AN845

Communicating With The MCP3221 Using PICmicro® Microcontrollers

Author: *Craig L. King*
Microchip Technology Inc.

OVERVIEW

The MCP3221 12-bit A/D Converter (ADC) communicates using a standard 2-wire I²C™ compatible interface. This application note will cover communications between this device and a PICmicro microcontroller. Hardware and software implementations of I²C will be covered. The code supplied with this application note is written as relocatable assembly code.

COMMUNICATION

Communication with the MCP3221 ADC is shown in Figure 1. Seven bit addressing is used with this device. The read/write bit (R/$\overline{W}$) in the address byte should always be logic '1' when executing a conversion. If one wishes to poll the MCP3221 to test for its presence, the R/$\overline{W}$ bit can be set to a logic '0' to accomplish this task. In this scenario, an acknowledge (ACK) will be sent back from the device without initiating a conversion.

Two data bytes follow the address byte. The first data byte will contain four zeros followed by the upper nibble of the 12-bit word. A lower data byte, containing the 8 LSBs, will follow.

Subsequent conversions can be initiated without addressing the device more than once (this is illustrated in Figure 2). The lower data byte will be followed by the upper data byte of the subsequent conversion.

In both situations, the internal conversion is initiated by the falling edge of SCL, either the LSB or the R/$\overline{W}$ bit. Maintaining a SCL frequency no greater than 400 kHz allows the internal conversion to complete prior to the data being clocked out of the device.

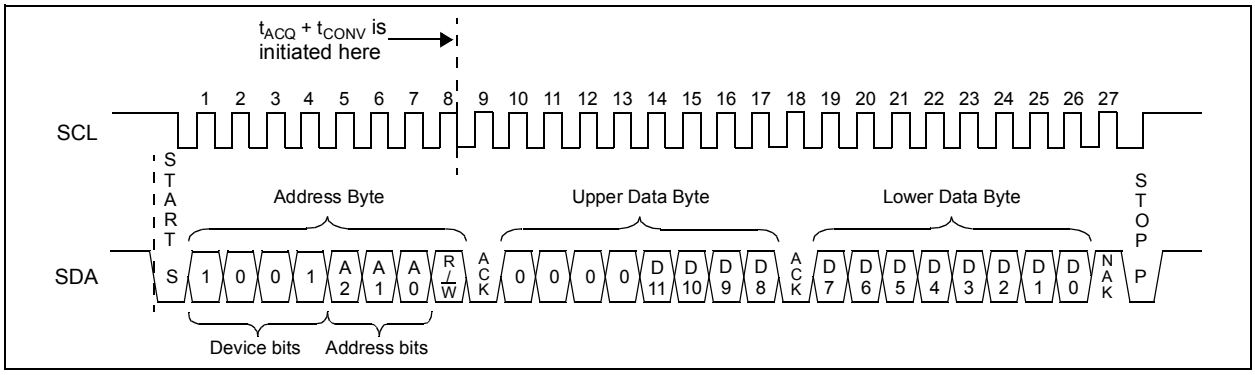


FIGURE 1: *Executing a Conversion*

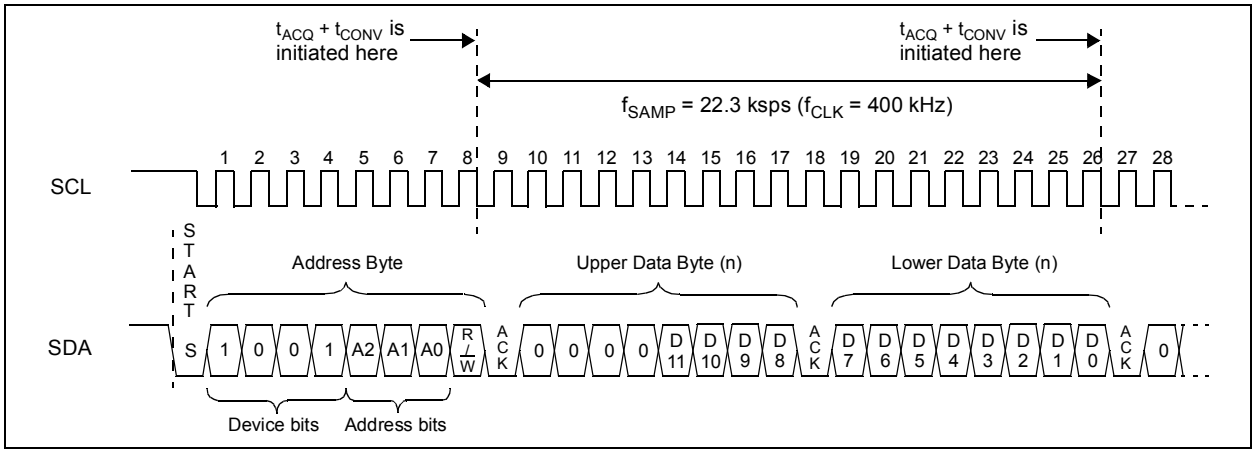


FIGURE 2: *Continuous Conversion*

HARDWARE MSSP IMPLEMENTATION

Appendix A contains relocatable assembly code using the hardware I^2C implementation. This code can be used with PICmicro microcontrollers containing a Master Synchronous Serial Port (MSSP) module.

The MSSP module is first initialized for I^2C communications. The init_j2c subroutine loads the appropriate MSSP registers. Two other routines follow, one for MCP3221 single conversion and one for MCP3221 continuous conversion. Both routines use the following I^2C MSSP subroutines:

- WrtStop - Initiates stop bit on I^2C bus
- WrtStart - Initiates start bit on I^2C bus
- SendWrtAddress - Write Byte to I^2C bus
- StartReadDataHigh - Read Byte from I^2C bus and save in high byte register
- StartReadDataLow - Read Byte from I^2C bus and save in low byte register
- Check_idle - Wait for MSSP idle state

SOFTWARE MSSP IMPLEMENTATION

Appendix B provides relocatable assembly code using software I^2C implementation ("bit banging"). The pins that are used to generate the clock and data signals are also used in the hardware I^2C example. Port initialization occurs, initially setting the SCL and SDA port pins to outputs.

Subroutines are included that generate the Start, Stop, Read, Write and Acknowledge commands.

After each conversion, the file registers ASAMH and ASAML contain the high and low byte of the 12-bit conversion data, respectively.

SCHEMATIC

The code for this application note was developed using the PIC16F876 on MXDEV® analog evaluation driver board, along with a prototype board containing the MCP3221 device. A DC signal source was used to test the device performance and 12-bit accuracy was achieved using proper layout techniques. An equivalent circuit used in this application note is shown in Appendix C. A full schematic of the MXDEV driver board can be found in the MXDEV Driver Board User's Manual (DS51221).

CONCLUSION

The example code supplied in this application note shows how to interface the MCP3221 device with any PICmicro microcontroller product.

REFERENCES

AN735, "Using the PICmicro® MSSP Module for Master I^2C™ Communications", Microchip Technology Inc., DS00735.

AN567, "Interfacing 24LCXXB Serial EEPROMs to the PIC16C54", Microchip Technology Inc., DS00567.

MCP3221 Device Data Sheet, Microchip Technology Inc., DS21732.

APPENDIX A: ASSEMBLY CODE USING HARDWARE I²C IMPLEMENTATION

```
;*****************************************************************
;                                                                *
;   MSSP I2C  Communication with the MCP3221 ADC using a PICmicro®    *
;                                                                *
;                                                                *
;*****************************************************************
;                                                                *
;   Filename:       hardware.asm                                 *
;   Date:           06/26/2002                                   *
;   Revision:       1.00                                         *
;                                                                *
;   Tools:          MPLAB   5.55.00                              *
;                   MPLINK  2.50.00                              *
;                   MPASM   2.50.00                              *
;                                                                *
;   Author:         Craig L. King                                *
;                                                                *
;   Company:        Microchip Technology Incorporated            *
;                                                                *
;*****************************************************************
;                                                                *
;   System files required:                                       *
;                                                                *
;                   hardware.asm (this file)                     *
;                                                                *
;                   p16f876.inc                                  *
;                   16f876.lkr                                   *
;                                                                *
;*****************************************************************
;                                                                *
;   Notes:                                                       *
;                                                                *
;   Device Fosc -> 20.00MHz                                      *
;       WDT -> on                                                *
;       Brownout -> on                                           *
;       Powerup timer -> on                                      *
;       Code Protect -> off                                      *
;                                                                *
;                                                                *
;                                                                *
;                                                                *
;*****************************************************************/


        #include <p16f876.inc>          ; processor specific definitions
```

```
#define  FOSC        D'20000000'            ; define FOSC to PICmicro
#define  I2CClock    D'400000'              ; define I2C bit rate
#define  ClockValue  (((FOSC/I2CClock)/4) -1);
#define  rw_done     7                      ; flag bit
#define  ack_error   0                      ; flag bit
#define  scl  portc,3
#define  sda  portc,4



;Local Variables
;---------------

        udata


temp_address    res     1                  ; I2C Address
eflag_event     res     1
sflag_event     res     1
count           res     1
asamh           res     1
asaml           res     1



;-------
;-Code--
;-------

prog1     code




;*****************************************************************
;  ****************** INITIALIZE MSSP MODULE  *******************
;*****************************************************************



init_i2c
            banksel SSPADD                ; select SFR bank
            movlw   ClockValue            ; read selected baud rate
            movwf   SSPADD                ; initialize I2C baud rate



            banksel OPTION_REG
            movlw   0x87                   ; PORTB pull-ups disabled, RB0 interrupt on
                                          ; falling edge
            movwf   OPTION_REG


            movlw   0x00
            movwf   PIE2
            movlw   0x1D
            movwf   SSPCON2
            movlw   0x80                  ; I2C mode
            movwf   SSPSTAT               ;


            movlw   0x06
            movwf   ADCON1                ; All pins configured as digital
            movlw   0x00
```

```
            movwf   TRISA               ; Configuring port I/O
            movlw   0xFF
            movwf   TRISB               ; Configuring port I/O
            movlw   0x00
            movwf   TRISC               ; Configuring port I/O


            banksel PORTA
            movlw   0x00
            movwf   PORTA               ; Clearing registers to known states
            movwf   PORTB
            clrf    PORTC


            movlw   0x28
            movwf   SSPCON              ; Enable MSSP, master mode, hardware controlled
            movlw   0x00
            movwf   ADCON0              ; A/D is off


            ;goto   MCP3221_CONTINUOUS_CONVERSION ; Uncomment this line for Continuous conversion


;-----------------------------------------------------------------------
;  ****************** SINGLE CONVERSION LOOP  ******************
;-----------------------------------------------------------------------


MCP3221_SINGLE_CONVERSION

            banksel temp_address
            clrf    temp_address
            movlw   b'10011011'       ; Load MCP3221 Address into register
            movwf   temp_address


            call    WrtStop             ; Send Stop Bit to Reset
            call    check_idle
            call    WrtStart            ; Start Bit
            call    check_idle
            call    SendWrtAddr         ; Send Address Byte


            call    check_idle
            call    StartReadDataHigh   ; Get High Byte of A/D Conversion
            call    check_idle
            call    SendReadAck         ; Acknowledge Low Byte


            call    check_idle
            call    StartReadDataLow    ; Get Low Byte of A/D Conversion
            call    check_idle
            call    SendReadNack        ; Non-Acknowledge to signal single conversion


            call    check_idle


            call    WrtStop             ; Send Stop Bit
            call    check_idle
```

```
            goto    MCP3221_SINGLE_CONVERSION
;*****************************************************************
;  ****************** CONTINUOUS CONVERSION LOOP  ******************
;                       (3 continous conversions implemented here)
;*****************************************************************

MCP3221_CONTINUOUS_CONVERSION

            banksel temp_address
            clrf    temp_address
            movlw   b'10011011'
            movwf   temp_address


            call    WrtStop                 ; Send Stop Bit to Reset
            call    check_idle
            call    WrtStart                ; Start Bit
            call    check_idle              ;
            call    SendWrtAddr             ; Send Address Byte

            call    check_idle
            call    StartReadDataHigh       ; Get High Byte of A/D Conversion #1
            call    check_idle
            call    SendReadAck             ; Acknowledge Low Byte


            call    check_idle
            call    StartReadDataLow        ; Get Low Byte of A/D Conversion #1
            call    check_idle
            call    SendReadAck              ; Acknowledge Low Byte to signal continuous
                                            ; conversion

            call    check_idle
            call    StartReadDataHigh       ; Get High Byte of A/D Conversion #2
            call    check_idle
            call    SendReadAck             ; Acknowledge Low Byte


            call    check_idle
            call    StartReadDataLow        ; Get Low Byte of A/D Conversion #2
            call    check_idle
            call    SendReadAck              ; Acknowledge Low Byte to signal continuous
                                            ; conversion


            call    check_idle
            call    StartReadDataHigh       ; Get High Byte of A/D Conversion #3
            call    check_idle
            call    SendReadAck             ; Acknowledge Low Byte


            call    check_idle
            call    StartReadDataLow        ; Get Low Byte of A/D Conversion #3
            call    check_idle
            call    SendReadNack             ; Non-acknowledge Low Byte to signal end of
                                            ; continuous conversion


            call    check_idle


            call    WrtStop                 ; Send Stop Bit
            call    check_idle
```

```
            goto MCP3221_CONTINUOUS_CONVERSION
;-----------------------------------------------------------------------
;  ****************** MSSP I2C SUBROUTINES ******************
;-----------------------------------------------------------------------


; Generate I2C bus start condition
WrtStart

            banksel SSPCON2             ; select SFR bank
            bsf     SSPCON2,SEN         ; initiate I2C bus start condition
            banksel PIR1
            btfss   PIR1,SSPIF
            goto    $-1
            banksel portc
            return                      ;


; Generate I2C address write
SendWrtAddr
            banksel temp_address        ; select GPR bank
            movf    temp_address,w
            banksel SSPCON2
            bcf     SSPCON2,RCEN
            banksel SSPBUF              ; select SFR bank
            movwf   SSPBUF              ; initiate I2C bus write condition
            banksel PIR1
            clrf    PIR1
            btfss   PIR1, SSPIF
            goto    $-1
            banksel portc
r           return;


; Generate I2C bus stop condition
WrtStop
            banksel SSPCON2             ; select SFR bank
            btfss   SSPCON2,ACKSTAT     ; test for acknowledge from slave
            goto    noerror             ; bypass setting error flag
            banksel eflag_event         ; select GPR bank
            bsf     eflag_event,ack_error ; set acknowledge error

noerror
            banksel SSPCON2             ; select SFR bank
            bsf     SSPCON2,PEN         ; initiate I2C bus stop condition
            return                      ;


; Check for MSSP Idle state
CHECK_IDLE
            global  CHECK_IDLE
            banksel SSPSTAT
            btfsc   SSPSTAT, R_W        ;transmit in progress?
            goto    $-1
            movf    SSPCON2, 0          ;get copy of SSPCON2
            andlw   0x1F                ;mask non-status
            btfss   STATUS, Z
            goto    $-3                 ;bus busy, test again
            banksel PIR1
            bcf     pir1,3
            return
```

```
;Read byte from Slave, save in ASAMH
StartReadDataHigh
            banksel sspcon2
            bsf     SSPCON2,RCEN            ; generate receive condition
            banksel PIR1
            btfss   PIR1,SSPIF
            goto    $-1
            movf    SSPBUF,w                ; save off byte into W
            movwf   asamh                   ; Save MCP3221 high byte into ASAMH FSR
            return


;READ byte from Slave, save in ASAML
StartReadDataLow
            banksel sspcon2
            bsf     SSPCON2,RCEN            ; generate receive condition
            banksel PIR1
            btfss   PIR1,SSPIF
            goto    $-1
            movf    SSPBUF,w                ; save off byte into W
            movwf   asaml                   ; Save MCP3221 low byte into ASAML FSR
            return


; Send Non Acknowledge
SendReadNack

            banksel SSPCON2                 ; select SFR bank
            bsf     SSPCON2,ACKDT           ; acknowledge bit state to send (not ack)
            bsf     SSPCON2,ACKEN           ; initiate acknowledge sequence
            banksel PIR1
            btfss   PIR1,SSPIF
            goto    $-1
            banksel portc
            return


; Send Acknowledge
SendReadAck

            banksel SSPCON2                 ; select SFR bank
            bcf     SSPCON2,ACKDT           ; acknowledge bit state to send
            bsf     SSPCON2,ACKEN           ; initiate acknowledge sequence
            btfsc   SSPCON2,ACKEN           ; ack cycle complete?
            goto    $-1                     ; no, so loop again
            banksel PIR1
            btfss   PIR1,SSPIF
            goto    $-1
            banksel portc
            return                          ;




;-----------------------------------------------------------------------
;   ****************** Generic bus idle check **********************
;-----------------------------------------------------------------------
; test for i2c bus idle state; not implemented in this code (example only)
i2c_idle
            banksel SSPSTAT                 ; select SFR bank
            btfsc   SSPSTAT,R_W             ; test if transmit is progress
            goto    $-1                     ; module busy so wait
            banksel SSPCON2                 ; select SFR bank
            movf    SSPCON2,w               ; get copy of SSPCON2 for status bits
```

```
andlw   0x1F                ; mask out non-status bits
btfss   STATUS,Z            ; test for zero state, if Z set, bus is idle
goto    $-3                 ; bus is busy so test again
return                      ; return to calling routine

END; required directive
```

APPENDIX B: ASSEMBLY CODE USING SOFTWARE I²C IMPLEMENTATION

```
;*****************************************************************
;                                                                *
;     Bit-bang communication with the MCP3221 ADC using a PICmicro®  *
;                                                                *
;                                                                *
;*****************************************************************
;                                                                *
;   Filename:       software.asm                                 *
;   Date:           06/26/2002                                   *
;   Revision:       1.00                                         *
;                                                                *
;   Tools:          MPLAB   5.55.00                              *
;                   MPLINK  2.50.00                              *
;                   MPASM   2.50.00                              *
;                                                                *
;  Author:          Craig L. King                                *
;                                                                *
;  Company:         Microchip Technology Incorporated            *
;                                                                *
;*****************************************************************
;                                                                *
;   System files required:                                       *
;                                                                *
;                  software.asm                                  *
;                                                                *
;                  p16f876.inc                                   *
;                  16f876.lkr     (modified for interrupts)      *
;                                                                *
;*****************************************************************
;                                                                *
;   Notes:                                                       *
;                                                                *
;   Device Fosc -> 20.00MHz                                      *
;       WDT -> on                                                *
;       Brownout -> on                                           *
;       Powerup timer -> on                                      *
;       Code Protect -> off                                      *
;                                                                *
;                                                                *
;                                                                *
;                                                                *
;*****************************************************************/

        list      p=16f876                  ; list directive to define processor
        #include <p16f876.inc>              ; processor specific variable definitions
        __CONFIG (_CP_OFF & _WDT_ON & _BODEN_ON & _PWRTE_ON & _HS_OSC & _WRT_ENABLE_ON & _LVP_OFF
& _CPD_OFF)


        errorlevel -302




#define  ack_error  0                     ;flag bit
#define  do         0                     ;transmit bit
#define  di         1                     ;transmit bit
#define  scl  portc,3                     ;pin used as scl
#define  sda  portc,4                     ;pin used as sda
```

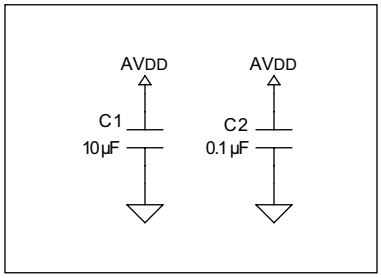
```
        udata


eflag_event     res      1              ; variable for i2c error status flags
txbuf           res      1              ; buffer for i2c bytes in transit
count           res      1              ; bit count variable
dtemp           res      1              ; transmit variable
address         res      1
asamh           res      1
asaml           res      1
datai           res      1




;-------
;-Code--
;-------

prog1     code



;*****************************************************************
;       MCP3221_SINGLE_CONVERSION Routine
;*****************************************************************
;

bb3221

        movlw   b'10011011'             ;set Address
        movwf   address


        clrf    eflag_event             ; clear all error flags

        call    BSTART                  ; generate start bit
        movf    address,w               ; move address
        movwf   txbuf                   ; into transmit buffer
        call    TX_BYTE                 ; and send it
        call    RX_BYTE                 ; read first byte from MCP3221
        movf    datai,w                 ; save data
        movwf   asamh                   ; to variable
        call    RX_BYTE2                ; read second byte from MCP3221
        movf    datai,w                 ; save data
        movwf   asaml                   ; to variable
        call    BSTOP                   ; send stop bit to end transmission

        goto    bb3221                  ; return
```

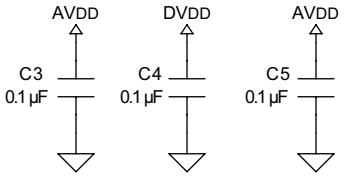
```
;**************
;**************
;Subroutines
;**************
;**************




;*************************************************************
;       Start Bit Subroutine
;       this routine generates a start bit
;       (Low going data line while clock is high)
;*************************************************************
;
BSTART
         bsf     sda                     ; make sure data is high
         movlw   b'11100111'
         tris    portc                   ; set data and clock lines for output
         bcf     scl                     ; make sure clock is low
         nop
         bsf     scl                     ; set clock high
         nop
         nop
         nop
         nop
         bcf     sda                     ; data line goes low during
                                         ; high clock for start bit
         nop
         nop
         nop
         nop
         nop                             ; timing adjustment
         bcf     scl                     ; start clock train
         nop
         nop
         nop
         nop
         nop
         nop
         retlw   0




;*************************************************************
;       Stop Bit Subroutine
;       This routine generates a stop bit
;       (High going data line while clock is high)
;*************************************************************
;
BSTOP
         movlw   b'11100111'
         tris    portc                   ; set data and clock lines for output
         bcf     sda                     ; make sure data line is low
         nop
         nop
         nop
         bsf     scl                     ; set clock high
         nop
         nop
         nop
         bsf     sda                     ; data goes high while clock high
                                         ; for stop bit
```

```
         nop
         nop
         bcf     scl                     ; set clock low again
         nop
         nop
         nop
         retlw   0



;*************************************************************
;       BITOUT routine takes one bit of data in 'do' and
;       transmits it to the serial EE device
;*************************************************************
BITOUT
         movlw   b'11100111'
         tris    portc                   ; set data and clock lines for output
         btfss   dtemp,do                ; check for state of data bit to xmit
         goto    bitlow                  ;
         bsf     sda                     ; set data line high
         goto    clkout                  ; go toggle the clock

bitlow   bcf     sda                     ; output a low bit
clkout   bsf     scl                     ; set clock line high
         nop
         nop
         nop
         nop
         bcf     scl                     ; return clock line low
         retlw   0




;*************************************************************
;       BITIN routine reads one bit of data from the
;       serial EE device and stores it in 'di'
;*************************************************************
BITIN
         bsf     dtemp,di                ; assume input bit is high
         movlw   b'11110111'
         tris    portc                   ; set data and clock lines for output

         bsf     sda                     ; set sdata line for input
         bsf     scl                     ; set clock line high
         nop                             ; just sit here a sec
         nop
         nop
         nop
         nop                             ;
         btfss   sda                     ; read the data bit
         bcf     dtemp,di                ; input bit was low
         bcf     scl                     ; set clock line low
                                         ;
         retlw   0                       ;



;*************************************************************
;       Transmit Byte Subroutine
;       This routine takes the byte of data stored in the
```

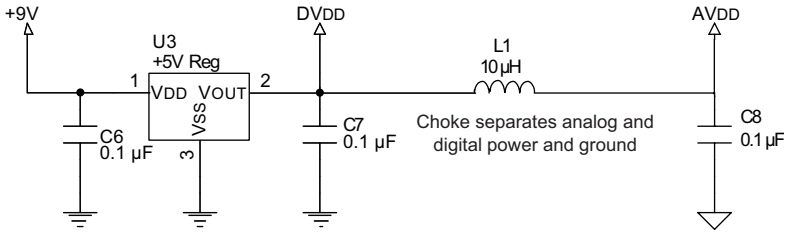
```
;       'datao' register and transmits it to the serial EE device.
;       It will then send 1 more clock to the serial EE for the
;       acknowledge bit.  If the ack bit from the part was low
;       then the transmission was sucessful.  If it is high, then
;       the device did not send a proper ack bit and the ack
;       fail LED will be turned on.
;*****************************************************************
TX_BYTE
         movlw   .8
         movwf   count                    ; set the #bits to 8
         ;
TXLP
         bcf     dtemp,do                 ; assume bit out is low
         btfsc   txbuf,7                  ; is bit out really low?
         bsf     dtemp,do                 ; otherwise data bit =1
         call    BITOUT                   ; serial data out
         rlf     txbuf, F                 ; rotate txbuf left
         decfsz  count, F                 ; 8 bits done?
         goto    TXLP                     ; no - go again
         call    BITIN                    ; read ack bit
         btfsc   dtemp,di                 ; check ack bit
         bsf     eflag_event,ack_error    ; set acknowledge fail flag
         ;
         retlw   0


;*****************************************************************
;       Receive Byte routine
;       This routine reads one byte of data from the part
;       into the 'datai' register.  It then sends a high
;       ack bit to indicate that no more data is to be read
;*****************************************************************
RX_BYTE
         clrf    datai                   ; clear input buffer
         movlw   .8                      ; set # bits to 8
         movwf   count
         bcf     status,0                ; make sure carry bit is low
RXLP     rlf     datai, F                ; rotate datai 1 bit left
         call    BITIN                   ; read a bit
         btfsc   dtemp,di
         bsf     datai,0                 ; set bit 0 if necessary
         decfsz  count, F                ; 8 bits done?
         goto    RXLP                    ; no, do another
         bcf     dtemp,do                ; set ack bit = 0
         call    BITOUT                  ; to finish transmission
         retlw   0

RX_BYTE2
         clrf    datai                   ; clear input buffer
         movlw   .8                      ; set # bits to 8
         movwf   count
         bcf     status,0                ; make sure carry bit is low
RXLP2    rlf     datai, F                ; rotate datai 1 bit left
         call    BITIN                   ; read a bit
         btfsc   dtemp,di
         bsf     datai,0                 ; set bit 0 if necessary
         decfsz  count, F                ; 8 bits done?
         goto    RXLP2                   ; no, do another
         bsf     dtemp,do                ; set ack bit = 1
         call    BITOUT                  ; to finish transmission
         retlw   0

    END
```

APPENDIX C: SCHEMATIC

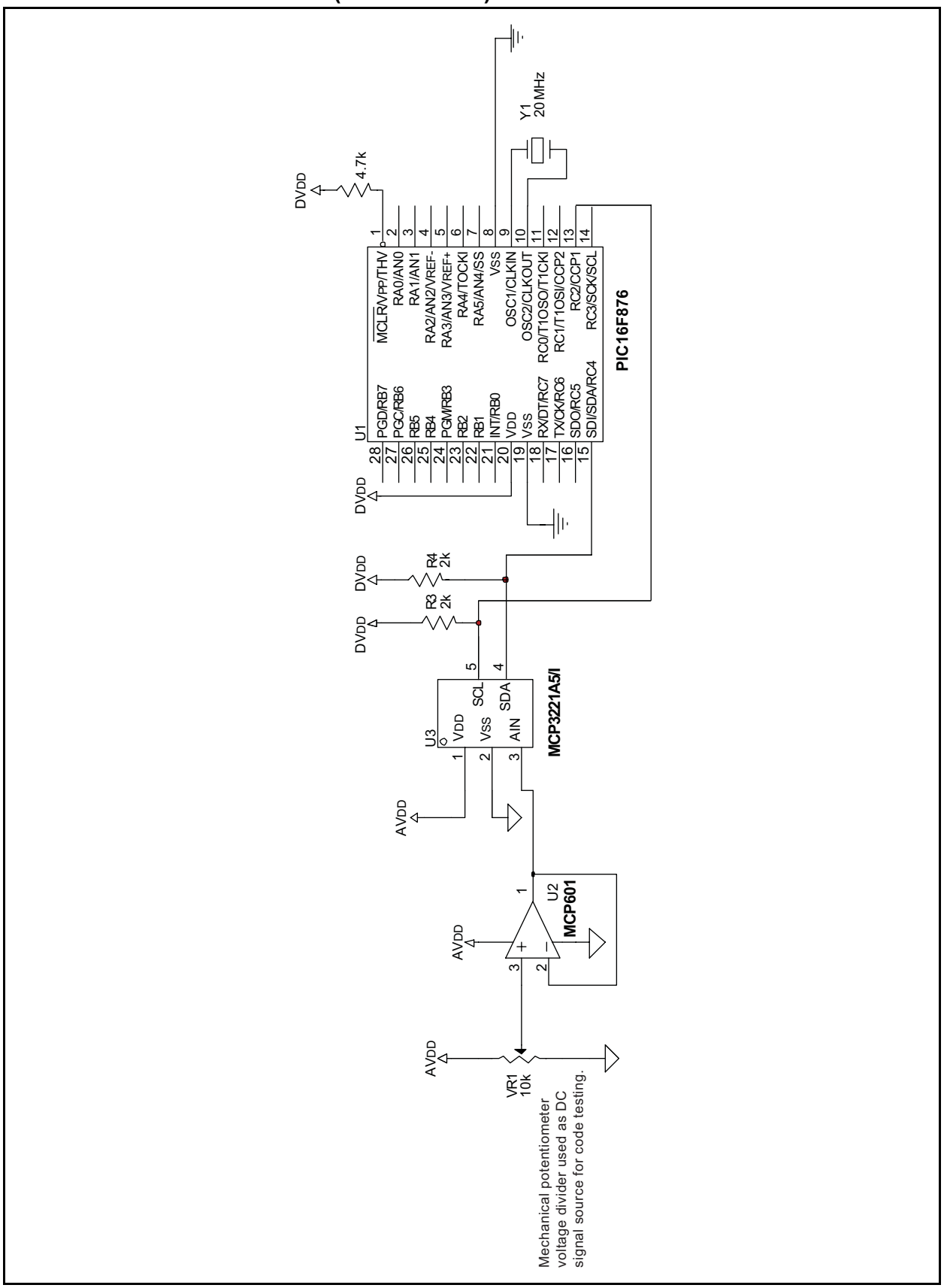


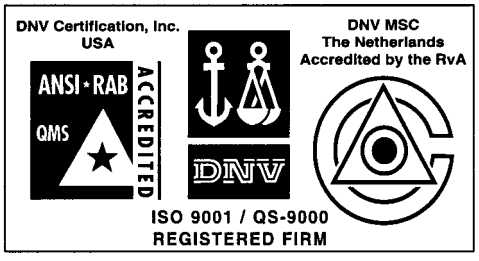


Bypass Capacitors for MCP3221 device. Smaller value should be placed closest to pin.



APPENDIX C: SCHEMATIC (CONTINUED)





Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-4338

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200 Fax: 86-28-86766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

China - Hong Kong SAR
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Microchip Technology (Barbados) Inc.,
Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Microchip Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

Austria
Microchip Technology Austria GmbH
Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

05/16/02