR    NACK         CLOCK WITHOUT 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
            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 BEFORE READ
            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 8TH   *
* 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   #$080         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   #$00
```

```
            STAA   DATA
            JSR    OUTBYT        SEND SLAVE ADDRESS
            JSR    NACK          SEE IF PART ACKNOWLEDGES
            CMPA   #$00
            BNE    AKLOOP        LOOP IF NO ACKNOWLEDGE
OUTACK:     PSHA
            JSR    START         SEND START
            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 OUTBYT 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
```