CRYSTAL
A DIVISION OF CIRRUS LOGIC®

AN88

Application Note

Interfacing the CS5525/6/9 to the PIC16F84

By Keith Coffey

INTRODUCTION

This application note details the interface of Crystal Semiconductor's CS5525/6/9 Analog-to-Digital Converter (ADC) to the Microchip PIC16 microcontroller series. This note takes the reader through a simple example describing how to communicate with the ADC. All algorithms discussed are included in the **Appendix** at the end of this note.

ADC DIGITAL INTERFACE

The CS5525/6/9 interfaces to the PIC16F84 through either a three-wire or a four-wire interface. Figure 1 depicts the interface between the two devices. Though this software was written to interface to Port A (RA) on the PIC16F84 with a four-wire interface, the algorithms can be easily modified to work with the three-wire format.

The ADC's serial port consists of four control lines: $\overline{\text{CS}}$, SCLK, SDI, and SDO.

$\overline{\text{CS}}$, Chip Select, is the control line which enables access to the serial port.

SCLK, Serial Clock, is the bit-clock which controls the shifting of data to or from the ADC's serial port.

SDI, Serial Data In, is the data signal used to transfer data from the PIC16F84 to the ADC.

SDO, Serial Data Out, is the data signal used to transfer output data from the ADC to the PIC16F84.

SOFTWARE DESCRIPTION

This note presents algorithms to initialize the PIC16F84 and the CS5525/6/9, perform a self-offset calibration, modify the CS5525/6/9 gain register, and then acquire a conversion. Figure 2 depicts

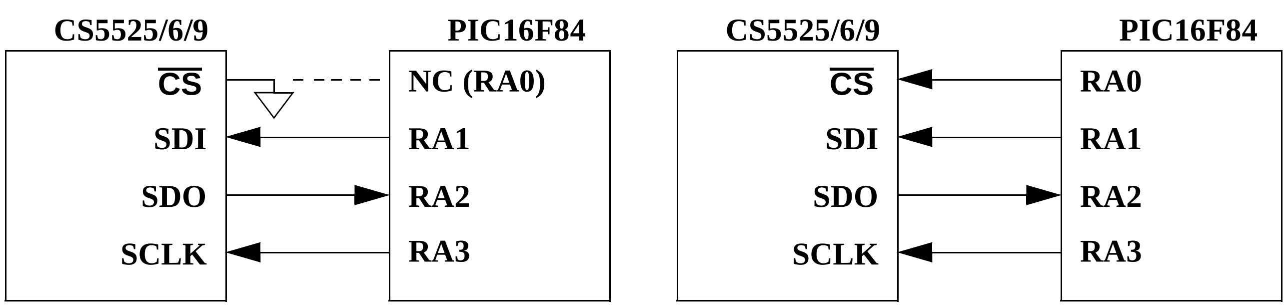


Figure 1. 3-Wire and 4-Wire Interfaces

Cirrus Logic, Inc.
Crystal Semiconductor Products Division
P.O. Box 17847, Austin, Texas 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.crystal.com

a block diagram overview. While reading this application note, please refer to the **Appendix** for the code listing.

Initialize

Initialize is a subroutine that configures Port A (RA) on the PIC16F84 and places the CS5525/6/9 in the command-state. First, RA's data direction is configured as depicted in Figure 1 (for more information on configuring ports refer to Microchip's PIC16F8X Data Sheet). After configuring the port, the controller enters a delay state to allow time for the CS5525/6/9's power-on-reset and oscillator to start-up (oscillator start-up time is typically 500 ms). The last step is to reinitialize the serial port on the ADC (reinitializing the serial port is unnecessary here, it was added for demonstration purposes only). This is implemented by sending the converter sixteen bytes of logic 1's followed by one final byte, with its LSB logic 0. Once sent, the sequence places the serial port of the ADC into the command-state, where it awaits a valid command.

After returning to *main*, the software demonstrates how to calibrate the converter's offset.

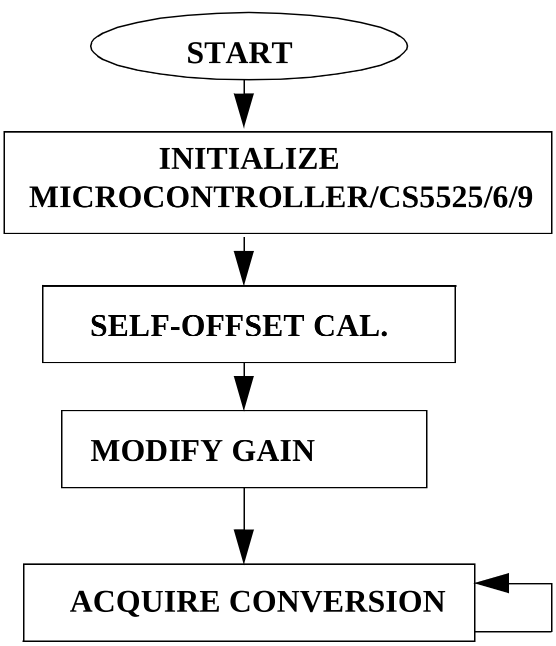


Figure 2. CS5525/6/9 Software Flowchart

Self-Offset Calibration

Calibrate is a subroutine that calibrates the converter's offset. *Calibrate* first sends 0x000001 (Hex) to the configuration register. This instructs the converter to perform a self-offset calibration. Then the Done Flag (DF) bit in the configuration register is polled until set. Once DF is set, it indicates that a valid calibration was performed. To minimize digital noise (while performing a calibration or a conversion), many system designers may find it advantageous to add a software delay equivalent to a conversion or calibration cycle before polling the DF bit.

Read/Write Gain Register

To modify the gain register the command-byte and data-byte variables are first initialized. This is accomplished by the MOVLW and MOVWF opcodes. The subroutine *write_register* uses these variables to set the contents of the gain register in the CS5525/6/9 to 0x800000 (HEX). To do this, *write_register* first asserts $\overline{CS}$ and then it calls *send_spi* four times (once for the command-byte and three additional times for the 24 bits of data). *Send_spi* is a subroutine used to 'bit-bang' a byte of information from the PIC16F84 to the CS5525/6/9. A byte is transferred one bit at a time, MSB (most significant bit) first, by placing an information bit on RA1 (SDI) and then pulsing RA3 (SCLK). This process is repeated eight times. Figure 3 depicts the timing diagram for the write-cycle in the CS5525/6/9's serial port. This algorithm demonstrates how to write to the gain register. It does not perform a gain calibration. To perform a gain calibration, follow the procedures outlined in the data sheet.

To verify if 0x800000 (HEX) was written to the gain register, *read_register* is called. It duplicates the read-cycle timing diagram depicted in Figure 4. *Read_register* first asserts $\overline{CS}$ and then calls *send_spi* once to transfer the command-byte to the CS5525/6/9. This places the converter into the data-state where it waits until data is read from its serial port. To receive the data, *read_register* calls *receive_spi* three times. *Receive_spi* is a subroutine used to ‘bit-bang’ a byte of information from the ADC to the PIC16F84. Similar to *send_spi*, *receive_spi* acquires this information one bit at a time MSB first. When the transfer is complete, the variables highbyte, midbyte, and lowbyte contain the CS5525/6/9’s 24-bit gain register.

Acquire Conversion

To acquire a conversion the subroutine *convert* is called. *Convert* sends the command-byte 0x0C to the converter. This instructs the converter to perform a single conversion. Then the Done Flag (DF) bit in the configuration register is polled. When set, DF indicates that a conversion was performed. Once DF is set, the controller reads the conversion data register to acquire the conversion.Figure 6 depicts how 16-bit and 20-bit conversion words are stored in the microcontroller.

An alternate method can be used to acquire a conversion. By setting the Port Flag bit (PF, the fifth bit in the configuration register), SDO’s function is modified to fall to logic 0 when a conversion is complete (refer to Figure 5). By tying SDO to the controller’s interrupt pin, conversions can be acquired via an interrupt service routine.

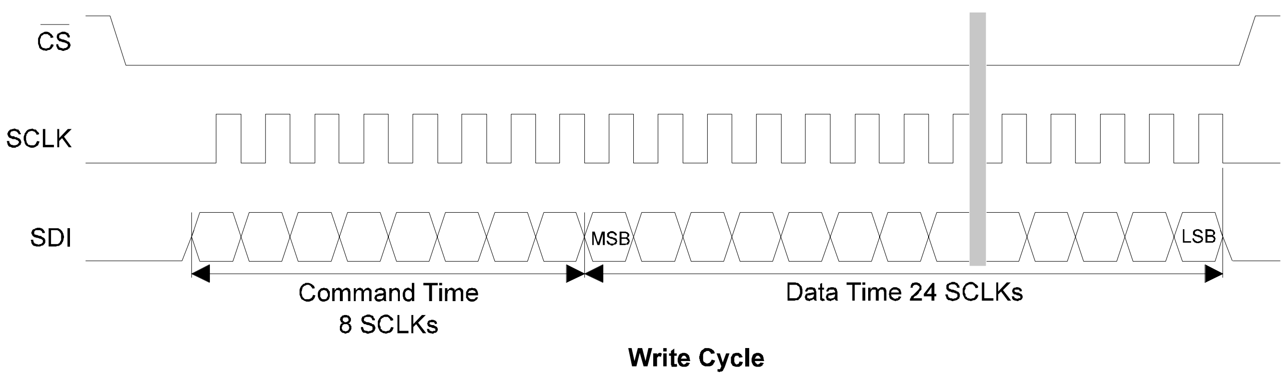


Figure 3. Write-Cycle Timing

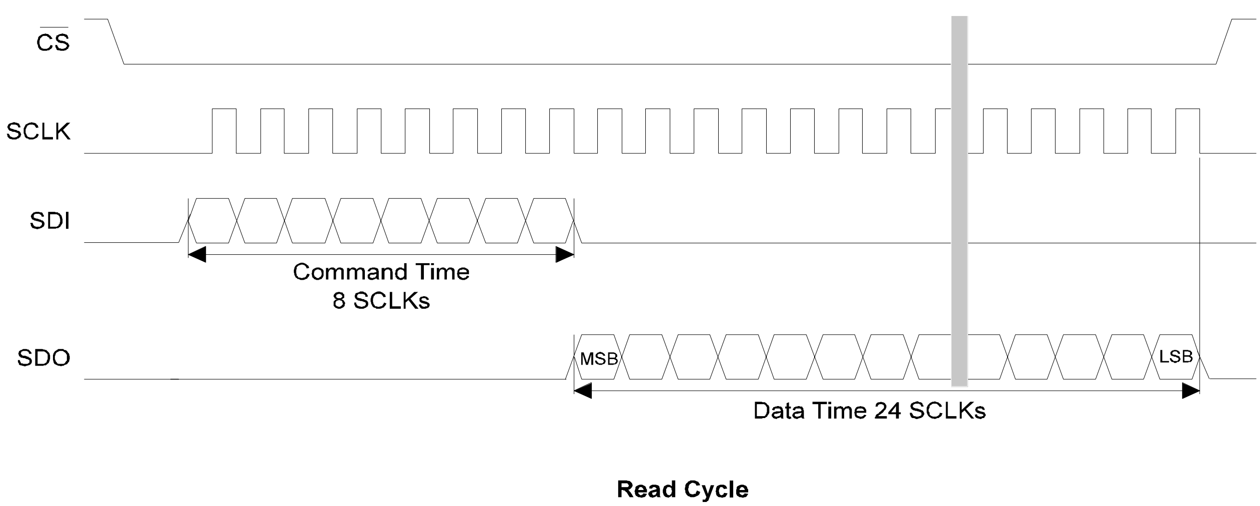
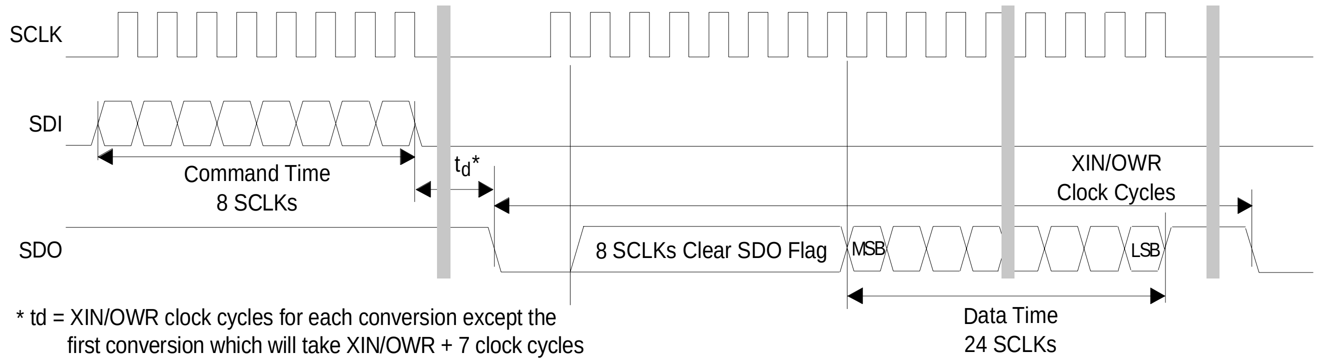


Figure 4. Read-Cycle Timing



Data SDO Continuous Conversion Read (PF bit = 1)

Figure 5. Conversion/Acquisition Cycle with the PF Bit Asserted

MSB **High-Byte**

D19	D18	D17	D16	D15	D14	D13	D12

Mid-Byte

D11	D10	D9	D8	D7	D6	D5	D4

Low-Byte

D3	D2	D1	D0	0	0	OD	OF

A) 20-Bit Conversion Data Word

MSB **High-Byte**

D15	D14	D13	D12	D11	D10	D9	D8

Mid-Byte

D7	D6	D5	D4	D3	D2	D1	D0

Low-Byte

1	1	1	1	0	0	OD	OF

B) 16-Bit Conversion Data Word

0- always zero, 1- always one,

OD - Oscillation Detect, OF - Overflow

Figure 6. Bit Representation/Storage in PIC16F84

MAXIMUM SCLK RATE

A machine cycle in the PIC16F84 consists 4 oscillator periods or 400 ns if the microcontroller's oscillator frequency is 10 MHz. Since the CS5525/6/9's maximum SCLK rate is 2MHz, additional no operation (NOP) delays may be necessary to reduce the transfer rate if the microcontroller system requires higher rate oscillators.

SERIAL PERIPHERAL INTERFACE

The Serial Peripheral Interface (SPI) developed for Microchip's controllers wasn't designed to be as flexible as the SPI port on Motorola's 68HC05. To get the Microchip's SPI port to function with the CS5525/6/9, the port needs to be initialized to idle high, and the CS5525/6/9's serial port needs to be reset anytime information is transmitted between the microcontroller and the converter.

DEVELOPMENT TOOL DESCRIPTION

The code in this application note was developed using *MPLAB*™, an integrated software development package from Microchip, Inc.

CONCLUSION

This application note presents an example of how to interface the CS5525/6/9 to the PIC16F84. It is divided into two main sections: hardware and software. The hardware section illustrates both a three-wire and a four-wire interface. The three-wire is *SPI*™ and *MICROWIRE*™ compatible. The software, developed with development tools from Microchip, Inc., illustrates how to initialize the converter and microcontroller, calibrate the converters offset, write to and read from the ADC's internal register, and acquire a conversion. The software is modularized and illustrates important subroutines, e.g. *write_register* and *read_register*. The software described in the note is included in the **Appendix** at the end of this document.

SPI™ is a trademark of Motorola.

MICROWIRE™ is a trademark of National Semiconductor.

MPLAB™ is a trademark of Microchip.

APPENDIX

PIC16F84 Microcode to Interface to the CS5525/6/9

```
;*********************************************************************************
;* File:          55261684.asm
;* Date:          November 15, 1996
;* Programmer:Keith Coffey
;* Revision:      0
;* Processor:     PIC16F84
;* Program entry point at routine "main".  The entry point is address 0x05.
;*********************************************************************************
;* Program is designed as an example to interface a PIC16F84 to a CS5525/6/9
;* ADC.  The program interfaces via a software SPI which controls the
;* serial communications, calibration, and conversion signals.  Other ADC's
;* (16-bit and 20-bit) in the product family can be used.
;*********************************************************************************
;******** Memory Map Equates
INDF          equu 0x00          ; Indirect Address Register
```

```
;*****************************************************************************************
;*      Program Code
;*****************************************************************************************
            processor           16C84               ; Set Processor Type
            org                 0x00                ; Reset Vector
            goto                Main                ; Start at Main

;*****************************************************************************************
;* Routine - Main
;* Input   - none
;* Output  - none
;* This is the entry point to the program.
;*****************************************************************************************
            org                 0x05
Main                                                ; Start from Reset Vector

;********  Initialize System and Perform SELF OFFSET Calibration
                CALL            initialize          ; Initialize the system
                CALL            calibrate           ; Calibrate the ADC Offset
;********  Write to the GAIN Register
                MOVLW           0x82                ; Prepare COMMANDBYTE
                MOVWF           COMMANDBYTE
                MOVLW           0x80                ; Prepare HIGHBYTE
                MOVWF           HIGHBYTE
                CLRF            MIDBYTE             ; Prepare MIDBYTE
                CLRF            LOWBYTE             ; Prepare LOWBYTE
                CALL            write_register      ; Write to Gain Register
;********  Read from the GAIN Register
                MOVLW           0x92                ; Prepare COMMANDBYTE
                MOVWF           COMMANDBYTE
                CALL            read_register       ; Read the Gain Register
;********  Perform Single Conversions
LOOP            CALL            convert             ; Convert Analog input
                goto            LOOP                ; Repeat Loop
;********  End MAIN
```

```
;*****************************************************************************************
;*      Subroutines
;*****************************************************************************************
;*****************************************************************************************
;* Routine  - initialize
;* Input      - none
;* Output   - none
;* This subroutine initializes port A for interfacing to the CS5525/6/9 ADC.
;* It provides a time delay for oscillator start-up/wake-up period.
;* A typical start-up time for a 32768 Hz crystal, due to high Q, is 500 ms.
;* Also 1003 XIN clock cycles are allotted for the ADC's power on reset.  The
;* total delay is 555 ms upon power-up (assume uC start-up time is zero).
;*****************************************************************************************
initialize      CLRF        PORTA           ; Initialize PORTA by setting output
                                            ; data latches.
                BSF         STATUS, RP0     ; Select Bank 1
                MOVLW       0x04            ; Value used to initialize direction
                MOVWF       TRISA           ; Set RA2 as inputs
                                            ; RA0, RA1, RA3, & RA4 as outputs

                BCF         STATUS, RP0     ; Select Bank 0
                BCF         PORTA,SDO       ; Clear SDO
                MOVLW       0x32            ; Load W with delay count
                CALL        delay           ; Delay, Power on Reset 1003 XIN
                MOVLW       0xFF            ; Load W with delay count
                CALL        delay           ; Delay, Oscillator start-up 158 ms
                CALL        delay           ; Delay, Oscillator start-up 158 ms
                CALL        delay           ; Delay, Oscillator start-up 158 ms
                CALL        delay           ; Delay, Oscillator start-up 158 ms
                MOVLW       0x0F            ; Reset Serial Port on ADC
                MOVWF       TEMP

                BCF         PORTA,CS        ; Clear CS
loop            MOVLW       0xFF            ; Load W with 0xFF
                CALL        send_spi        ; Send 15 0xFF through SPI
                DECFSZ      TEMP,1          ; Decrement the counter
                goto        loop            ; Repeat loop if counter not zero
                MOVLW       0xFE            ; Load W with last byte
                CALL        send_spi        ; Move 0xFE to SPDR
                BSF         PORTA,CS        ; Clear CS
                RETURN                      ; Exit subroutine
```

```
;*****************************************************************************************
;* Routine  - calibrate
;* Input      - none
;* Output   - none
;* This subroutine instructs the CS5525/6/9 to perform self-offset calibration.
;*****************************************************************************************
calibrate       MOVLW       0x84                ; set command byte for config write
                MOVWF       COMMANDBYTE         ; set COMMAND BYTE
                CLRF        HIGHBYTE            ; clear HIGHBYTE
                CLRF        MIDBYTE             ; clear MIDBYTE
                MOVLW       0x01                ; get ready for self offset cal
                MOVWF       LOWBYTE             ; set LOWBYTE
                CALL        write_register      ; Write to Config Register

                MOVLW       0x94                ; set command byte for config read
                MOVWF       COMMANDBYTE         ; set COMMAND BYTE
poll_done:      CALL        read_register       ; Poll done flag until cal complete
                BTFSS       LOWBYTE,3           ; repeat if flag not set
                goto        poll_done
                RETURN                          ; Exit subroutine


;*****************************************************************************************
;* Routine  - convert
;* Input      - none
;* Output   - Conversion results in memory locations HIGHBYTE, MIDBYTE and
;*              LOWBYTE.  This algorithm performs only single conversions.  If
;*              continuous conversions are needed the routine needs to be
;*              modified.  Port flag is zero.
;*                                    HIGHBYTE         MIDBYTE          LOWBYTE
;*                                   7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0   7 6 5 4 3 2  1 0
;* 16-bit results MSB                                      LSB 1 1 1 1 0 0 OD OF
;* 20-bit results MSB                                      LSB 0 0 OD OF
;* This subroutine initiates a single conversion.
;*****************************************************************************************
convert         MOVLW       0xC0                ; Set COMMANDBYTE for single CONV
                MOVWF       COMMANDBYTE
                BCF         PORTA,CS            ; Clear Chip Select
                CALL        send_spi            ; Transmit command out SPI
                MOVLW       0x94                ; Set command byte for config read
                MOVWF       COMMANDBYTE         ; Send COMMAND BYTE
done1        CALL           read_register       ; Poll done flag until CONV complete
                BTFSS       LOWBYTE,3           ; Repeat if Done Flag not Set
                goto        done1

                MOVLW       0x96                ; Set Byte to Read Conversion Reg.
                MOVWF       COMMANDBYTE         ; Store COMMAND BYTE
                CALL        read_register       ; Acquire the Conversion
                BSF         PORTA,CS            ; Set Chip Select
                RETURN                          ; Exit subroutine
```

```
;*****************************************************************************************
;* Routine  - write_register
;* Input      - COMMANDBYTE, HIGHBYTE, MIDBYTE, LOWBYTE
;* Output    - none
;*
;* This subroutine instructs the CS5525/6/9 to write to an internal register.
;*****************************************************************************************
write_register  BCF       PORTA,CS          ; Clear Chip Select
                MOVF      COMMANDBYTE,0     ; Load W with COMMANDBYTE
                CALL      send_spi          ; transfer byte
                MOVF      HIGHBYTE,0        ; Load W with HIGHBYTE
                CALL      send_spi          ; transfer byte
                MOVF      MIDBYTE,0         ; Load W with MIDBYTE
                CALL      send_spi          ; transfer byte
                MOVF      LOWBYTE,0         ; Load W with LOWBYTE
                CALL      send_spi          ; transfer byte
                BSF       PORTA,CS          ; Set Chip Select
                RETURN                      ; Exit Subroutine


;*****************************************************************************************
;* Routine  - read_register
;* Input      - COMMANDBYTE
;* Output    - HIGHBYTE, MIDBYTE, LOWBYTE
;* This subroutine reads an internal register of the ADC.
;*****************************************************************************************
read_register   BCF       PORTA,CS          ; Clear Chip Select
                MOVF      COMMANDBYTE,0     ; Load W with COMMANDBYTE
                CALL      send_spi          ; transfer byte
                CALL      receive_spi       ; receive byte
                MOVWF     HIGHBYTE          ; Move W to HIGHBYTE
                CALL      receive_spi       ; receive byte
                MOVWF     MIDBYTE           ; Move W to MIDBYTE
                CALL      receive_spi       ; receive byte
                MOVWF     LOWBYTE           ; Move W to LOWBYTE
                BSF       PORTA,CS          ; Set Chip Select
                RETURN                      ; Exit Subroutine
```

```
;****************************************************************************************
;* Routine  - send_spi
;* Input    - Byte to be transmitted is placed in W
;* Output   - None
;* This subroutine sends a byte to the ADC.
;****************************************************************************************
send_spi:     MOVWF      SPDR                ; Move W to SPDR
              MOVLW      0x08                ; Set COUNT to count to 8
              MOVWF      COUNT               ; to transmit byte out SPI
              BCF        PORTA,SCLK          ; Clear SCLK

wait0         ; Send Bit
              RLF        SPDR,1              ; Rotate SPDR, send MSB 1st
              BTFSC      STATUS,CARRY_BIT    ; If bit low skip next instruct.
              BSF        PORTA,SDI           ; Set SDI
              BTFSS      STATUS,CARRY_BIT    ; If bit high, skip next instruct.
              BCF        PORTA,SDI           ; Clear SDI

              BSF        PORTA,SCLK          ; Toggle Clock
              BCF        PORTA,SCLK
              DECFSZ     COUNT,1             ; Loop until byte is transmitted
              goto       wait0
              BCF        PORTA,SDI           ; Return Pin low
              RETURN                         ; Exit Subroutine


;****************************************************************************************
;* Routine  - receive_spi
;* Input    - none
;* Output   - Byte received is placed in W
;* This subroutine receives a byte from the ADC.
;****************************************************************************************
receive_spi:  MOVLW      0x08                ; Set COUNT to count to 8
              MOVWF      COUNT               ; to transmit byte out SPI
              BCF        PORTA,SCLK          ; Clear SCLK

wait1:        ; Receive bit
              BTFSC      PORTA,SDO           ; If bit low skip next instruct.
              BSF        STATUS,CARRY_BIT    ; Set SDI
              BTFSS      PORTA,SDO           ; If bit high, skip next instruct.
              BCF        STATUS,CARRY_BIT    ; Clear SDI
              RLF        SPDR,1              ; Rotate SPDR, Receive MSB 1st
              BSF        PORTA,SCLK          ; Toggle Clock
              BCF        PORTA,SCLK
              DECFSZ     COUNT,1             ; Loop until byte is transmitted
              goto       wait1

              MOVF       SPDR,0              ; Put byte attained in W
              RETURN                         ; Exit Subroutine
```

```
;*************************************************************************************
;* Routine  - delay
;* Input      - Count in register A
;* Output   - none
;* This subroutine delays by using count from register W.  The PIC16F84
;* development board uses a 10 MHz clock (E = 2.5 MHz), thus each cycle is
;* 400 nS.  This  delay is approximately equivalent to
;* (400ns)*(1545)*(count value), (a count of 720 provides a 445ms delay).
;*************************************************************************************
delay       MOVWFCOUNT                     ; Put the delay count into COUNT
outlp       CLRF        TEMP               ; TEMP used as inner loop count
innlp       NOP                            ; 1 cycle
            NOP                             ; 1 cycle
            NOP                            ; 1 cycle
            NOP                            ; 1 cycle
            DECFSZ      TEMP,1             ; FF-FE, FE-FD, ....1-0  256 loops
                                           ; 10 cycles*256*500ns=1.28 ms
            goto        innlp              ; If count not done repeat loop
            DECFSZ      COUNT,1            ; Countdown the accumulator
            goto        outlp              ; 2569 cycles*500ns*A
            RETURN                         ; Exit subroutine


;*************************************************************************************
;*      Interrupt Vectors
;*************************************************************************************
NOT_USED    RETFIE
            ORG         0x04               ; Originate Interrupt Vector here
            goto        NOT_USED           ; No Interrupts Enabled
        end                                ; End Program Listing
```

• Notes •

SMART Analog™