AN703

Using the MCP320X 12-Bit Serial A/D Converter with Microchip PICmicro® Devices

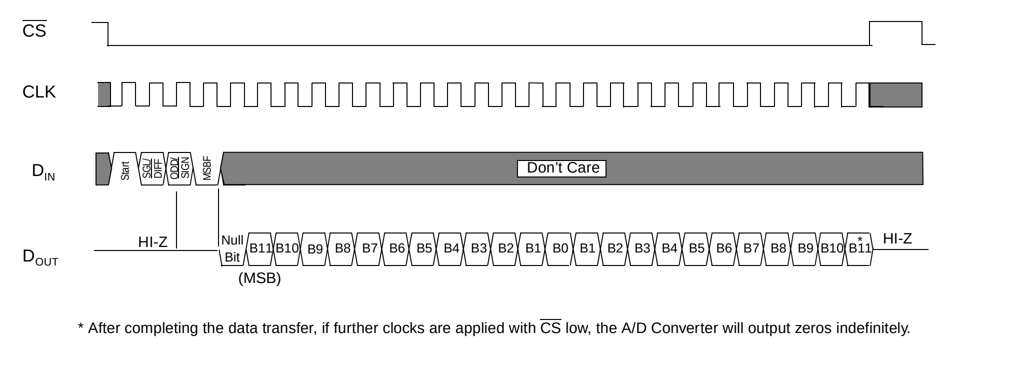
Author: Jake McKernan
Microchip Technology Inc.

OVERVIEW

The MCP320X devices comprise a family of 12-bit successive approximation Analog to Digital (A/D) Converters. These devices provide from one to eight analog inputs with both single ended and differential inputs. Data is transferred to and from the MCP320X through a simple SPI®-compatible 3-wire interface. This application note discusses how to interface the MCP320X devices to Microchip PICmicro® devices, using both software and hardware SPI with examples shown in C and Assembly languages. The programs in this application note were developed using a PIC16C62A and MCP3202 on a PICDEM-2 demonstration board. As a matter of convenience, the CLK, DO, and DI pins of the PIC16C62A are used for all examples, whether using the hardware SPI peripheral or the software SPI implementation. The software SPI may be adapted to I/O ports on any PICmicro device.

COMMUNICATION

Communication to the MCP3202 is accomplished via a synchronous SPI-compatible scheme. This interface consists of three lines; DOUT, DIN and CLK. Control information is loaded into the MCP320X through the DIN line and data is output on the DOUT line. The CLK signal is generated by the PICmicro and is used as both communication and conversion clock for the A/D Converter. Data bits are latched in from DIN on the rising edge of CLK and latched out to DOUT on the falling edge. A fourth line, $\overline{CS}$, is an active low signal used to select the chip and enable it for conversion and communication. See Figure 1 for a communication timing diagram.



* After completing the data transfer, if further clocks are applied with $\overline{CS}$ low, the A/D Converter will output zeros indefinitely.

FIGURE 1: COMMUNICATION WITH MCP3202 USING LSB FIRST FORMAT

SPI is a registered trademark of Motorola

© 1999 Microchip Technology Inc.

A 4-bit configuration command is issued to the MCP3202 to begin the conversion process. When communication of the command word to the MCP3202 begins, the first ‘1’ bit seen by the MCP3202 on the DIN line will be interpreted as a start bit. Leading 0’s may be clocked into the device with no effect. The start bit is followed by a mode selection bit, indicating whether the conversion result will be single-ended or differential. A mode select bit of '1' selects single-ended mode and '0' selects differential mode. Next, the channel select bit is clocked into the MCP3202, which sets the channel to be converted. A '0' in this bit position selects Channel 0, while a '1' selects Channel 1. If differential mode was selected, the channel select bit determines which channel will be subtracted from the other. Table 1 illustrates how the A/D result will be affected by the channel and mode selection bits. Finally, a data format bit is clocked into the MCP3202. This bit selects whether the result of the conversion will be shifted out in LSb format. A '0' in this bit position will cause the data to be shifted out in MSb only format. If a '1', the data will first be shifted out in MSb format, followed by the same data in LSb format. Keep in mind that the data will *always* be shifted out in MSb format, regardless of the state of the data format bit.

	CONFIG BITS		**CHANNEL SELECTION**		**GND**
	SGL/$\overline{\text{DIFF}}$	ODD/$\overline{\text{SIGN}}$	0	1	
SINGLE ENDED MODE	1	0	+		-
	1	1		+	-
PSEUDO-DIFFERENTIAL MODE	0	0	IN+	IN-	
	0	1	IN-	IN+	

TABLE 1: CONFIGURATION BITS FOR THE MCP3202

The command word is followed by the clocking in of a dummy bit, during which time the converter determines whether the MSb should be a 0 or 1. The 12-bit A/D result is then clocked out of the MCP3202 one bit at a time. The LSb of the A/D result is common to both data formats, i.e. the LSb is output only once while all other result bits are output twice (once for MSb first format, once for LSb first format). 0's will be clocked out of the DOUT line if CLK pulses are issued after all data bits are extracted from the converter.

IMPLEMENTATION

As previously mentioned, several code examples of interfacing to the MCP3202 are shown in this application note. All methods use essentially the same algorithm of performing an A/D conversion, displaying the result on PORTB, then waiting for a keypress. The examples cover hardware and software SPI, relocatable and absolute assembly and C.

Written in absolute assembly, Appendix A shows the use of the hardware SSP module in master SPI mode. The SSP is set up to clock data in on the rising edge, clock data out on the falling edge and drive the clock high when idle, with a frequency of Fosc/64. All bits of PORTB are configured as outputs and the port is cleared. To begin the conversion process, the MCP3202 is selected using the $\overline{\text{CS}}$ line and 0x01 is loaded into the SSPBUF of the 16C62A. This shifts out seven leading 0’s, followed by a start bit. The subroutine WAIT_BF then monitors the BF flag in the SSPSTAT register, which indicates when the 8-bit transfer is complete. Next, a value of 0xE0 is loaded into the SSPBUF, the MSb’s being the three configuration information bits, and the lower five bits being dummy information to round out the byte. The configuration bits in this example set the MCP3202 up for single-ended conversion on channel 1, with the output in MSb first format. During the transmission of the 5 LSb’s, the MCP3202 will begin shifting out A/D result data. The WAIT_BF subroutine is called after the SSPBUF is loaded, waiting for the transmission to be complete. Once the transmission is complete, the MSb’s of the result are read from the SSPBUF, masked, and displayed on PORTB for examination by the user. Finally, a dummy value of 0x00 is loaded into the SSPBUF to retrieve the final eight LSb’s of the A/D result from the MCP3202.

The WAIT_PRESS routine is then called, waiting for the RA4 button of the PICDEM-2 board to be pressed and released. Once the button has been pressed and released, the remaining data is read from the SSPBUF and displayed on the PORTB pins. This information is displayed until the RA4 button is again pressed and released (by calling the WAIT_PRESS subroutine), after which the A/D process begins again.

Appendix B demonstrates the same functionality as the program in Appendix A, but is written in the C language. This allows portability between platforms (12-bit, 14-bit or 16-bit cores), with a minimum of change to the program.

Appendices C and D are used together to show a hardware SPI implementation using relocatable assembly code. The main file (MCP3202c.asm) is shown in Appendix C and contains the main functionality of the program, while the assembly file shown in Appendix D (waitfcn.asm) contains the auxiliary functions (i.e. waiting for SPI transmission to complete and for RA4 press and release). The linker script (16c62a.lkr) shown in Appendix D controls where the relocatable segments

© 1999 Microchip Technology Inc.

are placed in the 16C62A program memory and defines the processor's available RAM space for the linker. Please consult the MPASM User's Guide for more details on how to write relocatable code.

Appendix E illustrates communication to the MCP3202 using firmware SPI rather than the hardware peripheral. The same I/O pins are used to generate the clock and data signals as with the hardware peripheral, for convenience. Program initialization occurs as with the previous examples, except that the hardware peripheral is excluded and replaced with initialization of PORTC bits. Three registers are initialized to be used as input and output buffers, and there are two new subroutines added to communicate to the MCP3202. The first routine called will be OUT_CONTROL, which issues the control word to the MCP3202. The control word to be sent is loaded into the OUTBUF register before the subroutine is called. Each of the four bits is then shifted out and clocked into the A/D Converter using the DOUT and CLK lines of PORTC, respectively. Once all bits are shifted out, the subroutine returns to the calling function. To retrieve the data from the A/D Converter, a second subroutine is implemented. The IN_DATA subroutine toggles the CLK line and reads the DIN line, shifting each new bit into the INBUFL and INBUFH registers. All 12 bits of the result are read by this subroutine which will return to the calling function once the transfer is complete. As with the previous examples, the MSb's are displayed on PORTB, while the program waits for RA4 to toggle. The LSb's are then displayed, the program waits for RA4 to toggle again, and the process repeats again.

Appendix F is a variation on Appendix E, demonstrating the use of relocatable assembly to implement a software SPI. The same subroutines are used for this example, but are declared as external. The wait functions and linker script (waitfcn.asm, 16c62a.lkr) files shown in Appendix C are used in this example. The ser_io.asm file shown in Appendix G contains the OUT_CONTROL and IN_DATA subroutines used in this example.

The final example, shown in Appendix H, illustrates the firmware SPI implementation in the C language. Two functions are added to this implementation, Output_Control and Input_Data. As with the previous example, the Output_Control shifts the 4-bit command out to the MCP3202 one bit at a time and Input_Data reads all 12 bits of the result. The data is then displayed on PORTB, waiting for input on RA4 before continuing on. In this program, the A/D result data may be accessed in one of two ways; as a 16-bit value or as two 8-bit values. When reading the value in from the MCP3202 using the Input_Data function, the A/D result is treated as a 16-bit value. During the display portion of the program, the result is accessed 8-bits at a time for display on PORTB.

SCHEMATIC

The code for this application note was developed on a PICDEM-2 demonstration board. An equivalent circuit of the board as used in this application note is shown in Appendix I. A full schematic of the PICDEM-2 board can be found in the PICDEM-2 User's Guide, available with the kit or from the Microchip web site (www.microchip.com).

The SPI communication lines CLK, DOUT and DIN are connected to RC3, RC4 and RC5, respectively. The $\overline{CS}$ signal is generated using RC2 as a general purpose output pin. PORTB is used entirely as an output port for display of A/D result data. All LED's are driven through 470Ω current limiting resistors. RA4 is connected to a momentary contact switch and pullup resistor for allowing the user to cycle through the A/D result data on PORTB.

Channel 1 of the A/D Converter is used throughout the application note, and must have an analog voltage applied to it to get meaningful results from the MCP3202. This was done using a 0-5v power supply output fed directly into pin three of the MCP3202.

The PIC16C62A uses the RC oscillator configuration as the main clock, operating at an approximate frequency of 4MHz. An RC network is also provided on the MCLR line to help ensure that the device is reset correctly on application of power.

CONCLUSION

The example code shown in this application note gives a firm grasp of how to interface the MCP3202 A/D Converter to PICmicro devices. The code has the potential to be adapted to any Microchip PICmicro device, an exercise left up to the user. Implementations in multiple languages and styles also gives the developer flexibility in successfully writing code and libraries to use this device in end-user applications.

© 1999 Microchip Technology Inc.

APPENDIX A: HARDWARE SPI, ABSOLUTE ASSEMBLY

```
;*********************************************************************************
;*
;*     This program demonstrates communication with the MCP3202 A/D converter
;*     using absolute assembly code.  This code was written for the midrange
;*     PICmicro devices (using a PICDEM-2 board and the 16C62A) and uses the SSP
;*     module in SPI mode for communication to the MCP3202.
;*
;*     Filename: mcp3202a.asm
;*
;*     (C) 1998 Microchip Technology, Inc.
;*     All Rights Reserved
;*
;*********************************************************************************

   list p=16c62a

   include “p16c62a.inc”


ADCS   equ    0x02                 ;chip select line for A/D


   ORG 0x0000

   clrf PCLATH                 ;reset PCLATH for Page0 operation
   clrf STATUS                 ;reset STATUS for Bank 0 operation
   clrf FSR                    ;clear FSR
   goto START                  ;begin main program

   ORG 0x0004
_ISR
   goto _ISR                   ;stay here if interrupt occurs

WAIT_BF
      bsf STATUS,RP0              ;select Bank0
      btfss SSPSTAT,BF            ;check for BF set
      goto WAIT_BF                ;continue to wait
      bcf STATUS,RP0              ;select Bank1
      return                      ;return to caller


WAIT_PRESS
      btfsc PORTA,4               ;check for button press
      goto WAIT_PRESS

WAIT_RLS
      btfss PORTA,4               ;check for button release
      goto WAIT_RLS
      return                      ;return to caller


START
      movlw 0x32                  ;set up SSP to clock data out on falling edge
      movwf SSPCON                ;clock data in on rising edge, clock idle high

      clrf PORTB                  ;clear PortB outputs

   bsf STATUS,RP0              ;select Bank1
   movlw 0x10
   movwf TRISC                 ;set up Port C for SPI master

   clrf TRISB                  ;configure PortB as outputs

   bcf STATUS,RP0              ;select Bank0
   bsf PORTC,ADCS              ;deselect A/D device

BEGIN_AD
      bcf PORTC,ADCS              ;select A/D device
      movlw 0x01
      movwf SSPBUF                ;output start bit

      call WAIT_BF                ;wait for transfer complete

      movlw 0xE0                  ;output 3 command and 5 dummy bits
      movwf SSPBUF                ;shift out command and receive 4 MSb’s
      call WAIT_BF                ;wait for transfer complete

      movf SSPBUF,W               ;read result (MSB’s of conversion)
      andlw 0x0F                  ;mask out MSb’s
      movwf PORTB                 ;display on PortB

      movlw 0x00                  ;load dummy value
      movwf SSPBUF                ;shift remaining bits
      call WAIT_BF                ;wait for transfer complete

      call WAIT_PRESS             ;wait for button press/release before advancing

      movf SSPBUF,W               ;read result (LSb’s)
      movwf PORTB                 ;display on PortB

      bsf PORTC,ADCS              ;de-select A/D converter

      call WAIT_PRESS             ;wait for button press/release before advancing

HERE
      goto BEGIN_AD               ;play it again, Sam


      END
```

APPENDIX B: HARDWARE SPI, C LANGUAGE

```
/*************************************************************************
*
*  This program is written to demonstrate interfacing the MCP3202 A/D
*  converter to Microchip PICmicro devices.  The code demonstrates
*  how to implement hardware SPI to communicate with the converter,
*  and is written in C for the HiTech PICC C compiler.  By modifying the
*  #include statement to “#include<16c62a.h>” the code may be compiled
*  using MPLAB-C 1.21.
*
*  Filename: mcp3202b.c
*
*  (C) 1998 Microchip Technology, Inc.
*  All Rights Reserved
*
*************************************************************************/


#include<pic1662.h>       /* modify this statement for use with the MPLAB-C compiler */


#define ADCS 0x04        /* I/O bit position for CS line */
#define BUSY 0x01        /* Bit0 of SSPSTAT, indicated when SPI xmission complete */
#define BUTTON 0x10      /* I/0 bit position for RA4 line */


void Wait_for_Press()
{
   while(PORTA & BUTTON)
   {
      /* wait for button press */
   }

   while(!(PORTA & BUTTON))
   {
      /* wait for button release */
   }
}


void main(void)
{
   TRISB = 0x00;
   PORTB = 0x00;         /* reset PortB outputs */

   SSPCON = 0x32;        /* set up SSP to clock data out on falling edge */
   TRISC = 0x10;         /* clock data in on rising edge, clock idle high */

   PORTC |= ADCS;        /* de-select A/D device */

   while(1)
   {
      PORTC &= ~ADCS;    /* select A/D device */

      SSPBUF = 0x01;     /* output start bit */

      while(!(SSPSTAT & BUSY))
      {
                         /* wait for transfer complete */
      }
```

© 1999 Microchip Technology Inc.

```
      SSPBUF = 0xE0;   /* output 3 command, 5 dummy bits */

      while(!(SSPSTAT & BUSY))
      {
                       /* wait for transfer complete */
      }

      PORTB = SSPBUF & 0x0F;      /* mask and output conversion MSb’s */

      SSPBUF = 0x00;       /* output dummy word */

      while(!(SSPSTAT & BUSY))
      {
                           /* wait for transfer complete */
      }

      PORTC |= ADCS;       /* de-select A/D device */

      Wait_for_Press();    /* wait for button press/release */

      PORTB = SSPBUF;      /* output LSb’s */

      Wait_for_Press();    /* wait for button press/release */
   }
}
```

© 1999 Microchip Technology Inc.

APPENDIX C: HARDWARE SPI, RELOCATABLE ASSEMBLY

```
;*********************************************************************************
;*
;*     This program demonstrates communication with the MCP3202 A/D converter
;*     using relocatable assembly code.  This code was written for the midrange
;*     PICmicro devices (using a PICDEM-2 board and the 16C62A) and uses the SSP
;*     module in SPI mode for communication to the MCP3202.
;*
;*     The two subroutines WAIT_BF and WAIT_PRESS are external functions, compiled
;*     and linked separately from the WAITFCN.ASM file.  These subroutines wait
;*     for the SPI transmission to complete and for RA4 to be pushed and released,
;*     respectively.
;*
;*     Filename: mcp3202c.asm
;*
;*     (C) 1998 Microchip Technology, Inc.
;*     All Rights Reserved
;*
;*********************************************************************************

list p=16C62a

#include “p16c62a.inc”

ADCSequ0x02                        ;CS line for MCP3202 (RC6)

EXTERN WAIT_BF                     ;define wait function call symbols
EXTERN WAIT_PRESS


RESETCODE                          ;select reset code section

clrf PCLATH                        ;reset PCLATH on powerup
clrf STATUS                        ;reset STATUS on powerup
clrf FSR                           ;reset FSR on powerup
goto START                         ;go start and initialize program

INTCODE                            ;select interrupt code section
_ISR
goto _ISR                          ;stay here if interrupt occurs


START                              ;initialization
movlw 0x32
movwf SSPCON                       ;setup SSP for operation

clrf PORTB                         ;reset LED output port

bsf STATUS,RP0                     ;select Bank1
movlw 0x10
movwf TRISC                        ;configure PORTC for operation

clrf TRISB                         ;configure PORTB as outputs

bcf STATUS,RP0                     ;select Bank0
bsf PORTC,ADCS                     ;deselect A/D converter

BEGIN_AD                           ;start A/D conversion
bcf PORTC,ADCS                     ;select A/D converter
movlw 0x01                         ;load start bit
```

```
movwf SSPBUF                       ;output start bit to A/D

call WAIT_BF                       ;wait for transmission complete

movlw 0xE0                         ;load 3 command and 5 dummy bits
movwf SSPBUF                       ;output on SPI port

call WAIT_BF                       ;wait for transmission complete

movf SSPBUF,W                      ;read A/D result MSb’s
   andlw 0x0F                      ;mask off garbage bits
movwf PORTB                        ;output MSb’s on PORTB LED’s

movlw 0x00                         ;load dummy data
movwf SSPBUF                       ;output on SPI (shifts in LSb’s)
call WAIT_BF                       ;wait for transmission complete

call WAIT_PRESS                    ;wait for button press/release

movf SSPBUF,W                      ;read A/D result LSb’s
movwf PORTB                        ;output LSb’s on PORTB LED’s

bsf PORTC,ADCS                     ;deselect A/D converter

call WAIT_PRESS                    ;wait for button press/release

HERE
goto BEGIN_AD                      ;repeat process

END
```

APPENDIX D: WAIT FUNCTIONS AND LINKER SCRIPT FOR APPENDIX C

```
;*************************************************************************
;*
;*     Wait functions for MCP3202 A/D converter demonstration.  These
;*     functions wait for SPI communication and RA4 button press/release
;*     on the PICDEM-2 board.  This file is to be assembled and linked
;*     with mcp3202c.ASM or mcp3202e.ASM for proper usage.
;*
;*     Filename: waitfcn.asm
;*
;*     (C) 1998 Microchip Technology, Inc.
;*     All Rights Reserved
;*
;*************************************************************************

       list p=16C62a
       #include “p16c62a.inc”
       CODE                             ;select code section

WAIT_BF                                 ;wait for SPI transmission complete
       GLOBAL WAIT_BF                   ;declare WAIT_BF visible to outside world
       bsf STATUS,RP0                   ;select Bank1
       btfss SSPSTAT,BF                 ;check for transmission complete (BF set)
       goto WAIT_BF                     ;not finished, continue waiting
       bcf STATUS,RP0                   ;select Bank0
       return                           ;return to calling function

WAIT_PRESS                              ;wait for RA4 press/release
       GLOBAL WAIT_PRESS                  ;declare WAIT_PRESS visible to outside world

       btfsc PORTA,4                    ;check for button press
       goto WAIT_PRESS                  ;not pressed, check again

WAIT_RLS                                ;button now pressed
       btfss PORTA,4                    ;check for button release
       goto WAIT_RLS                    ;not released, check again
       return                            ;button now released, return to calling func

       END

//*************************************************************************
//*
//*      16C62A Linker Script to be used with MCP3202C.ASM and WAITFCN.ASM
//*      to link the corresponding object files.
//*
//*      Filename: 16c62a.lkr
//*
//*      (C) 1998 Microchip Technology, Inc.
//*      All Right Reserved
//*
//*************************************************************************

CODEPAGE NAME=reset_vector START=0x00 END=0x03
CODEPAGE NAME=interrupt_vector START=0x04 END=0x7FF
DATABANK   NAME=gpr0     START=0x20     END=0x7F
DATABANK   NAME=gpr1     START=0xA0     END=0xBF
DATABANK   NAME=sfr0     START=0x0      END=0x1F     PROTECTED
DATABANK   NAME=sfr1     START=0x80     END=0x9F     PROTECTED
SECTION NAME=RESET ROM=reset_vector
SECTION NAME=INT ROM=interrupt_vector
```

© 1999 Microchip Technology Inc.

APPENDIX E: FIRMWARE SPI, ABSOLUTE ASSEMBLY

```
;*****************************************************************************
;*
;*     This program demonstrates communication with the MCP3202 A/D converter
;*     using absolute assembly code.  This code was written for the midrange
;*     PICmicro devices (using a PICDEM-2 board and the 16C62A) and uses firmware
;*     to implement the SPI module for communication to the MCP3202.
;*
;*     Filename: mcp3202d.asm
;*
;*     (C) 1998 Microchip Technology, Inc.
;*     All Rights Reserved
;*
;*****************************************************************************
list p=16c62a

include “p16c62a.inc”

ADCS   equu    0x02                 ;chip select line for A/D converter
DOUT   equ    0x05                 ;serial data out to A/D converter
DIN    equ    0x04                 ;serial data in from A/D converter
CLK    equ    0x03                 ;serial data clock to A/D converter

       CBLOCK 0x20
OUTBUF
INBUFH
INBUFL
COUNT
       ENDC

ORG 0x0000

clrf PCLATH                 ;reset PCLATH for Page0 operation
clrf STATUS                 ;reset STATUS for Bank 0 operation
clrf FSR                    ;clear FSR
goto START                  ;begin main program

ORG 0x0004
_ISR
goto _ISR                   ;stay here if interrupt occurs

OUT_CONTROL
       movwf OUTBUF         ;load control word into buffer
       swapf OUTBUF         ;rotate control word into position

       movlw 0x04
       movwf COUNT          ;init bit counter

BIT_OUT
       rlf OUTBUF           ;rotate bit into carry
       bcf PORTC,DOUT       ;pre-clear data out
       btfsc STATUS,C       ;check if bit should be set
       bsf PORTC,DOUT       ;set data out

       bsf PORTC,CLK        ;generate clock pulse
       nop
       bcf PORTC,CLK

       decfsz COUNT         ;decrement bit counter
       goto BIT_OUT         ;output next bit
       return                  ;finished, return to caller


IN_DATA
       clrf INBUFH
       clrf INBUFL             ;reset input buffer

       movlw 0x0D
       movwf COUNT             ;init bit counter

BIT_IN
       bsf PORTC,CLK           ;set clock to latch bit
       bcf STATUS,C            ;pre-clear carry
       btfsc PORTC,DIN         ;check for high or low bit
       bsf STATUS,C            ;set carry bit

       rlf INBUFL
       rlf INBUFH              ;rotate bit into position

       bcf PORTC,CLK           ;drop clock for next bit

       decfsz COUNT            ;decrement bit counter
       goto BIT_IN             ;get next bit
       return                  ;return to caller


WAIT_PRESS
       btfsc PORTA,0x04        ;check for button press
       goto WAIT_PRESS

WAIT_RLS
       btfss PORTA,0x04        ;check for button release
       goto WAIT_RLS
       return                  ;return to caller


START
       clrf PORTB              ;clear PortB outputs

       movlw 0x40
       movwf PORTC             ;initialize PortC: ADCS high, DO, CLK low

bsf STATUS,RP0                 ;select Bank1
movlw 0x10
movwf TRISC                    ;set up Port C for SPI master

clrf TRISB                     ;configure PortB as outputs

bcf STATUS,RP0                 ;select Bank0

clrf OUTBUF                    ;reset output buffer
clrf INBUFH                    ;reset input buffer
clrf INBUFL

BEGIN_AD
       bcf PORTC,ADCS          ;select A/D converter

       movlw 0x0F              ;load control word
       call OUT_CONTROL        ;output control word
```

© 1999 Microchip Technology Inc.

```
call IN_DATA                ;read data from A/D converter

bsf PORTC,ADCS              ;de-select A/D converter

movlw 0x0F                  ;load MSB mask
andwf INBUFH,W              ;mask out MSB’s and put result in W
movwf PORTB                 ;output MSB’s

call WAIT_PRESS             ;wait for button press

movf INBUFL,W               ;load LSB’s into W
movwf PORTB                 ;output LSB’s

call WAIT_PRESS             ;wait for button press
goto BEGIN_AD               ;play it again, Sam


END
```

© 1999 Microchip Technology Inc.

APPENDIX F: FIRMWARE SPI, RELOCATABLE ASSEMBLY

```
;*********************************************************************************
;*
;*     This program demonstrates communication with the MCP3202 A/D converter
;*     using relocatable assembly code.  This code was written for the midrange
;*     PICmicro devices (using a PICDEM-2 board and the 16C62A) and uses the SSP
;*     module in SPI mode for communication to the MCP3202.
;*
;*     The subroutine WAIT_PRESS is an external function, compiled and linked
;*     separately from the WAITFCN.ASM file.  This subroutine waits for RA4 to be
;*     pushed and released.
;*     The subroutines OUT_CONTROL and IN_DATA are also external functions, but
;*     compiled and linked from the SER_IO.ASM file.  INBUFH and INBUFL are data
;*     bytes that are used by the IN_DATA routine to return the A/D conversion
;*     result to the calling function.
;*
;*     Filename: mcp3202e.asm
;*
;*     (C) 1998 Microchip Technology, Inc.
;*     All Rights Reserved
;*
;*********************************************************************************
list p=16c62a

include “p16c62a.inc”

       EXTERN WAIT_PRESS
       EXTERN OUT_CONTROL
       EXTERN IN_DATA

       EXTERN INBUFH
       EXTERN INBUFL

ADCS   equ    0x02          ;chip select line for A/D converter


RESET  CODE
clrf PCLATH                 ;reset PCLATH for Page0 operation
clrf STATUS                 ;reset STATUS for Bank 0 operation
clrf FSR                    ;clear FSR
goto START                  ;begin main program


INT    CODE
_ISR
goto _ISR                   ;stay here if interrupt occurs

START
       clrf PORTB           ;clear PortB outputs

       movlw 0x40
       movwf PORTC          ;initialize PortC: ADCS high, DO, CLK low

bsf STATUS,RP0              ;select Bank1
movlw 0x10
movwf TRISC                 ;set up Port C for SPI master

clrf TRISB                  ;configure PortB as outputs

bcf STATUS,RP0              ;select Bank0
```

```
BEGIN_AD
       bcf PORTC,ADCS              ;select A/D converter

       movlw 0x0F                  ;load control word
       call OUT_CONTROL            ;output control word

       call IN_DATA                ;read data from A/D converter

       bsf PORTC,ADCS              ;de-select A/D converter

       movlw 0x0F                  ;load MSB mask
       andwf INBUFH,W              ;mask out MSB’s and put result in W
       movwf PORTB                 ;output MSB’s

       call WAIT_PRESS             ;wait for button press

       movf INBUFL,W               ;load LSB’s into W
       movwf PORTB                 ;output LSB’s

       call WAIT_PRESS             ;wait for button press
       goto BEGIN_AD               ;play it again, Sam


       END
```

APPENDIX G: RELOCATABLE ASSEMBLY FIRMWARE SPI FUNCTIONS FOR APPENDIX F

```
;***********************************************************************
;*
;*     Serial functions for MCP3202 A/D converter demonstration.  These
;*     functions perform SPI communication.  This file is to be assembled
;*     and linked with mcp3202e.ASM for proper usage.
;*
;*     Filename: ser_io.asm
;*
;*     (C) 1998 Microchip Technology, Inc.
;*     All Rights Reserved
;*
;***********************************************************************
list p=16c62a

#include “p16c62a.inc”


DOUT   equu    0x05                ;serial data out to A/D converter
DIN    equ    0x04                ;serial data in from A/D converter
CLK    equ    0x03                ;serial data clock to A/D converter

UDATA 0x20
OUTBUF res 1
INBUFH res 1
INBUFL res 1
COUNT  res 1

       GLOBAL INBUFH
       GLOBAL INBUFL

CODE

OUT_CONTROL
GLOBAL OUT_CONTROL
       movwf OUTBUF                ;load control word into buffer
       rlf OUTBUF
       rlf OUTBUF
       rlf OUTBUF
       rlf OUTBUF                  ;rotate control word into position

       movlw 0x04
       movwf COUNT                 ;init bit counter

BIT_OUT
       rlf OUTBUF                  ;rotate bit into carry
       bcf PORTC,DOUT              ;pre-clear data out
       btfsc STATUS,C              ;check if bit should be set
       bsf PORTC,DOUT              ;set data out

       bsf PORTC,CLK               ;generate clock pulse
       nop
       bcf PORTC,CLK

       decfsz COUNT                ;decrement bit counter
       goto BIT_OUT                ;output next bit
       return                      ;finished, return to caller
```

© 1999 Microchip Technology Inc.

```
IN_DATA
GLOBAL IN_DATA
       clrf INBUFH
       clrf INBUFL                 ;reset input buffer

       movlw 0x0D
       movwf COUNT                 ;init bit counter

BIT_IN
       bsf PORTC,CLK               ;set clock to latch bit
       bcf STATUS,C                ;pre-clear carry
       btfsc PORTC,DIN             ;check for high or low bit
       bsf STATUS,C                ;set carry bit

       rlf INBUFL
       rlf INBUFH                  ;rotate bit into position

       bcf PORTC,CLK               ;drop clock for next bit

       decfsz COUNT                ;decrement bit counter
       goto BIT_IN                 ;get next bit
       return                      ;return to caller

       END
```

© 1999 Microchip Technology Inc.

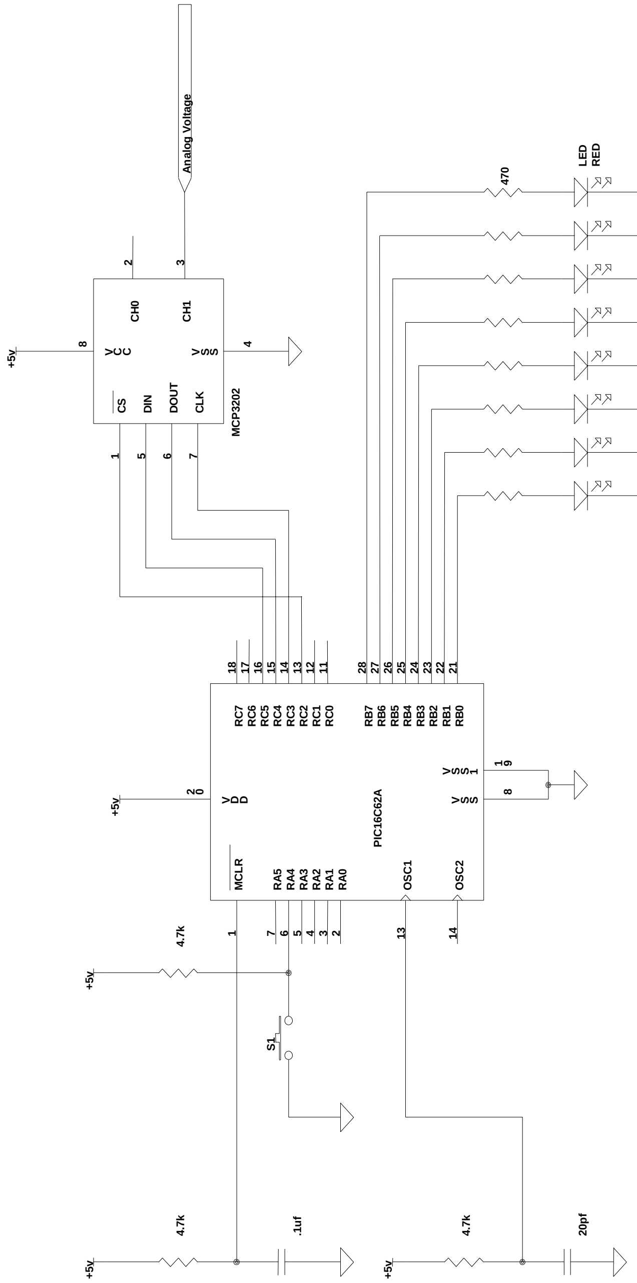
APPENDIX H: FIRMWARE SPI, C LANGUAGE

```
/***********************************************************************
*
*  This program is written to demonstrate interfacing the MCP3202 A/D
*  converter to Microchip PICmicro devices.  The code demonstrates
*  how to implement software SPI to communicate with the converter,
*  and is written in C for the HiTech C compiler, PICC.  Changing the
*  #include directive to “#include<16c62a.h>” will allow the use of the
*  MPLAB-C v1.21 C compiler to compile this file.
*
*  Filename: mcp3202f.c
*
*  (C) 1998 Microchip Technology, Inc.
*  All Rights Reserved
*
***********************************************************************/


#include <pic1662.h>    /* modify this statement for use with the MPLAB-C compiler */

#define ADCS 0x04       /* I/O bit position for CS line */
#define BUSY 0x01       /* Bit0 of SSPSTAT, indicated when SPI xmission complete */
#define BUTTON 0x10     /* I/0 bit position for RA4 line */

#define DOUT 0x20       /* data out to MCP3202 */
#define DIN 0x10        /* data in from MCP3202 */
#define CLK 0x08        /* clock out to MCP3202 */


/* Function Prototypes */

void Wait_for_Press(void);
void Output_Control(char TempChar);
int Input_Data(void);



void Wait_for_Press()
{
   while(PORTA & BUTTON)
   {
      /* wait for button press */
   }

   while(!(PORTA & BUTTON))
   {
      /* wait for button release */
   }
}


void Output_Control(char TempChar)
{
   unsigned char Mask = 0x08;       /* mask to test for 0/1 */
   unsigned char Count;             /* gen purpose bit counter */

   for(Count = 0x00; Count < 0x04; Count++)  /* count 4 bits */
   {
      PORTC &= ~DOUT;               /* pre-clear data line */
      if(TempChar & Mask)           /* check if bit should be high or low */
      {
         PORTC |= DOUT;             /* set data line */
      }

      PORTC |= CLK;                 /* send clock line high */

      Mask >>= 0x01;                /* rotate mask for next bit */
                                    /* also used to burn time for clock */
      PORTC &= ~CLK;                /* send clock line low */
   }
}



int Input_Data(void)
{
   unsigned char Count;             /* gen purpose bit counter */
   unsigned int Mask = 0x8000;      /* mask to insert ‘1’ at bit position */
   unsigned int Result = 0x0000;    /* A/D result register */

   for(Count = 0x00; Count < 0x0D; Count++)  /* count 13 bits */
   {                                /* 12-bit result + 1 null bit */
      if(PORTC & DIN)               /* check if bit is high or low */
      {
         Result |= Mask;            /* bit high, set bit in result */
      }

      PORTC |= CLK;                 /* send clock line high */

      Mask >>= 0x01;                /* rotate mask for next bit */
                                    /* also used to burn time for clock */
      PORTC &= ~CLK;                /* send clock line low */
   }

   Result >>= 0x03;                 /* rotate bits into position */
   Result &= 0x0FFF;                /* mask out 12-bit result */

   return(Result);                  /* return result to caller */
}




void main(void)
{
   union DualAccess                    /* declare union to allow access to */
   {                                   /* variable as 8 or 16-bit */
      unsigned int By_16;              /* allows 16-bit access */

      struct Bytewise                  /* struct provides for 8-bit access */
      {
         unsigned char Lo;             /* LSB of variable */
         unsigned char Hi;             /* MSB of variable */
      } By_8;
   } ADresult;


TRISB = 0x00;
PORTB = 0x00;                       /* reset PortB outputs */
```

```
PORTC = 0x40;                       /* init PortC (A/D de-selected) */
TRISC = 0x10;                       /* config PortC */

PORTC |= ADCS;                      /* de-select A/D converter */

while(1)
{
   PORTC &= ~ADCS;                  /* select A/D converter */

   Output_Control((char)0x0F);      /* output control word to A/D converter */

   ADresult.By_16 = Input_Data();   /* read result from converter */

   PORTC |= ADCS;                   /* de-select A/D converter */

   PORTB = ADresult.By_8.Hi;        /* display A/D MSb’s */

   Wait_for_Press();                /* wait for key press/release */

   PORTB = ADresult.By_8.Lo;        /* display A/D LSb’s */

   Wait_for_Press();                /* wait for key press/release */

}
}
```

© 1999 Microchip Technology Inc.

APPENDIX I: EQUIVALENT SCHEMATIC



NOTES:

© 1999 Microchip Technology Inc.

NOTES:

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rfPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

DNV Certification, Inc.
USA
ANSI★RAB
QMS
ACCREDITED
DNV
DNV MSC
The Netherlands
Accredited by the RvA
ISO 9001 / QS-9000
REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral 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: 480-792-7456

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: 86-28-6766200 Fax: 86-28-6766599

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

Hong Kong

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: 65-334-8870 Fax: 65-334-8850

Taiwan

Microchip Technology Taiwan
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

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

01/18/02