AN656

In-Circuit Serial Programming of Calibration Parameters Using a PICmicro™ Microcontroller

Author: John Day
Microchip Technology Inc.

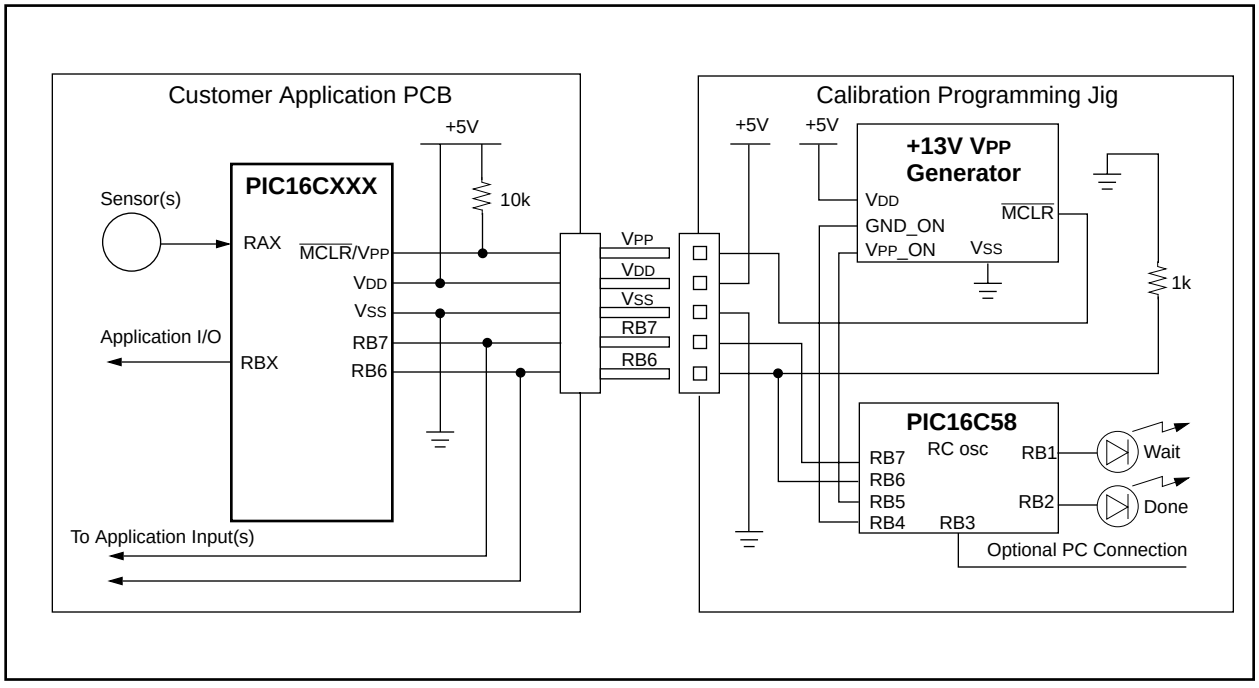
INTRODUCTION

Many embedded control applications, where sensor offsets, slopes and configuration information are measured and stored, require a calibration step. Traditionally, potentiometers or serial EEPROM devices are used to set up and store this calibration information. This application note will show how to construct a programming jig that will receive calibration parameters from the application midrange PICmicro and program this information into the application baseline PICmicro using the In-Circuit Programming protocol. This method uses the PIC16CXXX In Circuit Serial Programming algorithm of the 14-bit core microcontrollers.

PROGRAMMING FIXTURE

A programming fixture is needed to assist with the self programming operation. This is typically a small re-usable module that plugs into the application PCB being calibrated. Only 5 pin connections are needed and this programming fixture can draw its power from the application PCB to simplify the connections.

FIGURE 1:



Electrical Interface

There are a total of five electrical connections needed between the application PIC16CXXX microcontroller and the programming jig:

- **$\overline{\text{MCLR}}$/VPP** - High voltage pin used to place application PIC16CXX into programming mode
- **VDD** - +5 volt power supply connection to the application PIC16CXXX
- **VSS** - Ground power supply connection to the application PIC16CXXX
- **RB6** - PORTB, bit6 connection to application PIC16CXX used to clock programming data
- **RB7** - PORTB, bit7 connection to application PIC16CXX used to send programming data

This programming jig is intended to grab power from the application power supply through the VDD connection. The programming jig will require 100 mA of peak current during programming. The application will need to set RB6 and RB7 as inputs, which means external devices cannot drive these lines. The calibration data will be sent to the programming jig by the application PIC16CXXX through RB6 and RB7. The programming jig will later use these lines to clock the calibration data into the application PIC16CXXX.

Programming Issues

The PIC16CXXX Programming specification suggests verification of program memory Specification at both Maximum and Minimum VDD for each device. This is done to ensure proper programming margins and to detect (and reject) any improperly programmed devices. All production quality programmers vary VDD from VDD min to VDD max after programming and verify the device under each of these conditions.

Since both the application voltage and it's tolerances are known, it is not necessary to verify the PIC16CXXX calibration parameters at the device VDD Max and VDD Min. It is only necessary to verify at the application power supply Max and Min voltages. This application note shows the nominal (+5V) verification routine and hardware. If the power supply is a regulated +5V, this is adequate and no additional hardware or software is needed. If the application power supply is not regulated (such as a battery powered or poorly regulated system) it is important to complete a VDD min and VDD Max verification cycle following the +5V verification cycle. See programming specifications for more details on VDD verification procedures.

- PIC16C5X Programming Specifications - DS30190
- PIC16C55X Programming Specifications - DS30261
- PIC16C6X/7X/9XX Programming Specifications - DS30228
- PIC16C84 Programming Specifications - DS30189

Note:	The designer must consider environmental conditions, voltage ranges, and aging issues when determing VDD min/max verification levels. Please refer to the programming specification for the application device.

The calibration programming and initial verification MUST occur at +5V. If the application is intended to run at lower (or higher voltage), a second verification pass must be added where those voltages are applied to VDD and the device is verified.

Communication format (Application Microcontroller to Programming Jig)

Unused program memory, in the application PIC16CXXX, is left unprogrammed as all 1s; therefore the unprogrammed program memory for the calibration look-up table would contain 3FFF (hex). This is interpreted as a "ADDLW FF". The application microcontroller simply needs one "RETLW FF" instruction at the end of the space allocated in program memory for the calibration parameter look-up table. When the application microcontroller is powered up, it will receive a "FFh" for each calibration parameter that is looked up; therefore, it can detect that it is uncalibrated and jump to the calibration code.

Once the calibration constants are calculated by the application PICmicro, they need to be communicated to the (PIC16C58A based) programming jig. This is accomplished through the RB6 and RB7 lines. The format is a simple synchronous clock and data format as shown in Figure 2.

A pulldown on the clock line is used to hold it low. The application microcontroller needs to send the high and low bytes of the target start address of the calibration constants to the calibration jig. Next, the data bytes are sent followed by a checksum of the entire data transfer as shown in Figure 3.

Once the data transfer is complete, the checksum is verified by the programming jig and the data printed at 9600 baud, 8-bits, no parity, 1 stop bit through RB3. A connection to this pin is optional. Next the programming jig applies +13V, programs and verifies the application PIC16CXXX calibration parameters.

FIGURE 2:

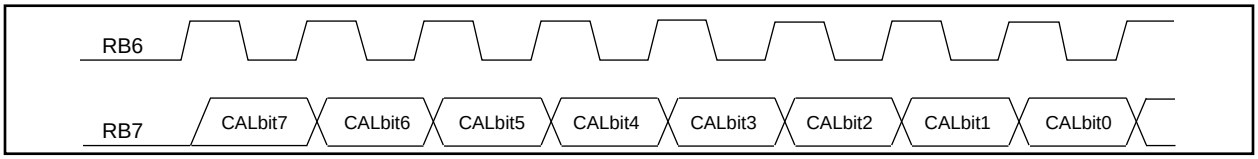
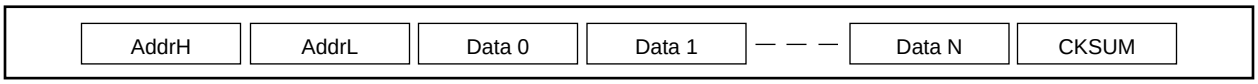


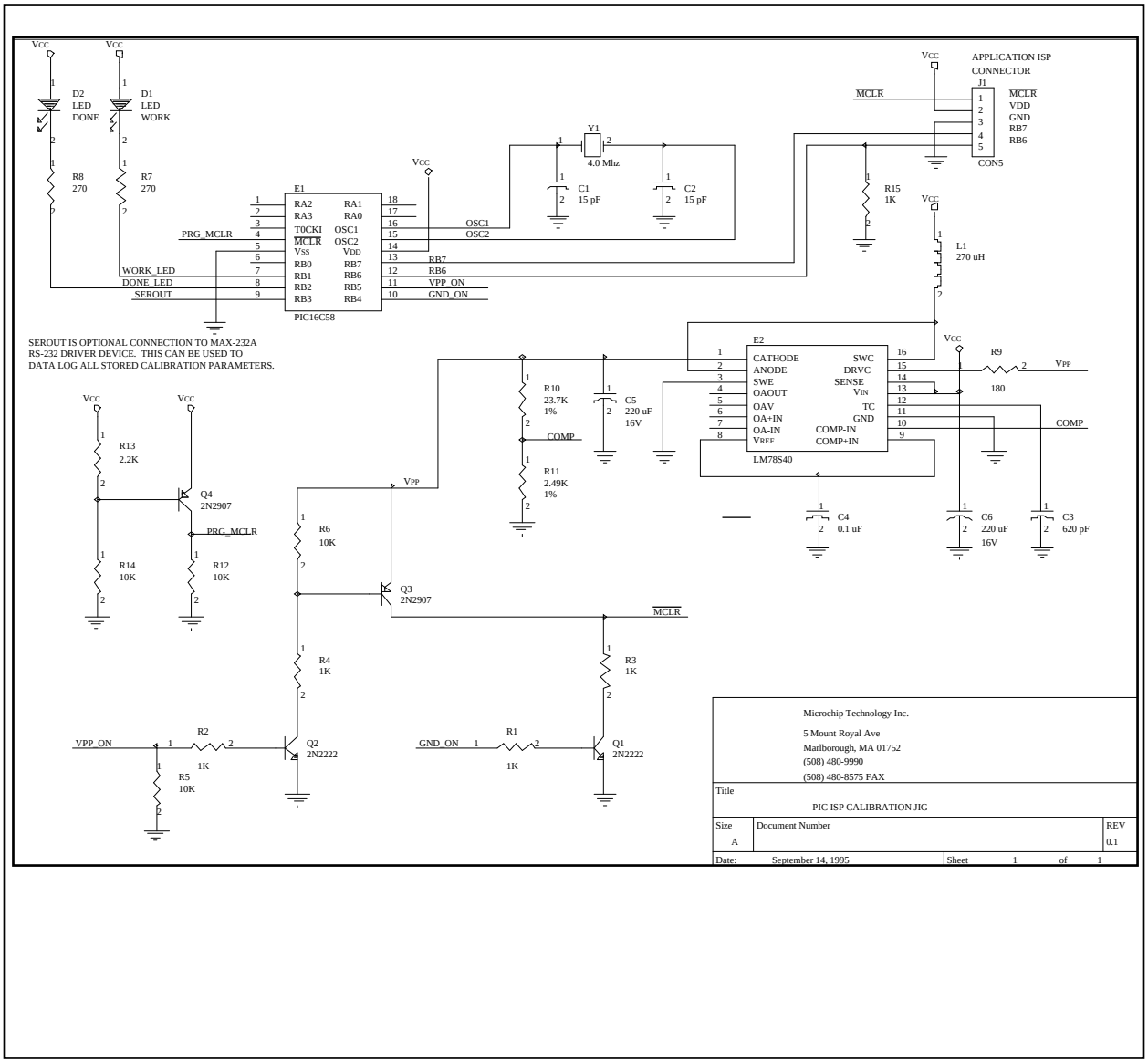
FIGURE 3:



LED Operation

When the programming jig is waiting for communication from the application PICmicro, both LEDs are OFF. Once a valid data stream is received (with at least one calibration byte and a correct checksum) the WORK LED is lit while the calibration parameters are printed through the optional RB3 port. Next, the DONE LED is lit to indicate that these parameters are being programmed and verified by the programming jig. Once the programming is finished, the WORK LED is extinguished and the DONE LED remains lit. If any parameters fail programming, the DONE LED is extinguished; therefore both LEDs would remain off.

FIGURE 4: ISP CALIBRATION JIG PROGRAMMER SCHEMATIC



Code Protection

Selection of the code protection configuration bit on PIC16CXXX microcontrollers prevents further programming of the program memory array. This would prevent writing self calibration parameters if the device is code protected prior to calibration. There are two ways to address this issue:

1. Do not code protect the device when programming it with the programmer. Add additional code (See the PIC16C6X/7X programming Spec) to the `ISPPRGM.ASM` to program the code protection bit after complete verification of the calibration parameters
2. Only code protect 1/2 or 3/4 of the program memory with the programmer. Place the calibration constants into the unprotected part of program memory.

Software Routines

There are two source code files needed for this application note:

1. `ISPTEST.ASM` (Appendix A) Contains the source code for the application PIC16CXXX, sets up the calibration look-up table and implements the communication protocol to the programming jig.

2. `ISPPRGM.ASM` (Appendix B) Source code for a PIC16C58A to implement the programming jig. This waits for and receives the calibration parameters from the application PIC16CXXX, places it into programming mode and programs/verifies each calibration word.

CONCLUSION

Typically, calibration information about a system is stored in EEPROM. For calibration data that does not change over time, the In-circuit Serial Programming capability of the PIC16CXXX devices provide a simple, cost effective solution to an external EEPROM. This method not only decreases the cost of a design, but also reduces the complexity and possible failure points of the application.

TABLE 1: PARTS LIST FOR PIC16CXXX ISP CALIBRATION JIG

Bill of Material			
Item	**Quantity**	**Reference**	**Part**
1	2	C1,C2	15 pF
2	1	C3	620 pF
3	1	C4	0.1 mF
4	2	C5,C6	220 mF
5	2	D1,D2	LED
6	1	E1	PIC16C58
7	1	E2	LM78S40
8	1	J1	CON5
9	1	L1	270 mH
10	2	Q1,Q2	2N2222
11	2	Q3,Q4	2N2907
12	5	R1,R2,R3,R4,R15	1k
13	4	R5,R6,R12,R14	10k
14	2	R7,R8	270
15	1	R9	180
16	1	R10	23.7k
17	1	R11	2.49k
18	1	R13	2.2k
19	1	Y1	4.0 MHz

APPENDIX A: ISPPRGM.ASM

```
MPASM 01.40.01 Intermediate   ISPPRGM.ASM   3-31-1997  10:57:03         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001 ; Filename: ISPPRGM.ASM
                     00002 ; ********************************************
                     00003 ; * Author:   John Day                       *
                     00004 ; *           Sr. Field Applications Engineer  *
                     00005 ; *           Microchip Technology             *
                     00006 ; * Revision: 1.0                            *
                     00007 ; * Date      August 25, 1995                *
                     00008 ; * Part:     PIC16C58                       *
                     00009 ; * Compiled using MPASM V1.40               *
                     00010 ; ********************************************
                     00011 ; * Include files:                           *
                     00012 ; *           P16C5X.ASM                     *
                     00013 ; ********************************************
                     00014 ; * Fuses:    OSC:  XT (4.0 Mhz xtal)        *
                     00015 ; *           WDT:  OFF                      *
                     00016 ; *            CP:  OFF                      *
                     00017
                     ;*******************************************************************************
                     00018 ; This program is intended to be used as a self programmer
                     00019 ; to store calibration constants into a lookup table
                     00020 ; within the main system processor.  A 4 Mhz crystal
                     00021 ; is needed and an optional 9600 baud seiral port will
                     00022 ; display the parameters to be programmed.
                     00023 ;
                     ;*******************************************************************************
                     00024 ; * Program Memory:                                                      *
                     00025 ; *    Words - communication with test jig                              *
                     00026 ; *    17 Words - calibration look-up table (16 bytes of data)          *
                     00027 ; *    13 Words - Test Code to generate Calibration Constants           *
                     00028 ; * RAM memory:                                                         *
                     00029 ; *    64 Bytes - Store up to 64 bytes of calibration constant          *
                     00030 ; *     9 Bytes - Store 9 bytes of temp variables (reused)              *
                     00031 ;
                     ;*****************************************************************************
                     00032
                     00033     list p=16C58A
                     00034     include <p16C5x.inc>
                     00001         LIST
                     00002 ; P16C5X.INC Standard Hdr File, Version 3.30 Microchip Technology, Inc.
                     00224         LIST
0FFF 0FF9            00035     __CONFIG _CP_OFF&_WDT_OFF&_XT_OSC
                     00036
                     00037 ; ***********************************
                     00038 ; * Port A (RA0-RA4) bit definitions *
                     00039 ; ***********************************
                     00040 ; No PORT A pins are used in this design
                     00041
                     00042 ; ***********************************
                     00043 ; * Port B (RB0-RB7) bit definitions *
                     00044 ; ***********************************
  00000006           00045 ISPCLOCK      EQU 6   ; Clock line for ISP and parameter comm
  00000007           00046 ISPDATA       EQU 7   ; Data line for ISP and parameter comm
  00000005           00047 VPPON         EQU 5   ; Apply +13V VPP voltage to MCLR (test mode)
  00000004           00048 GNDON         EQU 4   ; Apply +0V (gnd) voltage to MCLR (reset)
  00000003           00049 SEROUT        EQU 3   ; Optional RS-232 TX output (needs 12V driver)
  00000002           00050 DONELED       EQU 2   ; Turns on LED when done sucessfully program
  00000001           00051 WORKLED       EQU 1   ; On during programming, off when done
                     00052                       ; RB0 is not used in this design
                     00053
```

```
                00054 ; ***********************************************
                00055 ; * RAM register definition:                     *
                00056 ; *  07h - 0Fh - used for internal counters, vars *
                00057 ; *  10h - 7Fh - 64 bytes for cal param storage   *
                00058 ; ***********************************************
                00059 ; ***
                00060 ; *** The following VARS are used during ISP programming:
                00061 ; ***
00000007        00062 HIADDR         EQU 07h  ; High address of CAL params to be stored
00000008        00063 LOADDR         EQU 08h  ; Low address of CAL params to be stored
00000007        00064 HIDATA         EQU 07h  ; High byte of data to be sent via ISP
00000008        00065 LODATA         EQU 08h  ; Low byte of data to be sent via ISP
00000009        00066 HIBYTE         EQU 09h  ; High byte of data received via ISP
0000000A        00067 LOBYTE         EQU 0Ah  ; Low byte of data received via ISP
0000000B        00068 PULSECNT       EQU 0Bh  ; Number of times PIC has been pulse programmed
0000000C        00069 TEMPCOUNT      EQU 0Ch  ; TEMP var used in counters
0000000D        00070 TEMP           EQU 0Dh  ; TEMP var used throughout program
                00071 ; ***
                00072 ; *** The following VARS are used to receive and store CAL params:
                00073 ; ***
00000007        00074 COUNT          EQU 07h  ; Counter var used to receive cal params
00000008        00075 TEMP1          EQU 08h  ; TEMP var used for RS-232 comm
00000009        00076 DATAREG        EQU 09h  ; Data register used for RS-232 comm
0000000A        00077 CSUMTOTAL      EQU 0Ah  ; Running total of checksum (addr + data)
0000000B        00078 TIMEHIGH       EQU 0Bh  ; Count how long CLOCK line is high
0000000C        00079 TIMELOW        EQU 0Ch  ; Count how long CLOCK line is low
0000000E        00080 ADDRPTR        EQU 0Eh  ; Pointer to next byte of CAL storage
0000000F        00081 BYTECOUNT      EQU 0Fh  ; Number of CAL bytes received
                00082
                00083 ; ************************************
                00084 ; * Various constants used in program *
                00085 ; ************************************
00000001        00086 DATISPOUT      EQU b'00000001'  ; tris settings for ISP data out
00000081        00087 DATISPIN       EQU b'10000001'  ; tris settings for ISP data in
00000006        00088 CMDISPCNT      EQU 6            ; Number of bits for ISP command
00000010        00089 STARTCALBYTE   EQU 10h          ; Address in RAM where CAL byte data stored
00000007        00090 VFYYES         EQU PA2          ; Flag bit enables verification (STATUS)
00000006        00091 CMDISPINCRADDR EQU b'00000110'  ; ISP Pattern to increment address
00000008        00092 CMDISPPGMSTART EQU b'00001000'  ; ISP Pattern to start programming
0000000E        00093 CMDISPPGMEND   EQU b'00001110'  ; ISP Pattern to end programming
00000002        00094 CMDISPLOAD     EQU b'00000010'  ; ISP Pattern to load data for program
00000004        00095 CMDISPREAD     EQU b'00000100'  ; ISP Pattern to read data for verify
00000034        00096 UPPER6BITS     EQU 034h         ; Upper 6 bits for retlw instruction
                00097
                00098 ; ************************************
                00099 ; * delaybit macro                    *
                00100 ; * Delays for 104 uS (at 4 Mhz clock)*
                00101 ; * for 9600 baud communications      *
                00102 ; *     RAM used:       COUNT        *
                00103 ; ************************************
                00104 delaybit macro
                00105     local dlylabels
                00106 ; 9600 baud, 8 bit, no parity, 104 us per bit, 52 uS per half bit
                00107 ; (8) shift/usage + (2) setup + (1) nop + (3 * 31) literal = (104) 4Mhz
                00108     movlw   .31             ; place 31 decimal literal into count
                00109     movwf   COUNT           ; Initialize COUNT with loop count
                00110     nop                     ; Add one cycle delay
                00111 dlylabels
                00112     decfsz  COUNT,F         ; Decrement count until done
                00113     goto    dlylabels       ; Not done delaying - go back!
                00114     ENDM                    ; Done with Macro
                00115
                00116 ; ***********************************************
                00117 ; * addrtofsr macro                            *
                00118 ; * Converts logical, continuous address 10h-4Fh *
                00119 ; * to FSR address as follows for access to (4)  *
```

```
                00120 ; * banks of file registers in PIC16C58:         *
                00121 ; *       Logical Address       FSR Value       *
                00122 ; *       10h-1Fh               10h-1Fh         *
                00123 ; *       20h-2Fh               30h-3Fh         *
                00124 ; *       30h-3Fh               50h-5Fh         *
                00125 ; *       40h-4Fh               70h-7Fh         *
                00126 ; *     Variable Passed: Logical Address        *
                00127 ; *     RAM used:        FSR                    *
                00128 ; *                      W                      *
                00129 ; ************************************************
                00130 addrtofsr macro TESTADDR
                00131     movlw   STARTCALBYTE        ; Place base address into W
                00132     subwf   TESTADDR,w          ; Offset by STARTCALBYTE
                00133     movwf   FSR                 ; Place into FSR
                00134     btfsc   FSR,5               ; Shift bits 4,5 to 5,6
                00135     bsf     FSR,6
                00136     bcf     FSR,5
                00137     btfsc   FSR,4
                00138     bsf     FSR,5
                00139     bsf     FSR,4
                00140     endm
                00141
                00142
                00143 ; ************************************
                00144 ; * The PC starts at the END of memory *
                00145 ; ************************************
07FF            00146     ORG     7FFh
Message[306]: Crossing page boundary -- ensure page bits are set.
07FF 0A00       00147     goto    start
                00148
                00149 ; ************************************
                00150 ; * Start of CAL param read routine    *
                00151 ; ************************************
0000            00152     ORG     0h
0000            00153 start
0000 0C0A       00154     movlw   b'00001010'   ; Serial OFF, LEDS OFF, VPP OFF
0001 0026       00155     movwf   PORTB         ; Place "0" into port b latch register
0002 0CC1       00156     movlw   b'11000001'   ; RB7;:RB6, RB0 set to inputs
0003 0006       00157     tris    PORTB         ; Move to tris registers
0004 0040       00158     clrw                  ; Place 0 into W
0005 0065       00159     clrf    PORTA         ; Place all ZERO into latch
0006 0005       00160     tris    PORTA         ; Make all pins outputs to be safe..
0007 0586       00161     bsf     PORTB,GNDON   ; TEST ONLY-RESET PIC-NOT NEEDED IN REAL DESIGN!
0008            00162 clearram
0008 0C10       00163     movlw   010h          ; Place start of buffer into W
0009 0027       00164     movwf   COUNT         ; Use count for RAM pointer
000A            00165 loopclrram
                00166     addrtofsr COUNT       ; Set up FSR
000A 0C10           M     movlw   STARTCALBYTE  ; Place base address into W
000B 0087           M     subwf   COUNT,w       ; Offset by STARTCALBYTE
000C 0024           M     movwf   FSR           ; Place into FSR
000D 06A4           M     btfsc   FSR,5         ; Shift bits 4,5 to 5,6
000E 05C4           M     bsf     FSR,6
000F 04A4           M     bcf     FSR,5
0010 0684           M     btfsc   FSR,4
0011 05A4           M     bsf     FSR,5
0012 0584           M     bsf     FSR,4
0013 0060       00167     clrf    INDF           ; Clear buffer value
0014 02A7       00168     incf    COUNT,F        ; Move to next reg
0015 0C50       00169     movlw   050h           ; Move end of buffer addr to W
0016 0087       00170     subwf   COUNT,W        ; Check if at last MEM
0017 0743       00171     btfss   STATUS,Z       ; Skip when at end of counter
0018 0A0A       00172     goto    loopclrram     ; go back to next location
0019 0486       00173     bcf     PORTB,GNDON    ; TEST ONLY-LET IT GO-NOT NEEDED IN REAL DESIGN!
001A            00174 calget
001A 006A       00175     clrf    CSUMTOTAL      ; Clear checksum total byte
```

```
001B 0069      00176     clrf    DATAREG             ; Clear out data receive register
001C 0C10      00177     movlw   STARTCALBYTE        ; Place RAM start address of first cal byte
001D 002E      00178     movwf   ADDRPTR             ; Place this into ADDRPTR
001E           00179 waitclockpulse
001E 07C6      00180     btfss   PORTB,ISPCLOCK      ; Wait for CLOCK high pulse - skip when high
001F 0A1E      00181     goto    waitclockpulse      ; CLOCK is low - go back and wait!
0020           00182 loopcal
0020 0C08      00183     movlw   .8                  ; Place 8 into W (8 bits/byte)
0021 0027      00184     movwf   COUNT               ; set up counter register to count bits
0022           00185 loopsendcal
0022 006B      00186     clrf    TIMEHIGH            ; Clear timeout counter for high pulse
0023 006C      00187     clrf    TIMELOW             ; Clear timeout counter for low pulse
0024           00188 waitclkhi
0024 06C6      00189     btfsc   PORTB,ISPCLOCK      ; Wait for CLOCK high - skip if it is low
0025 0A29      00190     goto    waitclklo           ; Jump to wait for CLOCK low state
0026 02EB      00191     decfsz  TIMEHIGH,F          ; Decrement counter - skip if timeout
0027 0A24      00192     goto    waitclkhi           ; Jump back and wait for CLOCK high again
0028 0A47      00193     goto    timeout             ; Timed out waiting for high - check data!
0029           00194 waitclklo
0029 07C6      00195     btfss   PORTB,ISPCLOCK      ; Wait for CLOCK low - skip if it is high
002A 0A2E      00196     goto    clockok             ; Got a high to low pulse - jump to clockok
002B 02EC      00197     decfsz  TIMELOW,F           ; Decrement counter - skip if timeout
002C 0A29      00198     goto    waitclklo           ; Jump back and wait for CLOCK low again
002D 0A47      00199     goto    timeout             ; Timed out waiting for low - check data!
002E           00200 clockok
002E 0C08      00201     movlw   .8                  ; Place initial count value into W
002F 0087      00202     subwf   COUNT,W             ; Subtract from count, place into W
0030 0743      00203     btfss   STATUS,Z            ; Skip if we are at count 8 (first value)
0031 0A34      00204     goto    skipcsumadd         ; Skip checksum add if any other count value
0032 0209      00205     movf    DATAREG,W           ; Place last byte received into W
0033 01EA      00206     addwf   CSUMTOTAL,F         ; Add to checksum
0034           00207 skipcsumadd
0034 0503      00208     bsf     STATUS,C            ; Assume data bit is high
0035 07E6      00209     btfss   PORTB,ISPDATA       ; Skip if the data bit was high
0036 0403      00210     bcf     STATUS,C            ; Set data bit to low
0037 0369      00211     rlf     DATAREG,F           ; Rotate next bit into DATAREG
0038 02E7      00212     decfsz  COUNT,F             ; Skip after 8 bits
0039 0A22      00213     goto    loopsendcal         ; Jump back and send next bit
               00214     addrtofsr ADDRPTR           ; Convert pointer address to FSR
003A 0C10          M     movlw   STARTCALBYTE        ; Place base address into W
003B 008E          M     subwf   ADDRPTR,w           ; Offset by STARTCALBYTE
003C 0024          M     movwf   FSR                 ; Place into FSR
003D 06A4          M     btfsc   FSR,5               ; Shift bits 4,5 to 5,6
003E 05C4          M     bsf     FSR,6
003F 04A4          M     bcf     FSR,5
0040 0684          M     btfsc   FSR,4
0041 05A4          M     bsf     FSR,5
0042 0584          M     bsf     FSR,4
0043 0209      00215     movf    DATAREG,W           ; Place received byte into W
0044 0020      00216     movwf   INDF                ; Move recv'd byte into CAL buffer location
0045 02AE      00217     incf    ADDRPTR,F           ; Move to the next cal byte
0046 0A20      00218     goto    loopcal             ; Go back for next byte
0047           00219 timeout
0047 0C14      00220     movlw   STARTCALBYTE+4      ; check if we received (4) params
0048 008E      00221     subwf   ADDRPTR,W           ; Move current address pointer to W
0049 0703      00222     btfss   STATUS,C            ; Skip if we have at least (4)
004A 0A93      00223     goto    sendnoise           ; not enough params - print and RESET!
004B 0200      00224     movf    INDF,W              ; Move received checksum into W
004C 00AA      00225     subwf   CSUMTOTAL,F    ; Subtract received Checksum from calc'd checksum
004D 0743      00226     btfss   STATUS,Z            ; Skip if CSUM OK
004E 0A9F      00227     goto    sendcsumbad         ; Checksum bad - print and RESET!
004F           00228 csumok
004F 0426      00229     bcf     PORTB,WORKLED       ; Turn on WORK LED
0050 0C10      00230     movlw   STARTCALBYTE        ; Place start pointer into W
0051 008E      00231     subwf   ADDRPTR,W           ; Subtract from current address
0052 002F      00232     movwf   BYTECOUNT           ; Place into number of bytes into BYTECOUNT
```

```
0053 002B      00233     movwf   TIMEHIGH           ; TEMP store into timehigh reg
0054 0C10      00234     movlw   STARTCALBYTE       ; Place start address into W
0055 002E      00235     movwf   ADDRPTR            ; Set up address pointer
0056           00236 loopprintnums
               00237     addrtofsr ADDRPTR          ; Set up FSR
0056 0C10          M     movlw   STARTCALBYTE       ; Place base address into W
0057 008E          M     subwf   ADDRPTR,w          ; Offset by STARTCALBYTE
0058 0024          M     movwf   FSR                ; Place into FSR
0059 06A4          M     btfsc   FSR,5              ; Shift bits 4,5 to 5,6
005A 05C4          M     bsf     FSR,6
005B 04A4          M     bcf     FSR,5
005C 0684          M     btfsc   FSR,4
005D 05A4          M     bsf     FSR,5
005E 0584          M     bsf     FSR,4
005F 0380      00238     swapf   INDF,W             ; Place received char into W
0060 0E0F      00239     andlw   0Fh                ; Strip off upper digits
0061 002D      00240     movwf   TEMP               ; Place into TEMP
0062 0C0A      00241     movlw   .10                ; Place .10 into W
0063 00AD      00242     subwf   TEMP,F             ; Subtract 10 from TEMP
0064 0603      00243     btfsc   STATUS,C           ; Skip if TEMP is less than 9
0065 0A6D      00244     goto    printhiletter      ; Greater than 9 - print letter instead
0066           00245 printhinumber
0066 0380      00246     swapf   INDF,W             ; Place received char into W
0067 0E0F      00247     andlw  0Fh                 ; Strip off upper digits
0068 002D      00248     movwf   TEMP               ; Place into TEMP
0069 0C30      00249     movlw   '0'                ; Place ASCII '0' into W
006A 01CD      00250     addwf   TEMP,w             ; Add to TEMP, place into W
006B 09AE      00251     call    putchar            ; Send out char
006C 0A73      00252     goto    printlo            ; Jump to print next char
006D           00253 printhiletter
006D 0380      00254     swapf   INDF,W             ; Place received char into W
006E 0E0F      00255     andlw  0Fh                 ; Strip off upper digits
006F 002D      00256     movwf   TEMP               ; Place into TEMP
0070 0C37      00257     movlw   'A'-.10            ; Place ASCII 'A' into W
0071 01CD      00258     addwf   TEMP,w             ; Add to TEMP, place into W
0072 09AE      00259     call    putchar            ; send out char
0073           00260 printlo
0073 0200      00261     movf    INDF,W             ; Place received char into W
0074 0E0F      00262     andlw   0Fh                ; Strip off upper digits
0075 002D      00263     movwf   TEMP               ; Place into TEMP
0076 0C0A      00264     movlw   .10                ; Place .10 into W
0077 00AD      00265     subwf   TEMP,F             ; Subtract 10 from TEMP
0078 0603      00266     btfsc   STATUS,C           ; Skip if TEMP is less than 9
0079 0A81      00267     goto    printloletter      ; Greater than 9 - print letter instead
007A           00268 printlonumber
007A 0200      00269     movf    INDF,W             ; Place received char into W
007B 0E0F      00270     andlw   0Fh                ; Strip off upper digits
007C 002D      00271     movwf   TEMP               ; Place into TEMP
007D 0C30      00272     movlw   '0'                ; Place ASCII '0' into W
007E 01CD      00273     addwf   TEMP,w             ; Add to TEMP, place into W
007F 09AE      00274     call    putchar            ; send out char
0080 0A87      00275     goto    printnext          ; jump to print next char
0081           00276 printloletter
0081 0200      00277     movf    INDF,W             ; Place received char into W
0082 0E0F      00278     andlw   0Fh                ; Strip off upper digits
0083 002D      00279     movwf   TEMP               ; Place into TEMP
0084 0C37      00280     movlw   'A'-.10            ; Place ASCII 'A' into W
0085 01CD      00281     addwf   TEMP,w             ; Add to TEMP, place into W
0086 09AE      00282     call    putchar            ; send out char
0087           00283 printnext
0087 0C7C      00284     movlw   '|'                ; Place ASCII '|' into W
0088 09AE      00285     call    putchar            ; Send out character
0089 028E      00286     incf    ADDRPTR,W          ; Go to next buffer value
008A 0E0F      00287     andlw   0Fh                ; And with F

008B 0643      00288     btfsc   STATUS,Z           ; Skip if this is NOT multiple of 16
```

```
008C 09A9      00289     call    printcrlf           ; Print CR and LF every 16 chars
008D 02AE      00290     incf    ADDRPTR,F           ; go to next address
008E 02EF      00291     decfsz  BYTECOUNT,F         ; Skip after last byte
008F 0A56      00292     goto    loopprintnums       ; Go back and print next char
0090 09A9      00293     call    printcrlf           ; Print CR and LF
0091 05A3      00294     bsf     STATUS,PA0          ; Set page bit to page 1
Message[306]: Crossing page boundary -- ensure page bits are set.
0092 0A6B      00295     goto    programpartisp      ; Go to program part through ISP
0093           00296 sendnoise
0093 0C4E      00297     movlw  'N'                  ; Place 'N' into W
0094 09AE      00298     call    putchar             ; Send char in W to terminal
0095 0C4F      00299     movlw  'O'                  ; Place 'O' into W
0096 09AE      00300     call    putchar             ; Send char in W to terminal
0097 0C49      00301     movlw  'I'                  ; Place 'I' into W
0098 09AE      00302     call    putchar             ; Send char in W to terminal
0099 0C53      00303     movlw  'S'                  ; Place 'S' into W
009A 09AE      00304     call    putchar             ; Send char in W to terminal
009B 0C45      00305     movlw  'E'                  ; Place 'E' into W
009C 09AE      00306     call    putchar             ; Send char in W to terminal
009D 09A9      00307     call    printcrlf           ; Print CR and LF
009E 0A1A      00308     goto    calget              ; RESET!
009F           00309 sendcsumbad
009F 0C43      00310     movlw   'C'                 ; Place 'C' into W
00A0 09AE      00311     call    putchar             ; Send char in W to terminal
00A1 0C53      00312     movlw   'S'                 ; Place 'S' into W
00A2 09AE      00313     call    putchar             ; Send char in W to terminal
00A3 0C55      00314     movlw   'U'                 ; Place 'U' into W
00A4 09AE      00315     call    putchar             ; Send char in W to terminal
00A5 0C4D      00316     movlw   'M'                 ; Place 'M' into W
00A6 09AE      00317     call    putchar             ; Send char in W to terminal
00A7 09A9      00318     call    printcrlf           ; Print CR and LF
00A8 0A1A      00319     goto    calget              ; RESET!
               00320
               00321 ; ****************************************
               00322 ; * printcrlf                             *
               00323 ; * Sends char .13 (Carrage Return) and    *
               00324 ; * char .10 (Line Feed) to RS-232 port    *
               00325 ; * by calling putchar.                    *
               00326 ; *      RAM used:  W                     *
               00327 ; ****************************************
00A9           00328 printcrlf
00A9 0C0D      00329     movlw   .13                 ; Value for CR placed into W
00AA 09AE      00330     call    putchar             ; Send char in W to terminal
00AB 0C0A      00331     movlw   .10                 ; Value for LF placed into W
00AC 09AE      00332     call    putchar             ; Send char in W to terminal
00AD 0800      00333     retlw   0                   ; Done - return!
               00334
               00335 ; ****************************************
               00336 ; * putchar                               *
               00337 ; * Print out the character stored in W    *
               00338 ; * by toggling the data to the RS-232     *
               00339 ; * output pin in software.                *
               00340 ; *     RAM used:  W,DATAREG,TEMP1         *
               00341 ; ****************************************
00AE           00342 putchar
00AE 0029      00343     movwf   DATAREG             ; Place character into DATAREG
00AF 0C09      00344     movlw   09h                 ; Place total number of bits into W
00B0 0028      00345     movwf   TEMP1               ; Init TEMP1 for bit counter
00B1 0403      00346     bcf     STATUS,C            ; Set carry to send start bit
00B2 0AB4      00347     goto    putloop1            ; Send out start bit
00B3           00348 putloop
00B3 0329      00349     rrf     DATAREG,F           ; Place next bit into carry
00B4           00350 putloop1
00B4 0703      00351     btfss   STATUS,C            ; Skip if carry was set
00B5 0466      00352     bcf     PORTB,SEROUT        ; Clear RS-232 serial output bit
00B6 0603      00353     btfsc   STATUS,C            ; Skip if carry was clear
```

```
00B7 0566      00354     bsf     PORTB,SEROUT        ; Set RS-232 serial output bit
               00355     delaybit                    ; Delay for one bit time
  0000             M     local dlylabels
                   M     ; 9600 baud, 8 bit, no parity, 104 us per bit, 52 uS per half bit
                   M     ; (8) shift/usage + (2) setup + (1) nop + (3 * 31) literal = (104) 4Mhz
00B8 0C1F          M     movlw   .31                 ; place 31 decimal literal into count
00B9 0027          M     movwf   COUNT               ; Initialize COUNT with loop count
00BA 0000          M     nop                         ; Add one cycle delay
00BB               M dlylabels
00BB 02E7          M     decfsz  COUNT,F             ; Decrement count until done
00BC 0ABB          M     goto    dlylabels           ; Not done delaying - go back!
00BD 02E8      00356     decfsz  TEMP1,F             ; Decrement bit counter, skip when done!
00BE 0AB3      00357     goto    putloop             ; Jump back and send next bit
00BF 0566      00358     bsf     PORTB,SEROUT        ; Send out stop bit
               00359     delaybit                    ; delay for stop bit
  0000             M     local dlylabels
                   M     ; 9600 baud, 8 bit, no parity, 104 us per bit, 52 uS per half bit
                   M     ; (8) shift/usage + (2) setup + (1) nop + (3 * 31) literal = (104) 4Mhz
00C0 0C1F          M     movlw   .31                 ; place 31 decimal literal into count
00C1 0027          M     movwf   COUNT               ; Initialize COUNT with loop count
00C2 0000          M     nop                         ; Add one cycle delay
00C3               M dlylabels
00C3 02E7          M     decfsz  COUNT,F             ; Decrement count until done
00C4 0AC3          M     goto    dlylabels           ; Not done delaying - go back!
00C5 0800      00360     retlw   0                   ; Done - RETURN
               00361
               00362 ; *********************************************************************
               00363 ; *  ISP routines from PICSTART-16C                                   *
               00364 ; *  Converted from PIC17C42 to PIC16C5X code by John Day             *
               00365 ; *  Originially written by Jim Pepping                              *
               00366 ; *********************************************************************
0200           00367     ORG 200                     ; ISP routines stored on page 1
               00368
               00369 ; *********************************************************************
               00370 ; * poweroffisp                                                      *
               00371 ; * Power off application PIC - turn off VPP and reset device after *
               00372 ; * programming pass is complete                                     *
               00373 ; *********************************************************************
0200           00374 poweroffisp
0200 04A6      00375     bcf     PORTB,VPPON         ; Turn off VPP 13 volts
0201 0586      00376     bsf     PORTB,GNDON         ; Apply 0 V to MCLR to reset PIC
0202 0CC1      00377     movlw   b'11000001'         ; RB6,7 set to inputs
0203 0006      00378     tris    PORTB               ; Move to tris registers
0204 0486      00379     bcf     PORTB,GNDON   ; Allow MCLR to go back to 5 volts, deassert reset
0205 0526      00380     bsf     PORTB,WORKLED       ; Turn off WORK LED
0206 0800      00381     retlw 0                     ; Done so return!
               00382
               00383 ; *********************************************************************
               00384 ; * testmodeisp                                                      *
               00385 ; * Apply VPP voltage to place application PIC into test mode.      *
               00386 ; * this enables ISP programming to proceed                         *
               00387 ; *     RAM used:       TEMP                                        *
               00388 ; *********************************************************************
0207           00389 testmodeisp
0207 0C08      00390     movlw   b'00001000'         ; Serial OFF, LEDS OFF, VPP OFF
0208 0026      00391     movwf   PORTB               ; Place "0" into port b latch register
0209 04A6      00392     bcf     PORTB,VPPON         ; Turn off VPP just in case!
020A 0586      00393     bsf     PORTB,GNDON         ; Apply 0 volts to MCLR
020B 0C01      00394     movlw   b'00000001'         ; RB6,7 set to outputs
020C 0006      00395     tris    PORTB               ; Move to tris registers
020D 0206      00396     movf    PORTB,W             ; Place PORT B state into W
020E 002D      00397     movwf   TEMP                ; Move state to TEMP
020F 048D      00398     bcf     TEMP,4              ; Turn off MCLR GND
0210 05AD      00399     bsf     TEMP,5              ; Turn on VPP voltage
0211 020D      00400     movf    TEMP,W              ; Place TEMP into W
0212 0026      00401     movwf   PORTB               ; Turn OFF GND and ON VPP
```

```
0213 0546      00402     bsf     PORTB,DONELED       ; Turn ON GREEN LED
0214 0800      00403     retlw 0                     ; Done so return!
               00404
               00405 ; *****************************************************************
               00406 ; * p16cispout                                                    *
               00407 ; * Send 14-bit data word to application PIC for writing this data  *
               00408 ; * to it's program memory.   The data to be sent is stored in both *
               00409 ; * HIBYTE (6 MSBs only) and LOBYTE.                              *
               00410 ; *     RAM used:       TEMP, W, HIBYTE (inputs), LOBYTE (inputs)   *
               00411 ; *****************************************************************
0215           00412 P16cispout
0215 0C0E      00413     movlw   .14               ; Place 14 into W for bit counter
0216 002D      00414     movwf   TEMP              ; Use TEMP as bit counter
0217 04C6      00415     bcf     PORTB,ISPCLOCK    ; Clear CLOCK line
0218 04E6      00416     bcf     PORTB,ISPDATA     ; Clear DATA line
0219 0C01      00417     movlw   DATISPOUT         ; Place tris value for data output
021A 0006      00418     tris    PORTB             ; Set tris latch as data output
021B 04E6      00419     bcf     PORTB,ISPDATA     ; Send a start bit (0)
021C 05C6      00420     bsf     PORTB,ISPCLOCK    ; Set CLOCK output
021D 04C6      00421     bcf     PORTB,ISPCLOCK    ; Clear CLOCK output (clock start bit)
021E           00422 P16cispoutloop
021E 0403      00423     bcf     STATUS,C          ; Clear carry bit to start clean
021F 04E6      00424     bcf     PORTB,ISPDATA     ; Clear DATA bit to start (assume 0)
0220 0329      00425     rrf     HIBYTE,F          ; Rotate HIBYTE output
0221 032A      00426     rrf     LOBYTE,F          ; Rotate LOBYTE output
0222 0603      00427     btfsc   STATUS,C          ; Skip if data bit is zero
0223 05E6      00428     bsf     PORTB,ISPDATA     ; Set DATA line to send a one
0224 05C6      00429     bsf     PORTB,ISPCLOCK    ; Set CLOCK output
0225 04C6      00430     bcf     PORTB,ISPCLOCK    ; Clear CLOCK output (clock bit)
0226 02ED      00431     decfsz  TEMP,F            ; Decrement bit counter, skip when done
0227 0A1E      00432     goto    P16cispoutloop    ; Jump back and send next bit
0228 04E6      00433     bcf     PORTB,ISPDATA     ; Send a stop bit (0)
0229 05C6      00434     bsf     PORTB,ISPCLOCK    ; Set CLOCK output
022A 04C6      00435     bcf     PORTB,ISPCLOCK    ; Clear CLOCK output (clock stop bit)
022B 0800      00436     retlw 0                   ; Done so return!
               00437
               00438 ; *****************************************************************
               00439 ; * p16cispin                                                     *
               00440 ; * Receive 14-bit data word from application PIC for reading this  *
               00441 ; * data from it's program memory.   The data received is stored in *
               00442 ; * both HIBYTE (6 MSBs only) and LOBYTE.                          *
               00443 ; *     RAM used:       TEMP, W, HIBYTE (output), LOBYTE (output)   *
               00444 ; *****************************************************************
022C           00445 P16cispin
022C 0C0E      00446     movlw   .14               ; Place 14 data bit count value into W
022D 002D      00447     movwf   TEMP              ; Init TEMP and use for bit counter
022E 0069      00448     clrf    HIBYTE            ; Clear recieved HIBYTE register
022F 006A      00449     clrf    LOBYTE            ; Clear recieved LOBYTE register
0230 0403      00450     bcf     STATUS,C          ; Clear carry bit to start clean
0231 04C6      00451     bcf     PORTB,ISPCLOCK    ; Clear CLOCK output
0232 04E6      00452     bcf     PORTB,ISPDATA     ; Clear DATA output
0233 0C81      00453     movlw   DATISPIN          ; Place tris value for data input into W
0234 0006      00454     tris    PORTB             ; Set up tris latch for data input
0235 05C6      00455     bsf     PORTB,ISPCLOCK    ; Send a single clock to start things going
0236 04C6      00456     bcf     PORTB,ISPCLOCK    ; Clear CLOCK to start receive
0237           00457 P16cispinloop
0237 05C6      00458     bsf     PORTB,ISPCLOCK    ; Set CLOCK bit
0238 0000      00459     nop                       ; Wait one cycle
0239 0403      00460     bcf     STATUS,C          ; Clear carry bit, assume 0 read
023A 06E6      00461     btfsc   PORTB,ISPDATA     ; Check the data, skip if it was zero
023B 0503      00462     bsf     STATUS,C          ; Set carry bit if data was one
023C 0329      00463     rrf     HIBYTE,F          ; Move recevied bit into HIBYTE
023D 032A      00464     rrf     LOBYTE,F          ; Update LOBYTE
023E 04C6      00465     bcf     PORTB,ISPCLOCK    ; Clear CLOCK line
023F 0000      00466     nop                       ; Wait one cycle
0240 0000      00467     nop                       ; Wait one cycle
```

```
0241 02ED      00468     decfsz  TEMP,F              ; Decrement bit counter, skip when zero
0242 0A37      00469     goto    P16cispinloop       ; Jump back and receive next bit
0243 05C6      00470     bsf     PORTB,ISPCLOCK      ; Clock a stop bit (0)
0244 0000      00471     nop                         ; Wait one cycle
0245 04C6      00472     bcf     PORTB,ISPCLOCK      ; Clear CLOCK to send bit
0246 0000      00473     nop                         ; Wait one cycle
0247 0403      00474     bcf     STATUS,C            ; Clear carry bit
0248 0329      00475     rrf     HIBYTE,F            ; Update HIBYTE with the data
0249 032A      00476     rrf     LOBYTE,F            ; Update LOBYTE
024A 0403      00477     bcf     STATUS,C            ; Clear carry bit
024B 0329      00478     rrf     HIBYTE,F            ; Update HIBYTE with the data
024C 032A      00479     rrf     LOBYTE,F            ; Update LOBYTE with the data
024D 04C6      00480     bcf     PORTB,ISPCLOCK      ; Clear CLOCK line
024E 04E6      00481     bcf     PORTB,ISPDATA       ; Clear DATA line
024F 0C01      00482     movlw   DATISPOUT           ; Place tris value for data output into W
0250 0006      00483     tris    PORTB               ; Set tris to data output
0251 0800      00484     retlw 0                     ; Done so RETURN!
               00485
               00486 ; *****************************************************************
               00487 ; * commandisp                                                    *
               00488 ; * Send 6-bit ISP command to application PIC.  The command is sent *
               00489 ; * in the W register and later stored in LOBYTE for shifting.     *
               00490 ; *     RAM used: LOBYTE, W, TEMP                                 *
               00491 ; *****************************************************************
0252           00492 commandisp
0252 002A      00493     movwf   LOBYTE              ; Place command into LOBYTE
0253 0C06      00494     movlw   CMDISPCNT           ; Place number of command bits into W
0254 002D      00495     movwf   TEMP                ; Use TEMP as command bit counter
0255 04E6      00496     bcf     PORTB,ISPDATA       ; Clear DATA line
0256 04C6      00497     bcf     PORTB,ISPCLOCK      ; Clear CLOCK line
0257 0C01      00498     movlw   DATISPOUT           ; Place tris value for data output into W
0258 0006      00499     tris    PORTB               ; Set tris to data output
0259           00500 P16cispcmmdoutloop
0259 0403      00501     bcf     STATUS,C            ; Clear carry bit to start clean
025A 04E6      00502     bcf     PORTB,ISPDATA       ; Clear the DATA line to start
025B 032A      00503     rrf     LOBYTE,F            ; Update carry with next CMD bit to send
025C 0603      00504     btfsc   STATUS,C            ; Skip if bit is supposed to be 0
025D 05E6      00505     bsf     PORTB,ISPDATA       ; Command bit was a one - set DATA to one
025E 05C6      00506     bsf     PORTB,ISPCLOCK      ; Set CLOCK line to clock the data
025F 0000      00507     nop                         ; Wait one cycle
0260 04C6      00508     bcf     PORTB,ISPCLOCK      ; Clear CLOCK line to clock data
0261 02ED      00509     decfsz  TEMP,F              ; Decement bit counter TEMP, skip when done
0262 0A59      00510     goto    P16cispcmmdoutloop  ; Jump back and send next cmd bit
0263 0000      00511     nop                         ; Wait one cycle
0264 04E6      00512     bcf     PORTB,ISPDATA       ; Clear DATA line
0265 04C6      00513     bcf     PORTB,ISPCLOCK      ; Clear CLOCK line
0266 0C81      00514     movlw   DATISPIN            ; Place tris value for data input into W
0267 0006      00515     tris    PORTB               ; set as input to avoid any contention
0268 0000      00516     nop                         ; Wait one cycle
0269 0000      00517     nop                         ; Wait one cycle
026A 0800      00518     retlw 0                     ; Done - return!
               00519
               00520 ; *****************************************************************
               00521 ; * programpartisp                                                *
               00522 ; * Main ISP programming loop.  Reads data starting at STARTCALBYTE  *
               00523 ; * and calls programming subroutines to program and verify this     *
               00524 ; * data into the application PIC.                                 *
               00525 ; *     RAM used: LOADDR, HIADDR, LODATA, HIDATA, FSR, LOBYTE, HIBYTE*
               00526 ; *****************************************************************
026B           00527 programpartisp
026B 0907      00528     call    testmodeisp         ; Place PIC into test/program mode
026C 0064      00529     clrf    FSR                 ; Point to bank 0
026D 0210      00530     movf    STARTCALBYTE,W ; Upper order address of data to be stored into W
026E 0027      00531     movwf   HIADDR              ; place into counter
026F 0211      00532     movf    STARTCALBYTE+1,W ; Lower order address byte of data to be stored
0270 0028      00533     movwf   LOADDR              ; place into counter
```

```
0271 00E8      00534     decf    LOADDR,F            ; Subtract one from loop constant
0272 02A7      00535     incf    HIADDR,F            ; Add one for loop constant
0273           00536 programsetptr
0273 0C06      00537     movlw   CMDISPINCRADDR      ; Increment address command load into W
0274 0952      00538     call    commandisp          ; Send command to PIC
0275 02E8      00539     decfsz  LOADDR,F            ; Decrement lower address
0276 0A73      00540     goto    programsetptr       ; Go back again
0277 02E7      00541     decfsz  HIADDR,F            ; Decrement high address
0278 0A73      00542     goto    programsetptr       ; Go back again
0279 0C03      00543     movlw   .3                  ; Place start pointer into W, offset address
027A 008B      00544     subwf   TIMEHIGH,W          ; Restore byte count into W
027B 002F      00545     movwf   BYTECOUNT           ; Place into byte counter
027C 0C12      00546     movlw   STARTCALBYTE+2      ; Place start of REAL DATA address into W
027D 002E      00547     movwf   ADDRPTR             ; Update pointer
027E           00548 programisploop
027E 0C34      00549     movlw   UPPER6BITS          ; retlw instruction opcode placed into W
027F 0027      00550     movwf   HIDATA              ; Set up upper bits of program word
               00551     addrtofsr ADDRPTR           ; Set up FSR to point to next value
0280 0C10          M     movlw   STARTCALBYTE        ; Place base address into W
0281 008E          M     subwf   ADDRPTR,w           ; Offset by STARTCALBYTE
0282 0024          M     movwf   FSR                 ; Place into FSR
0283 06A4          M     btfsc   FSR,5               ; Shift bits 4,5 to 5,6
0284 05C4          M     bsf     FSR,6
0285 04A4          M     bcf     FSR,5
0286 0684          M     btfsc   FSR,4
0287 05A4          M     bsf     FSR,5
0288 0584          M     bsf     FSR,4
0289 0200      00552     movf    INDF,W              ; Place next cal param into W
028A 0028      00553     movwf   LODATA              ; Move it out to LODATA
028B 0208      00554     movf    LODATA,W            ; Place LODATA into LOBYTE
028C 002A      00555     movwf   LOBYTE              ;
028D 0207      00556     movf    HIDATA,W            ; Place HIDATA into HIBYTE
028E 0029      00557     movwf   HIBYTE              ;
028F 006B      00558     clrf    PULSECNT            ; Clear pulse counter
0290           00559 pgmispcntloop
0290 05E3      00560     bsf     STATUS,VFYYES       ; Set verify flag
0291 09B1      00561     call    pgmvfyisp           ; Program and verify this byte
0292 02AB      00562     incf    PULSECNT,F          ; Increment pulse counter
0293 0C19      00563     movlw   .25                 ; Place 25 count into W
0294 008B      00564     subwf   PULSECNT,w          ; Subtract pulse count from 25
0295 0643      00565     btfsc   STATUS,Z            ; Skip if NOT 25 pulse counts
0296 0AA9      00566     goto    pgmispfail          ; Jump to program failed - only try 25 times
0297 0209      00567     movf    HIBYTE,w            ; Subtract programmed and read data
0298 0087      00568     subwf   HIDATA,w
0299 0743      00569     btfss   STATUS,Z            ; Skip if programmed is OK
029A 0A90      00570     goto    pgmispcntloop       ; Miscompare - program it again!
029B 020A      00571     movf    LOBYTE,w            ; Subtract programmed and read data
029C 0088      00572     subwf   LODATA,w
029D 0743      00573     btfss   STATUS,Z            ; Skip if programmed is OK
029E 0A90      00574     goto    pgmispcntloop       ; Miscompare - program it again!
029F 0040      00575     clrw                        ; Clear W reg
02A0 01CB      00576     addwf   PULSECNT,W          ; now do 3 times overprogramming pulses
02A1 01CB      00577     addwf   PULSECNT,W
02A2 01CB      00578     addwf   PULSECNT,W
02A3 002B      00579     movwf   PULSECNT            ; Add 3X pulsecount to pulsecount
02A4           00580 pgmisp3X
02A4 04E3      00581     bcf     STATUS,VFYYES       ; Clear verify flag
02A5 09B1      00582     call    pgmvfyisp           ; Program this byte
02A6 02EB      00583     decfsz  PULSECNT,F          ; Decrement pulse counter, skip when done
02A7 0AA4      00584     goto    pgmisp3X            ; Loop back and program again!
02A8 0AAA      00585     goto    prgnextbyte         ; Done - jump to program next byte!
02A9           00586 pgmispfail
02A9 0446      00587     bcf     PORTB,DONELED       ; Failure - clear green LED!
02AA           00588 prgnextbyte
02AA 0C06      00589     movlw   CMDISPINCRADDR      ; Increiment address command load into W
02AB 0952      00590     call    commandisp          ; Send command to PIC
```

```
02AC 02AE      00591     incf    ADDRPTR,F           ; Increment pointer to next address
02AD 02EF      00592     decfsz  BYTECOUNT,F         ; See if we sent last byte
02AE 0A7E      00593     goto    programisploop      ; Jump back and send next byte
02AF 0900      00594     call    poweroffisp         ; Done - power off PIC and reset it!
02B0           00595 self
02B0 0AB0      00596     goto    self                ; Done with programming - wait here!
               00597
               00598
               00599
               00600 ; *****************************************************************
               00601 ; * pgmvfyisp                                                     *
               00602 ; * Program and/or Veryify a word in program memory on the        *
               00603 ; * application PIC.  The data to be programmed is in HIDATA and   *
               00604 ; * LODATA.                                                       *
               00605 ; *     RAM used: HIBYTE, LOBYTE, HIDATA, LODATA, TEMP           *
               00606 ; *****************************************************************
02B1           00607 pgmvfyisp
02B1           00608 loadcisp
02B1 0C02      00609     movlw   CMDISPLOAD          ; Place load data command into W
02B2 0952      00610     call    commandisp          ; Send load data command to PIC
02B3 0000      00611     nop                         ; Wait one cycle
02B4 0000      00612     nop                         ; Wait one cycle
02B5 0000      00613     nop                         ; Wait one cycle
02B6 0208      00614     movf    LODATA,w            ; Place LODATA byte into W
02B7 002A      00615     movwf   LOBYTE              ; Move it to LOBYTE reg
02B8 0207      00616     movf    HIDATA,w            ; Place HIDATA byte into W
02B9 0029      00617     movwf   HIBYTE              ; Move it to HIBYTE reg
02BA 0915      00618     call    P16cispout          ; Send data to PIC
02BB 0C08      00619     movlw   CMDISPPGMSTART      ; Place start programming command into W
02BC 0952      00620     call    commandisp          ; Send start programming command to PIC
02BD           00621 delay100us
02BD 0C20      00622     movlw   .32                 ; Place 32 into W
02BE 0000      00623     nop                         ; Wait one cycle
02BF 002D      00624     movwf   TEMP                ; Move it to TEMP for delay counter
02C0           00625 loopprgm
02C0 02ED      00626     decfsz  TEMP,F              ; Decrement TEMP, skip when delay done
02C1 0AC0      00627     goto    loopprgm            ; Jump back and loop delay
02C2 0C0E      00628     movlw   CMDISPPGMEND        ; Place stop programming command into W
02C3 0952      00629     call    commandisp          ; Send end programming command to PIC
02C4 07E3      00630     btfss   STATUS,VFYYES     ; Skip if we are supposed to verify this time
02C5 0800      00631     retlw 0                     ; Done - return!
02C6 0000      00632     nop                         ; Wait one cycle
02C7           00633 readcisp
02C7 0C04      00634     movlw   CMDISPREAD          ; Place read data command into W
02C8 0952      00635     call    commandisp          ; Send read data command to PIC
02C9 092C      00636     call    P16cispin           ; Read programmed data
02CA 0800      00637     retlw 0                     ; Done - return!
               00638     END
```

```
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXX---------- ---------------- ---------------- ----------------
0200 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0240 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0280 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
02C0 : XXXXXXXXXXX----- ---------------- ---------------- ----------------
07C0 : ---------------- ---------------- ---------------- ---------------X
0FC0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   402
Program Memory Words Free:  1646


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     2 reported,     0 suppressed
```

APPENDIX B: ISPTEST.ASM

```
MPASM 01.40.01 Intermediate   ISPTEST.ASM   3-31-1997  10:55:57         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001 ; Filename: ISPTEST.ASM
                     00002 ; ********************************************
                     00003 ; * Author:   John Day                       *
                     00004 ; *           Sr. Field Applications Engineer  *
                     00005 ; *           Microchip Technology             *
                     00006 ; * Revision: 1.0                            *
                     00007 ; * Date      August 25, 1995                *
                     00008 ; * Part:     PIC16CXX                       *
                     00009 ; * Compiled using MPASM V1.40               *
                     00010 ; ********************************************
                     00011 ; * Include files:                           *
                     00012 ; *           P16CXX.ASM                     *
                     00013 ; ********************************************
                     00014 ; * Fuses:    OSC:  XT (4.0 Mhz xtal)        *
                     00015 ; *           WDT:  OFF                      *
                     00016 ; *            CP:  OFF                      *
                     00017 ; *         PWRTE:  OFF                      *
                     00018 ; ********************************************************************************
                     00019 ; * This program is intended to be used as a code example to               *
                     00020 ; * show how to comunicate with a manufacturing test jig that            *
                     00021 ; * allows this PIC16CXX device to self program.   The RB6 and RB7        *
                     00022 ; * lines of this PIC16CXX device are used to clock the data from        *
                     00023 ; * this device to the test jig (running ISPPRGM.ASM).  Once the         *
                     00024 ; * PIC16C58 running ISPPRGM in the test jig receives the data,          *
                     00025 ; * it places this device in test mode and programs these parameters.    *
                     00026 ; * The code with comments “TEST -“ is used to create some fakecalibration *
                     00027 ; * parameters that are first written to addresses STARTCALBYTE through   *
                     00028 ; * ENDCALBYTE and later used to call the self-programming algorithm.    *
                     00029 ; * Replace this code with your parameter calculation procedure,         *
                     00030 ; * placing each parameter into the STARTCALBYTE to ENDCALBYTE           *
                     00031 ; * file register addresses (16 are used in this example).  The address  *
                     00032 ; * “lookuptable” is used by the main code later on for the final lookup *
                     00033 ; * table of calibration constants.  16 words are reserved for this lookup *
                     00034 ; * table.                                                              *
                     00035 ; ********************************************************************************
                     00036 ; * Program Memory:                                                    *
                     00037 ; *             49 Words - communication with test jig                 *
                     00038 ; *             17 Words - calibration look-up table (16 bytes of data)  *
                     00039 ; *             13 Words - Test Code to generate Calibration Constants  *
                     00040 ; * RAM Memory:                                                        *
                     00041 ; *             16 Bytes -Temporary- Store 16 bytes of calibration constant*
                     00042 ; *              4 Bytes -Temporary- Store 4 bytes of temp variables    *
                     00043 ; ********************************************************************************
                     00044
Warning[217]: Hex file format specified on command line.
                     00045     list p=16C71,f=inhx8m
                     00046     include <p16C71.inc>
                     00001         LIST
                     00002 ; P16C71.INC  Standard Header File, Version 1.00 Microchip Technology, Inc.
                     00142         LIST
2007 3FF1            00047     __CONFIG _CP_OFF&_WDT_OFF&_XT_OSC&_PWRTE_OFF
                     00048
                     00049 ; ***********************************
                     00050 ; * Port A (RA0-RA4) bit definitions *
                     00051 ; ***********************************
                     00052 ; Port A is not used in this test program
                     00053
                     00054 ; ***********************************
                     00055 ; * Port B (RB0-RB7) bit definitions *
```

```
                00056 ; ***********************************
                00057 #define     CLOCK   6 ; clock line for ISP
                00058 #define     DATA    7 ; data line for ISP
                00059 ; Port pins RB0-5 are not used in this test program
                00060
                00061 ; ***********************************
                00062 ; * RAM register usage definition    *
                00063 ; ***********************************
  0000000C      00064 CSUMTOTAL   EQU 0Ch ; Address for checksum var
  0000000D      00065 COUNT       EQU 0Dh ; Address for COUNT var
  0000000E      00066 DATAREG     EQU 0Eh ; Address for Data output register var
  0000000F      00067 COUNTDLY    EQU 0Fh ; Address for clock delay counter
                00068
                00069 ; These two symbols are used for the start and end address
                00070 ; in RAM where the calibration bytes are stored.  There are 16 bytes
                00071 ; to be stored in this example; however, you can increase or
                00072 ; decrease the number of bytes by changing the STARTCALBYTE or ENDCALBYTE
                00073 ; address values.
                00074
  00000010      00075 STARTCALBYTE    EQU 10h     ; Address pointer for start CAL byte
  0000002F      00076 ENDCALBYTE      EQU 2Fh     ; Address pointer for end CAL byte
                00077
                00078 ; Table length of lookup table (number of CAL parameters to be stored)
                00079
  00000020      00080 CALTABLELENGTH EQU  ENDCALBYTE - STARTCALBYTE + 1
                00081
0000            00082     ORG 0
                00083 ; ******************************************************************
                00084 ; * testcode routine                                               *
                00085 ; * TEST code - sets up RAM register with register address as data *
                00086 ; * Uses file register STARTCALBYTE through ENDCALBYTE to store the*
                00087 ; * calibration values that are to be programmed into the lookup    *
                00088 ; * table by the test jig running ISPPRGM.                         *
                00089 ; * Customer would place calibration code here and make sure that  *
                00090 ; * calibration constants start at address STARTCALBYTE           *
                00091 ; ******************************************************************
0000            00092 testcode
0000 3010       00093     movlw   STARTCALBYTE    ; TEST -
0001 0084       00094     movwf   FSR             ; TEST - Init FSR with start of RAM addres
0002            00095 looptestram
0002 0804       00096     movf    FSR,W           ; TEST - Place address into W
0003 0080       00097     movwf   INDF            ; TEST - Place address into RAM data byte
0004 0A84       00098     incf    FSR,F           ; TEST - Move to next address
0005 0804       00099     movf    FSR,W           ; TEST - Place current address into W
0006 3C30       00100     sublw   ENDCALBYTE+1    ; TEST - Subtract from end of RAM
0007 1D03       00101     btfss   STATUS,Z        ; TEST - Skip if at END of ram
0008 2802       00102     goto    looptestram     ; TEST - Jump back and init next RAM byte
0009 0103       00103     clrw                    ; TEST - Clear W
000A 200F       00104     call    lookuptable     ; TEST - Get first CAL value from lookup table
000B 3CFF       00105     sublw   0FFh            ; TEST - Check if lookup CAL table is blank
000C 1903       00106     btfsc   STATUS,Z        ; TEST - Skip if table is NOT blank
000D 2830       00107     goto    calsend         ; TEST - Table blank - send out cal parameters
000E            00108 mainloop
000E 280E       00109     goto    mainloop        ; TEST - Jump back to self since CAL is done
                00110
                00111 ; ******************************************************************
                00112 ; * lookuptable                                                    *
                00113 ; * Calibration constants look-up table.  This is where the CAL    *
                00114 ; * Constants will be stored via ISP protocol later.  Note it is   *
                00115 ; * blank, since these values will be pogrammed by the test jig    *
                00116 ; * running ISPPRGM later.                                        *
                00117 ; *     Input Variable:  W stores index for table lookup         *
                00118 ; *     Output Variable: W returns with the calibration constant  *
                00119 ; * NOTE: Blank table when programmed reads “FF” for all locations *
                00120 ; ******************************************************************
000F            00121 lookuptable
```

```
000F 0782      00122     addwf   PCL,F           ; Place the calibration constant table here!
               00123
002F           00124     ORG     lookuptable + CALTABLELENGTH
002F 34FF      00125     retlw   0FFh            ; Return FF at last location for a blank table
               00126
               00127 ; *****************************************************************
               00128 ; * calsend subroutine                                            *
               00129 ; * Send the calibration data stored in locations STARTCALBYTE     *
               00130 ; * through ENDCALBYTE in RAM to the programming jig using a serial*
               00131 ; * clock and data protocol                                       *
               00132 ; *     Input Variables:   STARTCALBYTE through ENDCALBYTE       *
               00133 ; *****************************************************************
0030           00134 calsend
0030 018C      00135     clrf    CSUMTOTAL       ; Clear CSUMTOTAL reg for delay counter
0031 018D      00136     clrf    COUNT           ; Clear COUNT reg to delay counter
0032           00137 delayloop                   ; Delay for 100 mS to wait for prog jig wakeup
0032 0B8D      00138     decfsz  COUNT,F         ; Decrement COUNT and skip when zero
0033 2832      00139     goto    delayloop       ; Go back and delay again
0034 0B8C      00140     decfsz  CSUMTOTAL,F     ; Decrement CSUMTOTAL and skip when zero
0035 2832      00141     goto    delayloop       ; Go back and delay again
0036 0186      00142     clrf    PORTB           ; Place “0” into port b latch register
0037 1683      00143     bsf     STATUS,RP0      ; Switch to bank 1
0038 303F      00144     movlw   b’00111111’     ; RB6,7 set to outputs
Message[302]: Register in operand not in bank 0.  Ensure that bank bits are correct.
0039 0086      00145     movwf   TRISB           ; Move to TRIS registers
003A 1283      00146     bcf     STATUS,RP0      ; Switch to bank 0
003B 018C      00147     clrf    CSUMTOTAL       ; Clear checksum total byte
003C 3001      00148     movlw   high lookuptable+1 ; place MSB of first addr of cal table into W
003D 204D      00149     call    sendcalbyte     ; Send the high address out
003E 3010      00150     movlw   low lookuptable+1  ; place LSB of first addr of cal table into W
003F 204D      00151     call    sendcalbyte     ; Send low address out
0040 3010      00152     movlw   STARTCALBYTE    ; Place RAM start address of first cal byte
0041 0084      00153     movwf   FSR             ; Place this into FSR
0042           00154 loopcal
0042 0800      00155     movf    INDF,W          ; Place data into W
0043 204D      00156     call    sendcalbyte     ; Send the byte out
0044 0A84      00157     incf    FSR,F           ; Move to the next cal byte
0045 0804      00158     movf    FSR,W           ; Place byte address into W
0046 3C30      00159     sublw   ENDCALBYTE+1    ; Set Z bit if we are at the end of CAL data
0047 1D03      00160     btfss   STATUS,Z        ; Skip if we are done
0048 2842      00161     goto    loopcal         ; Go back for next byte
0049 080C      00162     movf    CSUMTOTAL,W     ; place checksum total into W
004A 204D      00163     call    sendcalbyte     ; Send the checksum out
004B 0186      00164     clrf    PORTB           ; clear out port pins
004C           00165 calsenddone
004C 284C      00166     goto    calsenddone     ; We are done - go home!
               00167
               00168 ; *****************************************************************
               00169 ; * sendcalbyte subroutine                                       *
               00170 ; * Send one byte of calibration data to the programming jig      *
               00171 ; *     Input Variable: W contains the byte to be sent           *
               00172 ; *****************************************************************
004D           00173 sendcalbyte
004D 008E      00174     movwf   DATAREG         ; Place send byte into data register
004E 078C      00175     addwf   CSUMTOTAL,F     ; Update checksum total
004F 3008      00176     movlw   .8              ; Place 8 into W
0050 008D      00177     movwf   COUNT           ; set up counter register
0051           00178 loopsendcal
0051 1706      00179     bsf     PORTB,CLOCK     ; Set clock line high
0052 205C      00180     call    delaysend       ; Wait for test jig to synch up
0053 0D8E      00181     rlf     DATAREG,F       ; Rotate to next bit
0054 1786      00182     bsf     PORTB,DATA      ; Assume data bit is high
0055 1C03      00183     btfss   STATUS,C        ; Skip if the data bit was high
0056 1386      00184     bcf     PORTB,DATA      ; Set data bit to low
0057 1306      00185     bcf     PORTB,CLOCK     ; Clear clock bit to clock data out
0058 205C      00186     call    delaysend       ; Wait for test jig to synch up
```

```
0059 0B8D      00187     decfsz  COUNT,F         ; Skip after 8 bits
005A 2851      00188     goto    loopsendcal     ; Jump back and send next bit
005B 0008      00189     return                  ; We are done with this byte so return!
               00190
               00191 ; *****************************************************************
               00192 ; * delaysend subroutine                                          *
               00193 ; * Delay for 50 ms to wait for the programming jig to synch up    *
               00194 ; *****************************************************************
005C           00195 delaysend
005C 3010      00196     movlw   10h             ; Delay for 16 loops
005D 008F      00197     movwf   COUNTDLY        ; Use COUNTDLY as delay count variable
005E           00198 loopdelaysend
005E 0B8F      00199     decfsz  COUNTDLY,F      ; Decrement COUNTDLY and skip when done
005F 285E      00200     goto    loopdelaysend   ; Jump back for more delay
0060 0008      00201     return
               00202         END

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX ---------------- ---------------X XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX X--------------- ----------------
2000 : -------X-------- ---------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:    66
Program Memory Words Free:   958


Errors   :     0
Warnings :     1 reported,     0 suppressed
Messages :     1 reported,     0 suppressed
```

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: http://www.microchip.com

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

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

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

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton
Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

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

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong
Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India
Microchip Technology India
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

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

Shanghai
Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hongiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore
Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

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

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleone
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81-4-5471- 6166 Fax: 81-4-5471-6122

5/8/97

MICROCHIP