```
        TITLE "Simple PICTIC Time Interval Counter - Richard H McCorkle, 12/20/08"

;Copyright © 2008, Richard H McCorkle. All rights reserved. This program
;may be used only for non-commercial purposes, and carries no warranty
;of any kind, including any implied warranty of fitness for any particular
;purpose. The interpolator design and support code may be used at no
;cost for non-commercial applications only. For commercial applications
;contact the author at mccorkle@ptialaska.net for licensing terms.

;*****************************************************************
;                          CPU Type & Fuses:
;*****************************************************************

        LIST n=58, p=PIC16F688
        errorlevel     1
        include P16F688.INC
        __CONFIG _INTOSCIO & _PWRTE_ON & _WDT_OFF & _FCMEN_OFF & _IESO_OFF & _BOD_OFF &
_CPD_OFF & _CP_OFF & _MCLRE_OFF

;*****************************************************************
;                        Function Description:
;*****************************************************************
;  The Simple PICTIC demonstrates the interpolation hardware and
;support code featured in the PICTIC Module. It incorporates only
;the core hardware and software features of the PICTIC Module and
;requires leading edge triggered 1PPS TTL inputs with a small delay
;between the start and stop events for proper operation. It is
;intended as a platform for evaluating the interpolator design
;for use in other high resolution TIC applications.
;  Time to voltage interpolation is used to drastically reduce the
;hardware required for low cost and ease of assembly. Modern PIC
;microcontrollers include multiple 10-bit A/D inputs and have a
;fast enough conversion time to read the start and stop capacitor
;voltages directly between samples if they are buffered using a
;CMOS op-amp. The PICTIC interpolators use just enough circuitry
;to charge two sample capacitors with a constant current during
;the start and stop intervals, buffer and read the capacitor
;voltages using the PIC ADC, and discharge the capacitors after
;the data is read. A fourIC gating circuit provides start and
;stop outputs to the interpolators and a gate output synchronized
;to the timebase for the main counter in the PIC. By using
;minimal hardware and adapting a traditional interpolator design
;for direct logic drive and maximum use of features already
;present in a PIC the cost and complexity of an interpolating
;time interval counter is drastically reduced.
;  Interpolation accuracy is reduced due to variations in supply
;voltage, capacitance, charge current, DAC accuracy, and offset
;voltage with changing temperature and component age. To maintain
;accuracy a means of calibrating the interpolator is required. By
;only using a portion of the A/D range for the data and a portion
;for offset and scaling variations, sufficient A/D range is
;available to compensate for variations in temperature and aging
;dynamically in software so common low-cost components can be
;used. Using a software calibration routine to compensate for
;variations dynamically relaxes the temperature and stability
;requirements of the components used, resulting in a low cost,
;high accuracy, dual interpolator.
;  The key to understanding the PICTIC is each interpolator
;sample is a minimum of one clock period and a maximum of two
;clock periods in duration due to the synchronizers used. The
;initial clock period provides sufficient time for the sample
;switch to turn on fully before the measurement interval begins.
;The difference between the switch turn on time and the beginning
;of the measurement interval sets the minimum ADC count returned.
;Whenever the input and timebase leading edges pass through phase
;coincidence the ADC count passes from maximum to minimum or
;minimum to maximum allowing determination of the interpolator
;minimum and maximum ADC count limits from the sample data.
;  Dual software peak detectors are used over many sample periods
;to determine the minimum and maximum ADC values returned over
;a calibration period. If sufficient samples are taken the zero
```

```
;offset and span of the interpolator hardware can be determined.
;A low stability timebase varies sufficiently to insure there
;are frequent phase coincidences between the timebase and the
;inputs and simplifies calibration. When the timebase drifts off
;frequency the inputs pass coincidence at a recurring rate. If
;the inputs maintain phase coincidence with the timebase for an
;extended period the calibration routine stops functioning
;correctly as no zero and span limits are reached during the
;calibration period. This is important to note as the calibration
;routine may need to use longer calibration periods or be
;disabled completely when using a high stability timebase.
;  During initial setup the interpolator sample capacitor value
;is selected for the timebase frequency and charge current used.
;The charge current is then adjusted to return a peak to peak
;span of exactly 400 ADC counts. Typically ADC counts from
;250-350 minimum to 650-750 maximum will be returned. The actual
;minimum and maximum values will vary depending on the hardware
;used, but the currents are adjusted for identical peak to peak
;spans of 400 counts in both interpolators.
;  To simplify initial calibration the peak values can be
;displayed continuously and the peak detectors can be reset
;manually after an adjustment to speed initial setup. Once
;calibration has been performed allow a full calibration cycle
;(2 hrs) to complete to place the zero and span values in
;EEPROM memory before disabling the calibration routine. The
;zero and span values for each interpolator can also be entered
;directly into memory in HEX and saved manually to EEPROM
;providing fixed calibration values for use at restart. In
;the manual calibration mode the last zero and span values
;entered by the user or determined by the autocal routine
;and stored are used as fixed constants in the correction
;routine.
;  During operation in automatic calibration mode the peak
;values are read and the peak detectors are reset every 3600
;samples or hourly with 1PPS inputs. The peak values establish
;the hardware zero and span and these values are stored to
;EEPROM and used during the next calibration period to correct
;the data mathematically for an exact span of 0 - 400 even when
;the hardware span or offset is slightly different due to
;temperature variations and component aging. After correction
;processing the interpolator data has a fixed range from 0-400
;representing one clock cycle. If the interpolator data is
;outside the zero or span limits during correction processing
;the data is limited to 0 - 400 and a status bit is set to
;provide an indication of the interpolator limit condition.
;During startup a zero or full scale ADC output can occur,
;so a 1-minute delay is added before the peak detectors are
;reset and the first calibration cycle begins. The correction
;takes the form expressed in the following equations:

;Imin = Data offset from zero
;Data - Imin = offset corrected data
;Imax - Imin = actual hardware span
; 400 = Desired Data Span

;Corrected Interpolator Value = offset corrected data *
;desired data span / actual hardware span
;                        OR
;((Data - Imin) * 400) / (Imax - Imin) = Corrected Data

;Total Delay = ((Counter Value * 400) + Start Delay) - Stop Delay

;  To determine the time delay the counter value is multiplied
;by the interpolator data span, the corrected start delay is
;added, and the corrected stop delay is subtracted from the
;result before display. Offset and gain drift with temperature
;occurs similarly in both interpolators, so the temperature
;variations in the interpolators tend to cancel out when the
;stop value is subtracted from the start value in calculating
;the total delay. The 32-bit data is converted to 10-digit BCD
;and displayed immediately after each sample. In this evaluation
```

```
;version no overflow detection is performed on the data before
;display.
;  The Simple PICTIC allows two different counter configurations
;to be explored. For timebase rates up to 16.7 MHz the 16F688
;TMR1 counter can be used in gated mode with the timebase directly
;clocking TMR1. With slower timebase rates the PICTIC can be
;implemented using 7 IC packages. For faster timebase rates
;a 4-bit binary prescaler can be installed that feeds TMR1 in
;the PIC allowing up to 66 MHZ timebase rates to be used. This
;demonstrates how the system can be implemented on other PIC
;devices without a gated TMR1 capability.
;  Compromises were made in the Simple PICTIC design to reduce
;cost and complexity but these were considered acceptable
;to achieve a low cost, high accuracy time interval counter.
;Using multiple analog reads in the interpolator increases the
;processing time needed between samples reducing the maximum
;continuous sample rate. The input stages are cross connected
;for a fixed start before stop requiring a small delay between
;the start and stop events. When the delay exceeds 980ms with
;1PPS inputs the counter reset may not occur in time before the
;next start input, resulting in errors in the reading or
;skipped samples. Sufficient delay between the end of a sample
;and the beginning of the next sample is required to allow time
;for the PIC to read and reset the counter and interpolators
;between samples.
;  Various data types can be toggled on or off for display during
;the update providing a view of the interpolators and autocal
;routines in action. A 4-bit status message can be enabled to
;display the configuration and interpolator overflow status.
;The first bit indicates normal (0) or prescaler (1) mode,
;the second bit indicates normal (0) or high gain (1) mode,
;the third bit indicates autocal off (0)or on (1), and the
;fourth bit indicates (1) when an interpolator value exceeded
;the limit.
;  Displaying only the 10-digit counter data the maximum update
;rate is about 80 Hz to allow sufficient time to transmit the
;data at 9600 baud between samples. When additional data types
;are displayed the maximum update rate must be reduced
;accordingly. Tie RTS and CTS (pins 7 & 8) together at the PC
;or through the cable or serial commands won't function
;correctly. An RS232 conditional flag is included but can be
;disabled for inverted TXD from the 16F688 PIC. A transistor
;inverter for RXD and inverted TXD from the PIC can be used with
;a short connecting cable to provide full duplex communication
;with a PC at TTL levels. A 10 MHz PICTIC with 250ps resolution
;for GPS monitoring can be implemented with just 6 IC packages
;for lowest cost using inverted TXD in your own design.
;  For initial evaluation a 10 MHz XO timebase with unity gain
;interpolator buffer is recommended. Once initial calibration
;has been performed and operation of the interpolator and autocal
;routines are fully understood experimentation with higher gains,
;higher frequencies, and high stability external timebases will
;provide greater insight into the capabilities of the deceptively
;Simple PICTIC interpolating time interval counter design.

;*********************************************************************
;                          Default EEPROM Values
;*********************************************************************
;The EEPROM should be pre-loaded as shown below during the
;PIC programming process.

;Zero = 250, Mid = 450, Span = 400, Time = 3600,
;Direct mode, AutoCal enabled, Print final result only

;00-07  00 FA 01 C2 01 90 00 FA
;08-0F  01 C2 01 90 0E 10 20 02

;*********************************************************************
;                           I/O Pin Assignments:
;*********************************************************************
```

```
;    Register Bit      Pin     Function
;      PORTA   0       (13)    A/D Ch 0 In
;              1       (12)    A/D Ch 1 In
;              2       (11)    Interrupt In
;              3       ( 4)    Spare
;              4       ( 3)    Gate In
;              5       ( 2)    D3 In (TMR1 Clk)
;      PORTC   0       (10)    D0 In
;              1       ( 9)    D1 In
;              2       ( 8)    D2 In
;              3       ( 7)    Reset Out
;              4       ( 6)    USART Async TX Out
;              5       ( 5)    USART Async RX In

;*********************************************************************
;                       Serial Command Functions:
;*********************************************************************
; The Expanded Command set allows process variables to
; be printed or modified by serial port command.

; Basic Command Set:

;     Command                   Function
;       #             Aborts current command or data entry

;      @@A            Calibrate A Interpolator Commands
;      @@B            Calibrate B Interpolator Commands
;      @@C            Calibration Commands
;      @@D            Select display parameters
;      @@M            Operating Mode Commands
;      @@P            Print Commands - once to serial TX
;      @@R            Run Command - Resets and starts counter
;      @@S            Stop Command - Stops counter
;      @@U            Update EEPROM Calibration Values

; Expanded Command Set:

;      @@A            Calibrate A Interpolator Commands
;      @@Ac           Set Ch A Center Value (xxx HEX)
;      @@As           Set Ch A Span Value (xxx HEX)
;      @@Az           Set Ch A Zero Value    (xxx HEX)

;      @@B            Calibrate B Interpolator Commands
;      @@Bc           Set Ch B Center Value (xxx HEX)
;      @@Bs           Set Ch B Span Value (xxx HEX)
;      @@Bz           Set Ch B Zero Value    (xxx HEX)

;      @@C            Calibration Commands
;      @@Cd           Disable Autocal Mode, Manual Cal Only
;      @@Ce           Enable Autocal Mode
;      @@Cr           Reset Autocal Peak Setectors
;      @@Ct           Set Calibration Time in Samples (xxxx HEX)
;      @@Cp           Print Calibration Values and Time in HEX

;      @@D            Display parameters
;      @@Dc           Toggle Display Calibration Values in BCD
;      @@Dd           Toggle Display 10-Digit Time Delay in BCD
;      @@Di           Toggle Display Corrected Start and Stop in BCD
;      @@Dp           Toggle Display Peak Detector Values
;      @@Dr           Toggle Display Raw Interpolator Data in BCD
;      @@Ds           Toggle Display Status Digits
;      @@Dt           Toggle Display TIC Counter Data in BCD
;      @@Dv           Toggle Display Interpolated Value in BCD

;      @@M            Operating Mode Commands
;      @@Md           Direct Input to Counter
;      @@Mh           High Gain Mode (Gain = 800)
;      @@Mn           Normal Gain Mode (Gain = 400)
;      @@Mp           Prescaler Input to Counter

;      @@P            Print parameters:
```

```
;       @@Pc            Print Calibration Data in BCD
;       @@Pi            Print Interpolator Data in BCD
;       @@Ps            Print Status Digits
;       @@Pt            Print Calibration Time in BCD

;*****************************************************************
;                           Change History:
;*****************************************************************

; 12/20/2008 Simple PICTIC Release Code                     (Rev 1.00)

;*****************************************************************
;                       Conditional Assembly Flags
;*****************************************************************

#define RS232          ;RS-232 converter, normal TXD.

;*****************************************************************
;                       Define Storage Locations
;*****************************************************************
; Maintain registers in sequence H->L for indexed ops!

        CBLOCK  0x20                    ;Bank 0 Registers - 96 max, 86 used
                TX_BUF                  ;the next char to be xmitted
                RX_BUF                  ;the last char received
                flg0                    ;flag bit storage
                cHdr0,cHdr1             ;command header flags
                Ha,Ma,La,Sa,SSa         ;40 bit math reg A
                Hb,Mb,Lb,Sb,SSb         ;40 bit math reg B
                bCnt,bTst               ;Binary-to-BCD regs
                BD4,BD3,BD2,BD1,BD0     ;must stay in sequence BD4 -> BD0
                Hbyte,Mbyte,Lbyte       ;print register storage
                Sbyte
                Hdata,Mdata             ;phase data from counter
                Ldata,Sdata,SSdata
                HPS,MPS,LPS             ;print data storage
                SPS,SSPS
                HOFctr,LOFctr           ;overflow counter
                btmp,btmp2              ;background temp regs
                BytCt,Dadr              ;serial data entry regs
                Htmp,Ltmp               ;16-bit temp storage
                HcalT,LcalT             ;autocal timers
                temp                    ;interrupt temporary storage
                Padr                    ;EEPROM address register
                TSS                     ;temp serial storage
                HriA,LriA               ;interpolator A reg
                HriB,LriB               ;interpolator B reg
                HiaM,LiaM               ;Ch A Max count
                HibM,LibM               ;Ch B Max count
                HiaO,LiaO               ;Ch A Min count
                HibO,LibO               ;Ch B Min count
                Hstart,Lstart           ;Start delay
                Hstop,Lstop             ;Stop delay
                HIval,LIval             ;Interpolator value

;Registers saved and restored to EEPROM - Keep in order!

                HiaZ,LiaZ               ;Ch A zero
                HiaC,LiaC               ;Ch A Center
                HiaS,LiaS               ;Ch A span
                HibZ,LibZ               ;Ch B zero
                HibC,LibC               ;Ch B Center
                HibS,LibS               ;Ch B span
                Hcal,Lcal               ;autocal time storage
                DMB                     ;Display Mode Bits
                CMB                     ;Command bits
        ENDC

;Registers 70-7F common in all pages in 16F688 so stack is stored here
;NOTE: If not common in PIC used, don't use page 1 variables in the
;background without disabling interrupts as background routine could
```

```
;be interrupted on either page and W may not get saved properly in
;W_TEMP on interrupt of a page 1 routine.

        CBLOCK  0x7d
                W_TEMP          ;temp storage for W on interrupt
                STAT_TEMP       ;ditto for status reg
                FSR_TEMP        ;ditto for FSR
        ENDC

;*****************************************************************
;                       Hardware Bit Assignments:
;*****************************************************************
;Hardware bit flags used to simplify reassignment of pins

#define t1e     T1CON,TMR1ON    ;Timer 1 Enable
#define t1f     PIR1,TMR1IF     ;Timer 1 Overflow Flag
#define adF     ADCON0,1        ;A/D Converter start/done flag
#define adCH    ADCON0,2        ;A/D Converter Channel
#define Zflag   STATUS,Z        ;Zero Flag
#define Cflag   STATUS,C        ;Carry Flag
#define CRST    PORTC,3         ;Reset Out

;*****************************************************************
;                          Flag Bit Assignments:
;*****************************************************************
;Flag bits used to optimize memory usage

; CMB:
; B0=HoldF, B1=ACM, B2=HGM, B3=PSM
; B4=N/U, B5=N/U, B6=N/U, B7=N/U

; DMB:
; B0=TDE, B1=IPE, B2=CVE, B3=DCV
; B4=DID, B5=DTD, B6=PSE, B7=DPE

; flg0:
; B0=BrdyF, B1=LoByt, B2=Nrdy, B3=DUF
; B4=FpF, B5=Drdy, B6=negFlag, B7=Ilmt

;*****************************************************************

#define HoldF   CMB,0           ;Hold flag
#define ACM     CMB,1           ;Autocal Mode Flag
#define HGM     CMB,2           ;High Gain Mode Flag
#define PSM     CMB,3           ;Prescaler Mode Flag
#define TDE     DMB,0           ;TIC Direct Print Enable
#define IPE     DMB,1           ;Raw Interpolator Print Enable
#define CVE     DMB,2           ;Calibration Values Print Enable
#define DCV     DMB,3           ;Corrected Start and Stop Print Enable
#define DID     DMB,4           ;Interpolated Value Print Enable
#define DTD     DMB,5           ;10-Digit Time Delay Print Enable
#define PSE     DMB,6           ;Status Print Enable
#define DPE     DMB,7           ;Peak Data Print Enable
#define BrdyF   flg0,0          ;Byte Ready Flag
#define LoByt   flg0,1          ;Low Byte Flag
#define Nrdy    flg0,2          ;Not Ready Flag
#define DUF     flg0,3          ;Data Update Flag
#define FpF     flg0,4          ;First Cal Flag
#define Drdy    flg0,5          ;Print Data Ready Flag
#define negFlag flg0,6          ;Neg Flag
#define Ilmt    flg0,7          ;I Limit Flag

;*****************************************************************
;                         Command Header Flags:
;*****************************************************************

#define Hdr0 cHdr0,0            ;Header 0 Flag (@   )
#define Hdr1 cHdr0,1            ;Header 1 Flag (@@  )
#define Hdr2 cHdr0,2            ;Header 2 Flag (@@A )
#define Hdr3 cHdr0,3            ;Header 3 Flag (@@B )
#define Hdr4 cHdr0,4            ;Header 4 Flag (@@C )
```

```
#define Hdr5 cHdr0,5          ;Header 5 Flag (@@D )
#define Hdr6 cHdr0,6          ;Header 6 Flag (@@M )
#define Hdr7 cHdr0,7          ;Header 7 Flag (@@P )
#define Hdr8 cHdr1,0          ;Header 8 Flag (Nrtn)

;*********************************************************************
;                          Initialization
;*********************************************************************
;set interrupt vector and start of code

        org     0             ;initialize code
        nop                   ;required for the ICD
        clrf    STATUS        ;ensure we are at page 0
        clrf    PCLATH        ;ensure bank bits are cleared
        goto    start
        org     4             ;interrupt routine
        movwf   W_TEMP        ;"push" instructions
        swapf   STATUS,W      ;swapf affects NO status bits
        bcf     STATUS,RP0    ;select page 0
        movwf   STAT_TEMP     ;save STATUS
        movf    FSR,W
        movwf   FSR_TEMP      ;save FSR
        bcf     INTCON,INTF   ;clear interrupt, else int again immediately!
        goto    int_srv

;InitCon initializes controller ports and registers

start   call    InitCon       ;init regs, get constants from prom
        bsf     CRST          ;set reset to enable
        bsf     t1e           ;enable TMR1
        bsf     INTCON,GIE    ;enable interrupts

;*********************************************************************
;                        Background Routine
;*********************************************************************
;Handles communication and print tasks between interrupts

Bkgnd   btfsc   t1f           ;if TMR1 overflow set
        call    UpdOF         ;update TMR1 overflow counter
        call    Rx232         ;check for character received
        btfsc   BrdyF         ;if byte ready
        call    CmdProc       ;Process command
        btfsc   DUF           ;data update flag set,
        call    D2prom        ;update PROM with new data
        btfss   Drdy          ;if data ready flag set
        goto    Bkgnd
        bcf     Drdy          ;clear data ready flag
        btfsc   HoldF         ;in hold mode
        goto    Bkgnd         ;just loop
        btfsc   TDE           ;if TIC display enabled
        call    prtData       ;print counter data
        btfsc   IPE           ;if enabled
        call    prtID         ;print interpolator data
        btfsc   DPE           ;if enabled
        call    prtPD         ;print peak detector data
        btfsc   CVE           ;if enabled
        call    prtCD         ;print calibration data
        btfsc   DCV           ;if enabled
        call    prtCV         ;print corrected start and stop
        btfsc   DID           ;if enabled
        call    prtIV         ;print interpolated value
        btfsc   DTD           ;if enabled
        call    prtPS         ;print time delay
        btfsc   PSE           ;if status display enabled
        call    prtSt         ;print status
        call    TxCrLf        ;print EOL
        bcf     Ilmt          ;clear I limit flag
        goto    Bkgnd
```

```
;*********************************************************************
;                        Interrupt Service Routine
;                           Copyright © 2008
;                          Richard H McCorkle
;*********************************************************************
;Calculate counter correction from interpolator and add to data
;Full clock cycles in TMR1, partial clock cycle determined from
;interpolators by (((start count - start zero) * Ires) / start span)
; - (((stop count - stop zero) * Ires) / stop span) for corrected
;interpolator value.

int_srv call    readctr         ;get data values, reset counter
        call    Acal            ;read cal values
        call    A2strt          ;A to start
        call    B2stop          ;B to stop
        call    data2B          ;Counter to B
        call    Res2A           ;Resolution to A
        call    MB40X16         ;multiply B * A
        call    Aclr
        movf    Hstart,W        ;start to A
        movwf   Sa
        movf    Lstart,W
        movwf   SSa
        call    addAL           ;add start delay to counter
        call    Aclr
        movf    Hstop,W         ;stop to A
        movwf   Sa
        movf    Lstop,W
        movwf   SSa
        call    subAL           ;subtract stop delay
        call    B2PS            ;put value in print register
        bsf     Drdy            ;set data ready flag
        btfss   DID             ;if display interpolated value set
        goto    pop
        call    Aclr            ;put stop in top bits of A
        movf    Hstop,W
        movwf   Ha
        movf    Lstop,W
        movwf   Ma
        call    Bclr            ;put start in top bits of B
        movf    Hstart,W
        movwf   Hb
        movf    Lstart,W
        movwf   Mb
        call    subA            ;Start - Stop = interpolated value in B
        movf    Hb,W            ;save value for print
        movwf   HIval
        movf    Mb,W
        movwf   LIval
pop     bcf     STATUS,RP0      ;bank 0
        movf    FSR_TEMP,W      ;restore FSR
        movwf   FSR
        swapf   STAT_TEMP,W     ;restore STATUS
        movwf   STATUS
        swapf   W_TEMP          ;set status bits
        swapf   W_TEMP,W        ;restore W
        retfie                  ;return from interrupt

;*********************************************************************
;                               Subroutines
;*********************************************************************
;update TMR1 overflow counter

UpdOF   incf    LOFctr          ;add 1 to OFctr
        btfsc   Zflag
        incf    HOFctr
        bcf     t1f             ;clear TMR1 overflow flag
        return

;*********************************************************************
;clear time counter
```

```
clrTC   bcf     t1e             ;stop TMR1
        bcf     CRST            ;clear the interpolator
        bsf     CRST
        clrf    TMR1L           ;clear TMR1
        clrf    TMR1H
        clrf    HOFctr          ;clear overflow counter
        clrf    LOFctr
        bcf     t1f             ;clear TMR1 overflow flag
        bcf     Ilmt            ;clear I limit flag
        bsf     t1e             ;start TMR1
        goto    clrSC           ;init peak detectors

;*********************************************************************
;                       Read PICTIC Counter
;                        Copyright © 2008
;                       Richard H McCorkle
;*********************************************************************
;Full clock cycles in prescaler, TMR1, and OFctr, partial
;clock cycles determined from time to voltage interpolators.

readctr bcf     t1e             ;stop TMR1
        movf    TMR1H,W         ;move TMR1 to Data
        movwf   Ldata
        movf    TMR1L,W
        movwf   Sdata
        btfss   PSM             ;if prescaler mode
        goto    $ + 6
        swapf   PORTC,W         ;read prescaler
        andlw   0x70            ;mask unused bits
        movwf   SSdata
        btfss   PORTA,5         ;if MSB = 1
        bsf     SSdata,7        ;set high bit
        btfsc   t1f             ;if TMR1 overflow set
        call    UpdOF           ;update TMR1 overflow counter
        movf    LOFctr,W        ;move LOFctr to Mdata
        movwf   Mdata
        movf    HOFctr,W        ;move HOFctr to Hdata
        movwf   Hdata
        bsf     adF             ;set convert flag to start conversion
        btfsc   adF             ;wait till conversion done
        goto    $ - 1
        bsf     adCH            ;select channel 1
        bsf     STATUS,RP0      ;select bank 1
        movf    ADRESL,W        ;get A/D result in W
        bcf     STATUS,RP0      ;select bank 0
        movwf   LriA            ;save raw interpolator A
        movf    ADRESH,W
        movwf   HriA
        goto    $ + 1           ;wait 5us for input to charge
        goto    $ + 1           ;(longer in other PIC types)
        bsf     adF             ;set convert flag to start conversion
        btfsc   adF             ;wait till conversion done
        goto    $ - 1
        bcf     adCH            ;select channel 0
        bsf     STATUS,RP0      ;select bank 1
        movf    ADRESL,W        ;get A/D result in W
        bcf     STATUS,RP0      ;select bank 0
        movwf   LriB            ;save raw interpolator B
        movf    ADRESH,W
        movwf   HriB
        bcf     CRST            ;clear the interpolators
        bsf     CRST            ;and logic
        clrf    TMR1L           ;clear TMR1
        clrf    TMR1H
        bsf     t1e             ;start TMR1
        clrf    HOFctr          ;clear OFctr
        clrf    LOFctr
        bcf     t1f             ;clear overflow flag
        movlw   0x08            ;shift data 8 bits right
        btfsc   PSM             ;if prescaler mode
```

```
        movlw   0x04            ;shift 4 bits
        bcf     Cflag           ;to 40-bit counter value
        rrf     Hdata
        rrf     Mdata
        rrf     Ldata
        rrf     Sdata
        rrf     SSdata
        addlw   0xff
        btfss   Zflag
        goto    $ - 8
        return

;*********************************************************************
;                           AutoCal Routine
;                         Copyright © 2008
;                        Richard H McCorkle
;*********************************************************************
;peak detector routines determine min and max interpolator values
;for calculating actual interpolator zero and span. Min count is
;lowest resolution or zero, max count is span as no sample can be
;< 1 or > 2 clock cycles with this design.

Acal    movf    LriA,W          ;put raw A value in temp
        movwf   Ltmp
        movf    HriA,W
        movwf   Htmp
        movf    LiaC,W          ;subtract center value
        subwf   Ltmp
        btfss   Cflag
        decf    Htmp
        movf    HiaC,W
        subwf   Htmp
        btfsc   Htmp,7          ;if sample < center
        goto    ACJ1            ;do min test
        movf    LriA,W          ;put raw A value in temp
        movwf   Ltmp
        movf    HriA,W
        movwf   Htmp
        movf    LiaM,W          ;subtract Ch A Max from temp
        subwf   Ltmp
        btfss   Cflag
        decf    Htmp
        movf    HiaM,W
        subwf   Htmp
        btfsc   Htmp,7          ;if positive, value is > Max
        goto    ACJ2
        movf    HriA,W          ;save raw A value as new Ch A Max
        movwf   HiaM
        movf    LriA,W
        movwf   LiaM
        goto    ACJ2
ACJ1    movf    HiaO,W          ;put Ch A offset in temp
        movwf   Htmp
        movf    LiaO,W
        movwf   Ltmp
        movf    LriA,W          ;subtract raw A value from offset
        subwf   Ltmp
        btfss   Cflag
        decf    Htmp
        movf    HriA,W
        subwf   Htmp
        btfsc   Htmp,7          ;if positive, value is < offset
        goto    ACJ2
        movf    HriA,W          ;save value as new Ch A offset
        movwf   HiaO
        movf    LriA,W
        movwf   LiaO
ACJ2    movf    LriB,W          ;put raw B value in temp
        movwf   Ltmp
        movf    HriB,W
        movwf   Htmp
```

```
        movf    LibC,W          ;subtract center value
        subwf   Ltmp
        btfss   Cflag
        decf    Htmp
        movf    HibC,W
        subwf   Htmp
        btfsc   Htmp,7          ;if sample < center
        goto    ACJ3            ;do min test
        movf    LriB,W          ;put raw B value in temp
        movwf   Ltmp
        movf    HriB,W
        movwf   Htmp
        movf    LibM,W          ;subtract Ch B Max from temp
        subwf   Ltmp
        btfss   Cflag
        decf    Htmp
        movf    HibM,W
        subwf   Htmp
        btfsc   Htmp,7          ;if positive, value is > Max
        goto    ACJ4
        movf    HriB,W          ;save raw B value as new Ch B Max
        movwf   HibM
        movf    LriB,W
        movwf   LibM
        goto    ACJ4
ACJ3    movf    HibO,W          ;put Ch B offset in temp
        movwf   Htmp
        movf    LibO,W
        movwf   Ltmp
        movf    LriB,W          ;subtract raw B value from temp
        subwf   Ltmp
        btfss   Cflag
        decf    Htmp
        movf    HriB,W
        subwf   Htmp
        btfsc   Htmp,7          ;if positive, value is < offset
        goto    ACJ4
        movf    HriB,W          ;save raw B value as new Ch B offset
        movwf   HibO
        movf    LriB,W
        movwf   LibO
ACJ4    movlw   0xff            ;dec cal timer
        addwf   LcalT
        btfss   Cflag
        addwf   HcalT
        movf    HcalT,W         ;if not zero return
        btfss   Zflag
        return
        movf    LcalT,W
        btfss   Zflag
        return
        btfss   FpF             ;don't update first pass
        goto    $ + 3
        bcf     FpF
        goto    clrSC
        btfss   ACM             ;if autocal flag set
        goto    clrSC
        movf    HiaO,W          ;save new offset as zero values
        movwf   HiaZ
        movf    LiaO,W
        movwf   LiaZ
        movf    HibO,W
        movwf   HibZ
        movf    LibO,W
        movwf   LibZ
        movf    HiaM,W          ;save Max values in span
        movwf   HiaS
        movf    LiaM,W
        movwf   LiaS
        movf    HibM,W
        movwf   HibS
```

```
        movf    LibM,W
        movwf   LibS
        movf    LiaO,W          ;max - offset = new span values
        subwf   LiaS
        btfss   Cflag
        decf    HiaS
        movf    HiaO,W
        subwf   HiaS
        movf    LibO,W
        subwf   LibS
        btfss   Cflag
        decf    HibS
        movf    HibO,W
        subwf   HibS
        movf    HiaS,W          ;A span to tmp
        movwf   Htmp
        movf    LiaS,W
        movwf   Ltmp
        bcf     Cflag           ;/2
        rrf     Htmp
        rrf     Ltmp
        movf    LiaZ,W          ;add offset
        addwf   Ltmp
        btfsc   Cflag
        incf    Htmp
        movf    HiaZ,W
        addwf   Htmp
        movf    Htmp,W          ;save A center
        movwf   HiaC
        movf    Ltmp,W
        movwf   LiaC
        movf    HibS,W          ;B span to tmp
        movwf   Htmp
        movf    LibS,W
        movwf   Ltmp
        bcf     Cflag           ;/2
        rrf     Htmp
        rrf     Ltmp
        movf    LibZ,W          ;add offset
        addwf   Ltmp
        btfsc   Cflag
        incf    Htmp
        movf    HibZ,W
        addwf   Htmp
        movf    Htmp,W          ;save B center
        movwf   HibC
        movf    Ltmp,W
        movwf   LibC
        bsf     DUF             ;set flag to update values in EEPROM
clrSC   movf    HiaC,W          ;init peak detectors to center
        movwf   HiaM
        movwf   HiaO
        movf    LiaC,W
        movwf   LiaM
        movwf   LiaO
        movf    HibC,W
        movwf   HibM
        movwf   HibO
        movf    LibC,W
        movwf   LibM
        movwf   LibO
        movf    Hcal,W          ;reload cal timer
        movwf   HcalT
        movf    Lcal,W
        movwf   LcalT
        return

;*****************************************************************
;                   Interpolator Correction Routines
;                          Copyright © 2008
;                         Richard H McCorkle
```

```
;*********************************************************************
;Interpolator A to start delay

A2strt  movf    LriA,W          ;save Ch A as start count
        movwf   Lstart
        movf    HriA,W
        movwf   Hstart
        movf    LiaZ,W          ;remove zero offset
        subwf   Lstart          ;from start count
        btfss   Cflag
        decf    Hstart
        movf    HiaZ,W
        subwf   Hstart
        btfss   Hstart,7        ;if neg, data < zero
        goto    $ + 5
        clrf    Lstart          ;make data zero
        clrf    Hstart
        bsf     Ilmt            ;set I limit flag
        return
        call    Bclr            ;move start to B
        movf    Lstart,W
        movwf   Mb
        movf    Hstart,W
        movwf   Hb
        call    Res2A           ;Resolution to A
        call    MB16X16         ;multiply B * A
        call    Aclr
        movf    LiaS,W          ;actual span in A
        movwf   Ma
        movf    HiaS,W
        movwf   Ha
        call    DB24X16         ;divide B / A
        movf    Lb,W            ;Save corrected start
        movwf   Lstart
        movf    Mb,W
        movwf   Hstart
        movwf   Hb              ;move start high in B
        movf    Lb,W
        movwf   Mb
        clrf    Lb
        call    Res2A           ;Resolution to A
        call    subA            ;sub max from count
        btfsc   Hb,7            ;if pos data > limit
        return
        call    Res2A           ;Resolution limit to A
        movf    Ma,W            ;save limit as start
        movwf   Lstart
        movf    Ha,W
        movwf   Hstart
        bsf     Ilmt            ;set I limit flag
        return

;*********************************************************************
;Interpolator B to stop delay

B2stop  movf    LriB,W          ;save Ch B as stop count
        movwf   Lstop
        movf    HriB,W
        movwf   Hstop
        movf    LibZ,W          ;remove zero offset
        subwf   Lstop           ;from stop count
        btfss   Cflag
        decf    Hstop
        movf    HibZ,W
        subwf   Hstop
        btfss   Hstop,7         ;if neg, data < zero
        goto    $ + 5
        clrf    Lstop           ;make data zero
        clrf    Hstop
        bsf     Ilmt            ;set I limit flag
        return
```

```
        call    Bclr            ;move stop to B
        movf    Lstop,W
        movwf   Mb
        movf    Hstop,W
        movwf   Hb
        call    Res2A           ;Resolution to A
        call    MB16X16         ;multiply B * A
        call    Aclr
        movf    LibS,W          ;actual span in A
        movwf   Ma
        movf    HibS,W
        movwf   Ha
        call    DB24X16         ;divide B * A
        movf    Lb,W            ;Save corrected stop
        movwf   Lstop
        movf    Mb,W
        movwf   Hstop
        movwf   Hb              ;move stop high in B
        movf    Lb,W
        movwf   Mb
        clrf    Lb
        call    Res2A           ;Resolution to A
        call    subA            ;sub max from count
        btfsc   Hb,7            ;if pos data > limit
        return
        call    Res2A           ;Resolution limit to A
        movf    Ma,W            ;save limit as stop
        movwf   Lstop
        movf    Ha,W
        movwf   Hstop
        bsf     Ilmt            ;set I limit flag
        return

;*********************************************************************
;                          Data Movement Routines
;*********************************************************************
;Interpolator resolution to A - set for 400/800 or as required

Res2A   call    Aclr
        btfsc   HGM
        goto    $ + 6
        movlw   0x01            ;400 in normal mode
        movwf   Ha
        movlw   0x90
        movwf   Ma
        return
        movlw   0x03            ;800 in high gain mode
        movwf   Ha
        movlw   0x20
        movwf   Ma
        return

;*********************************************************************
;40-bit B to A

B2A     movf    Hb,W            ;B to A 40-bit data
        movwf   Ha
        movf    Mb,W
        movwf   Ma
        movf    Lb,W
        movwf   La
        movf    Sb,W
        movwf   Sa
        movf    SSb,W
        movwf   SSa
        return

;*********************************************************************
;40-bit data to B

data2B  movf    SSdata,W        ;40-bit data to 40-bit B
```

```
        movwf   SSb
        movf    Sdata,W
        movwf   Sb
        movf    Ldata,W
        movwf   Lb
        movf    Mdata,W
        movwf   Mb
        movf    Hdata,W
        movwf   Hb
        return

;*****************************************************************
;B to 40-bit print storage

B2PS    movf    SSb,W           ;40-bit B to 40-bit print storage
        movwf   SSPS
        movf    Sb,W
        movwf   SPS
        movf    Lb,W
        movwf   LPS
        movf    Mb,W
        movwf   MPS
        movf    Hb,W
        movwf   HPS
        return

;*****************************************************************
;                               Math Routines
;*****************************************************************
;24-bit & 40-bit math routines (adapted from AN611)

Aclr    clrf    Ha              ;clear A
        clrf    Ma
        clrf    La
        clrf    Sa
        clrf    SSa
        return

;*****************************************************************

addAL   movf    SSa,W           ;A + B -> B  (40 bits)
        addwf   SSb             ;add low byte
        btfsc   Cflag           ;add in carry if necessary
        call    add3
        movf    Sa,W
        addwf   Sb              ;add mid bytes
        btfsc   Cflag
        call    add2
addA    movf    La,W            ;A + B -> B  (24 bits)
        addwf   Lb              ;add mid bytes
        btfsc   Cflag
        call    add1
        movf    Ma,W
        addwf   Mb              ;add mid bytes
        btfsc   Cflag
        incf    Hb
        movf    Ha,W
        addwf   Hb              ;add hi byte
        return
add3    incf    Sb              ;skip chain
        btfsc   Zflag
add2    incf    Lb
        btfsc   Zflag
add1    incf    Mb
        btfsc   Zflag
        incf    Hb
        return

;*****************************************************************
cmpA    comf    La              ;24-bit 2s complement of A -> A
        comf    Ma
```

```
        comf    Ha
        goto    add5

;*****************************************************************

cmpAL   comf    SSa             ;40-bit 2s complement of A -> A
        comf    Sa              ;invert all the bits in A
        comf    La
        comf    Ma
        comf    Ha
        incf    SSa             ;add one to A
        btfsc   Zflag           ;skip chain
add6    incf    Sa
        btfsc   Zflag
add5    incf    La
        btfsc   Zflag
add4    incf    Ma
        btfsc   Zflag
        incf    Ha
        return

;*****************************************************************

Bclr    clrf    Hb              ;clear B
        clrf    Mb
        clrf    Lb
        clrf    Sb
        clrf    SSb
        return

;*****************************************************************
cmpB    comf    Lb              ;24-bit 2s complement of B -> B
        comf    Mb
        comf    Hb
        goto    add2

;*****************************************************************
cmpBL   comf    SSb             ;40-bit 2s complement of B -> B
        comf    Sb
        comf    Lb
        comf    Mb
        comf    Hb
        incf    SSb
        btfsc   Zflag
        goto    add3
        return

;*****************************************************************
;Divide unsigned 24 bit number in B by unsigned 16 bit divisor in
;A on call with result as 24 bit unsigned number in B

;Adaption of code 8-July-2000 by Nikolai Golovchenko, see
;http://www.piclist.com/techref/microchip/math/div/16by8lzf-ng.htm

DB24X16 movlw   D'24'           ;24-bit data
        movwf   temp            ;loop counter
        clrf    Ltmp            ;use tmp register as temporary
        clrf    Htmp
        bcf     Cflag
DB3     rlf     Lb              ;shift next msb into temporary
        rlf     Mb
        rlf     Hb
        rlf     Ltmp
        rlf     Htmp
        movf    Ltmp,W          ;save temp in low A for restore
        movwf   SSa
        movf    Htmp,W
        movwf   Sa
        bcf     negFlag         ;clear borrow flag
        movf    Ma,W            ;subtract divisor from temporary
        subwf   Ltmp
```

```
        btfsc   Cflag           ;if borrow
        goto    $ + 5
        movlw   0xff            ;dec Htmp
        addwf   Htmp
        btfss   Cflag           ;if borrow
        bsf     negFlag         ;set negFlag
        movf    Ha,W
        subwf   Htmp
        btfss   Cflag           ;if borrow
        bsf     negFlag         ;set negFlag
        bcf     Cflag
        btfss   negFlag         ;if no borrow, set Cflag
        bsf     Cflag           ;carry is the next bit of result
        btfss   negFlag         ;if borrow,
        goto    $ + 5
        movf    SSa,W           ;restore temp
        movwf   Ltmp
        movf    Sa,W
        movwf   Htmp
        decfsz  temp            ;repeat 24 times to find result
        goto    DB3
        rlf     Lb              ;shift last bit into B
        rlf     Mb
        rlf     Hb
        return

;*****************************************************************
;multiply unsigned 16-bit number in B by unsigned
;16-bit number in A into 24-bit result in B.

MB16X16 movf    Ha,W            ;move A to tmp
        movwf   Htmp
        movf    Ma,W
        movwf   Ltmp
        call    Aclr            ;move B to low A
        movf    Hb,W
        movwf   Ma
        movf    Mb,W
        movwf   La
        call    Bclr            ;clear B (result)
        movlw   D'16'           ;multiply tmp x A -> B
        movwf   temp            ;bit counter (16 bit multiplier)
mlp0    bcf     Cflag           ;left-shift 24 bit B reg
        rlf     Lb              ;rotate result left (*2)
        rlf     Mb
        rlf     Hb
        rlf     Ltmp
        rlf     Htmp
        btfsc   Cflag           ;if bit is clear then no add
        call    addA            ;add A to B
        decfsz  temp            ;loop til done
        goto    mlp0
        return                  ;return with correction in B

;*****************************************************************
;multiply unsigned 40-bit number in B by unsigned
;16-bit number in A into 40-bit result in B.

MB40X16 movf    Ha,W            ;move A to tmp
        movwf   Htmp
        movf    Ma,W
        movwf   Ltmp
        call    B2A             ;move B to A
        call    Bclr            ;clear B (result)
        movlw   D'16'           ;multiply tmp x A -> B
        movwf   temp            ;bit counter (16 bit multiplier)
mlp1    bcf     Cflag           ;left-shift 40 bit B reg
        rlf     SSb             ;rotate result left (*2)
        rlf     Sb
        rlf     Lb
        rlf     Mb
```

```
        rlf     Hb
        rlf     Ltmp
        rlf     Htmp
        btfsc   Cflag           ;if bit is clear then no add
        call    addAL           ;add A to B
        decfsz  temp            ;loop til done
        goto    mlp1
        return                  ;return with correction in B

;*****************************************************************

subA    call    cmpA            ;B - A -> B (24 bits)
        goto    addA            ;addA returns to caller

;*****************************************************************

subAL   call    cmpAL           ;B - A -> B (40 bits)
        goto    addAL           ;addAL returns to caller

;*****************************************************************
;                        Hex Print Routines
;                         Copyright © 2006
;                        Richard H McCorkle
;*****************************************************************
;Provides display of constant data in the same Hex format used
;during data entry.

;print the zero, center, and span cal data in HEX (12 bits)

hpCD    movf    HiaZ,W          ;IA Zero
        movwf   Mbyte
        movf    LiaZ,W
        movwf   Lbyte
        call    hp12
        movf    HiaC,W          ;IA Center
        movwf   Mbyte
        movf    LiaC,W
        movwf   Lbyte
        call    hp12
        movf    HiaS,W          ;IA Span
        movwf   Mbyte
        movf    LiaS,W
        movwf   Lbyte
        call    hp12
        movf    HibZ,W          ;IB Zero
        movwf   Mbyte
        movf    LibZ,W
        movwf   Lbyte
        call    hp12
        movf    HibC,W          ;IB Center
        movwf   Mbyte
        movf    LibC,W
        movwf   Lbyte
        call    hp12
        movf    HibS,W          ;IB Span
        movwf   Mbyte
        movf    LibS,W
        movwf   Lbyte
        goto    hp12

;*****************************************************************
;print the Cal Time in HEX (16 bits)

hpCT    movf    Hcal,W          ;print Calibration time
        movwf   Mbyte
        movf    Lcal,W
        movwf   Lbyte
        goto    hp16

;*****************************************************************
;output 16 bit chars in Mbyte, Lbyte as HEX ASCII to TX DATA
```

```
hp16    swapf   Mbyte,W
        movwf   TX_BUF
        call    TxHex           ;print Mbyte hi nibble in hex
hp12    movf    Mbyte,W
        movwf   TX_BUF
        call    TxHex           ;print Mbyte low nibble in hex
hp8     swapf   Lbyte,W
        movwf   TX_BUF
        call    TxHex           ;print Lbyte hi nibble in hex
hp4     movf    Lbyte,W
        movwf   TX_BUF
        call    TxHex           ;print Lbyte low nibble in hex
        goto    TxSp            ;Tx space returns to caller

;*********************************************************************
;convert lo nibble in TX_BUF to HEX ASCII and send

TxHex   movf    TX_BUF,W        ;get transmit data
        andlw   0x0f            ;mask hi bits
        sublw   0x09            ;9 - W if >9, Cflag = 0
        movf    TX_BUF,W        ;get data
        andlw   0x0f            ;mask hi bits
        btfss   Cflag           ;is input >9
        addlw   0x07            ;if >9 add 7 for A-F
        addlw   "0"
        goto    Tx

;*********************************************************************
;                           BCD Print Routines
;*********************************************************************
;print four status bits
;The first bit indicates normal (0) or prescaler (1) mode,
;the second bit indicates normal (0) or high gain (1) mode,
;the third bit indicates autocal off (0)or on (1), and the
;fourth bit indicates (1) when an interpolator value exceeded
;the limits.

prtSt   movlw   "0"
        btfsc   PSM             ;prescaler mode
        movlw   "1"
        call    Tx
        movlw   "0"
        btfsc   HGM             ;high gain mode
        movlw   "1"
        call    Tx
        movlw   "0"
        btfsc   ACM             ;autocal mode
        movlw   "1"
        call    Tx
        movlw   "0"
        btfsc   Ilmt            ;I limit flag
        movlw   "1"
        call    Tx
        goto    TxSp

;*********************************************************************
;print interpolated value as +/- 3-digit BCD range

prtIV   movf    HIval,W         ;print interpolated value
        movwf   Hbyte
        movf    LIval,W
        movwf   Mbyte
        movlw   " "
        btfss   Hbyte,7         ;if neg
        goto    $ + 7
        comf    Hbyte           ;invert neg value
        comf    Mbyte
        incf    Mbyte
        btfsc   Zflag
        incf    Hbyte
```

```
        movlw   "-"             ;print neg sign
        call    Tx
        goto    prt3D

;*********************************************************************
;print corrected start and stop data

prtCV   movf    Hstart,W        ;start
        movwf   Hbyte
        movf    Lstart,W
        movwf   Mbyte
        call    prt3D
        movf    Hstop,W         ;stop
        movwf   Hbyte
        movf    Lstop,W
        movwf   Mbyte
        goto    prt3D

;*********************************************************************
;print interpolator values start data, stop data,
;Ch A zero, Ch A span, Ch B zero, Ch B span

prtIC   call    prtID
prtCD   movf    HiaZ,W          ;IA Zero
        movwf   Hbyte
        movf    LiaZ,W
        movwf   Mbyte
        call    prt3D
        movf    HiaS,W          ;IA Span
        movwf   Hbyte
        movf    LiaS,W
        movwf   Mbyte
        call    prt3D
        movf    HibZ,W          ;IB Zero
        movwf   Hbyte
        movf    LibZ,W
        movwf   Mbyte
        call    prt3D
        movf    HibS,W          ;IB Span
        movwf   Hbyte
        movf    LibS,W
        movwf   Mbyte
        goto    prt3D

;*********************************************************************
;print interpolator zero, center, span values

prtCC   movf    HiaZ,W          ;IA Zero
        movwf   Hbyte
        movf    LiaZ,W
        movwf   Mbyte
        call    prt3D
        movf    HiaC,W          ;IA Center
        movwf   Hbyte
        movf    LiaC,W
        movwf   Mbyte
        call    prt3D
        movf    HiaS,W          ;IA Span
        movwf   Hbyte
        movf    LiaS,W
        movwf   Mbyte
        call    prt3D
        movf    HibZ,W          ;IB Zero
        movwf   Hbyte
        movf    LibZ,W
        movwf   Mbyte
        call    prt3D
        movf    HibC,W          ;IB Center
        movwf   Hbyte
        movf    LibC,W
        movwf   Mbyte
```

```
        call    prt3D
        movf    HibS,W          ;IB Span
        movwf   Hbyte
        movf    LibS,W
        movwf   Mbyte
        goto    prt3D

;*********************************************************************
;print interpolator peak values

prtPD   movf    HiaO,W          ;IA Min
        movwf   Hbyte
        movf    LiaO,W
        movwf   Mbyte
        call    prt3D
        movf    HiaM,W          ;IA Max
        movwf   Hbyte
        movf    LiaM,W
        movwf   Mbyte
        call    prt3D
        movf    HibO,W          ;IB Min
        movwf   Hbyte
        movf    LibO,W
        movwf   Mbyte
        call    prt3D
        movf    HibM,W          ;IB Max
        movwf   Hbyte
        movf    LibM,W
        movwf   Mbyte
        goto    prt3D

;*********************************************************************
;output 16 bit BCD chars in BD0,BD1,BD2 as ASCII to TX DATA
;print as 0-999 3-digit BCD interpolator values

prtID   movf    HriA,W          ;IA Raw
        movwf   Hbyte
        movf    LriA,W
        movwf   Mbyte
        call    prt3D
        movf    HriB,W          ;IB Raw
        movwf   Hbyte
        movf    LriB,W
        movwf   Mbyte
prt3D   call    B16_BCD         ;convert bytes to BCD
        movf    BD1,W           ;send BD1
        movwf   TX_BUF
        call    TxChar
        call    TXBD2
        goto    TxSp

;*********************************************************************
;output 16 bit BCD chars in BD0,BD1,BD2 as ASCII to TX DATA
;max count = 65535 or 5-digit BCD

prtCT   movf    Hcal,W          ;print Calibration time
        movwf   Hbyte
        movf    Lcal,W
        movwf   Mbyte
prt16   call    B16_BCD         ;convert bytes to BCD
        call    TXBD1
        call    TXBD2
        goto    TxSp            ;Tx returns to caller

;*********************************************************************
;output 32 bit BCD chars in BD0,BD1,BD2,BD3,BD4 as ASCII to TX DATA
;max count = 4,294,967,296 or 10-digit BCD

prtData movf    Mdata,W         ;print low 32 bits of data
        movwf   Hbyte
        movf    Ldata,W
```

```
        movwf   Mbyte
        movf    Sdata,W
        movwf   Lbyte
        movf    SSdata,W
        movwf   Sbyte
        goto    prt32
prtPS   movf    SSPS,W          ;print low 32 bits of 40-bit value
        movwf   Sbyte
        movf    SPS,W
        movwf   Lbyte
        movf    LPS,W
        movwf   Mbyte
        movf    MPS,W
        movwf   Hbyte
prt32   call    B32_BCD         ;convert bytes to BCD
        swapf   BD0,W           ;send MSD first
        movwf   TX_BUF          ;send BD0 hi nibble
        call    TxChar
        call    TXBD1           ;send BCD
        call    TXBD2
        call    TXBD3
        call    TXBD4
        goto    TxSp            ;Tx space returns to caller

;*********************************************************************
;16 bit binary to BCD conversion (adapted from AN544)
;input in Hbyte, Mbyte and output in BD0, BD1, BD2

B16_BCD bcf     Cflag           ;clear carry bit
        movlw   D'16'
        movwf   bCnt            ;set bCnt = 16 bits
        call    clrBCD          ;clear work registers
RL16    rlf     Mbyte           ;rotate 1 bit
        rlf     Hbyte
        rlf     BD2
        rlf     BD1
        rlf     BD0
        decfsz  bCnt            ;16 bits done?
        goto    $ + 2           ;no, process more BCD
        return                  ;yes, return
        movlw   BD2             ;load addr of BD2 as indir addr
        movwf   FSR
        movlw   0x03            ;process 3 registers
        call    CnvBCD          ;convert to BCD
        goto    RL16            ;get next bit

;*********************************************************************
;32 bit binary to BCD conversion (adapted from AN544)
;input in Hbyte, Mbyte, Lbyte, Sbyte and output in BD0, BD1, BD2, BD3, BD4

B32_BCD bcf     Cflag           ;clear carry bit
        movlw   D'32'
        movwf   bCnt            ;set bCnt = 32 bits
        call    clrBCD          ;clear work registers
RL32    rlf     Sbyte           ;rotate 1 bit
        rlf     Lbyte
        rlf     Mbyte
        rlf     Hbyte
        rlf     BD4
        rlf     BD3
        rlf     BD2
        rlf     BD1
        rlf     BD0
        decfsz  bCnt            ;32 bits done?
        goto    $ + 2           ;no, process more BCD
        return                  ;yes, return
        movlw   BD4             ;load addr of BD4 as indir addr
        movwf   FSR
        movlw   0x05            ;process 5 registers
        call    CnvBCD          ;convert to BCD
        goto    RL32
```

```
;*********************************************************************
;clear BCD work registers

clrBCD  clrf    BD0
        clrf    BD1
        clrf    BD2
        clrf    BD3
        clrf    BD4
        return

;*********************************************************************
;convert to BCD

CnvBCD  movwf   btmp            ;W has # regs on call
Adj     movf    INDF,W          ;get reg via indirect addr
        addlw   0x03
        movwf   bTst            ;sum to bTst for test
        btfsc   bTst,3          ;test if >0x07
        movwf   INDF            ;yes - store sum
        movf    INDF,W          ;get original or sum
        addlw   0x30            ;test hi byte
        movwf   bTst            ;sum to bTst for test
        btfsc   bTst,7          ;test result >0x70
        movwf   INDF            ;save as BCD
        incf    FSR             ;next reg
        decfsz  btmp            ;Done?
        goto    Adj             ;no, do next adj
        return                  ;yes, return

;*********************************************************************
;send nibbles of BD0,BD1,BD2,BD3,BD4,BD5,BD6 as BCD ASCII to TX DATA

TXBD1   movf    BD0,W           ;send MSD first
        movwf   TX_BUF          ;send BD0 low nibble
        call    TxChar
        movf    BD1,W           ;send BD1
TXBD    movwf   TX_BUF
        swapf   TX_BUF          ;BD1 hi nibble
        call    TxChar
        swapf   TX_BUF          ;BD1 low nibble
        goto    TxChar
TXBD2   movf    BD2,W           ;send BD2
        goto    TXBD
TXBD3   movf    BD3,W           ;get BD3
        goto    TXBD
TXBD4   movf    BD4,W           ;get BD4
        goto    TXBD

;*********************************************************************
;convert lo nibble in TX_BUF to BCD ASCII and send

TxChar  movf    TX_BUF,W        ;get the buffer
        andlw   0x0f            ;mask high nibble
        addlw   "0"             ;make into ASCII
Tx      btfss   TXSTA,TRMT      ;test for Tx buffer empty
        goto    $ - 1           ;wait till buffer empty
        movwf   TXREG           ;send it
        btfsc   t1f             ;if TMR1 overflow set
        call    UpdOF           ;update TMR1 overflow counter
        return

;*********************************************************************
;transmit carriage return, line feed

TxCrLf  movlw   "\r"            ;send CR direct to Tx
        call    Tx
        movlw   "\n"            ;send LF direct to Tx
        goto    Tx              ;Tx returns to caller

;*********************************************************************
```

```
;transmit space to separate values

TxSp    movlw   " "             ;send space
        goto    Tx              ;Tx returns to caller

;*********************************************************************
; Get data via USART rcv mode

Rx232   btfss   PIR1,RCIF       ;have we received a char?
        return                  ;no - nothing to do
        movf    RCSTA,W         ;check for rcv status for error
        andlw   0x06            ;select only error bits
        btfss   Zflag           ;if any set, jump to
        goto    RxErr           ;error service routine
        movf    RCREG,W         ;get char from input buffer
        movwf   RX_BUF          ;store in RX_BUF
        bsf     BrdyF           ;set byte available flag
        return
RxErr   bcf     RCSTA,CREN      ;clear CREN to clear overrun error
        movf    RCREG,W         ;read RCREG to clear framing error
        bsf     RCSTA,CREN      ;set CREN to rcv
        bcf     BrdyF           ;clear byte ready flag
        clrf    cHdr0           ;clear header flags
        clrf    cHdr1
        return

;*********************************************************************
;                       Serial Command Decoder
;           Based on Simple Commands by Richard H McCorkle
;    http://www.piclist.com/techref/member/RHM-SSS-SC4/Scmds.htm
;*********************************************************************
;Check command received via USART rcv mode. # aborts current command

CmdProc bcf     BrdyF           ;clear byte ready flag
        movlw   "#"             ;is char a "#" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 5           ;no, continue
        bcf     Nrdy            ;clear transfer flags
        bcf     LoByt
        clrf    TSS             ;clear temp reg
        goto    Chf             ;clear header flags and exit
        btfss   Hdr8            ;is header 8 flag set? (Number Return)
        goto    H7              ;no, test next header
        call    GetByt          ;get data bytes
        btfsc   Nrdy            ;check not ready
        return                  ;if set, get next data
        goto    Chf             ;clear header flags and exit

H7      btfss   Hdr7            ;header 7 flag set (@@P received)?
        goto    H6              ;no, test for next header

; @@Pc  Print Calibration Data in BCD
        movlw   "c"             ;is char a "c" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        call    prtCC           ;print Calibration Data
        call    prtCT
        goto    Prtn            ;clear header flags and exit

; @@Pi  Print Interpolator & Cal Data in BCD
        movlw   "i"             ;is char an "i" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        call    prtIC           ;print Interpolator & Cal Data
        goto    Prtn            ;clear header flags and exit

; @@Ps  Print Status
        movlw   "s"             ;is char an "s" ?
```

```
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        call    prtSt
        goto    Prtn            ;clear header flags and exit

; @@Pt  Print Calibration Time in BCD
        movlw   "t"             ;is char a "t" ?
        call    RxChk
        btfss   Zflag
        goto    Chf             ;no valid @@P command, exit
        call    prtCT           ;print Calibration time
Prtn    call    TxCrLf          ;send new line
        goto    Chf             ;clear header flags and exit

H6      btfss   Hdr6            ;header 6 flag set (@@M received)?
        goto    H5              ;no, test for next header

; @@Md  Set Direct Input to Counter
        movlw   "d"             ;is char a "d" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 5           ;no, next header
        bcf     PSM             ;clear Prescaler Mode Flag
        movlw   0xC6            ;set TMR1 as f_osc/1, async, + gate, ext clk
        movwf   T1CON
        goto    Chf             ;clear header flags and exit

; @@Mh  Set High Gain Mode (Gain = 800)
        movlw   "h"             ;is char an "h" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bsf     HGM             ;set High Gain Mode Flag
        goto    Chf             ;clear header flags and exit

; @@Mn  Set Normal Gain Mode (Gain = 400)
        movlw   "n"             ;is char an "n" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bcf     HGM             ;clear High Gain Mode Flag
        goto    Chf             ;clear header flags and exit

; @@Mp  Set Prescaler Input to Counter
        movlw   "p"             ;is char a "p" ?
        call    RxChk
        btfss   Zflag
        goto    Chf             ;no valid @@M command, exit
        bsf     PSM             ;set Prescaler Mode Flag
        movlw   0x06            ;set TMR1 as f_osc/1, async, no gate, ext clk
        movwf   T1CON
        goto    Chf             ;clear header flags and exit

H5      btfss   Hdr5            ;header 5 flag set (@@D received)?
        goto    H4              ;no, test for next header

; @@Dc  Toggle Display Cal Values (Z,S)
        movlw   "c"             ;is char a "c" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        movf    DMB,W
        xorlw   0x04            ;toggle bit 2
        goto    tdb

; @@Dd  Toggle Display 10-Digit Time Delay in BCD
        movlw   "d"             ;is char a "d" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
```

```
        movf    DMB,W
        xorlw   0x20            ;toggle bit 5
        goto    tdb

; @@Di  Toggle Display Corrected Start and Stop in BCD
        movlw   "i"             ;is char an "i" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        movf    DMB,W
        xorlw   0x08            ;toggle bit 3
        goto    tdb

; @@Dp  Toggle Display Peak Detector Values
        movlw   "p"             ;is char a "p" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        movf    DMB,W
        xorlw   0x80            ;toggle bit 7
        goto    tdb

; @@Dr  Toggle Display Raw Interpolator Data in BCD
        movlw   "r"             ;is char an "r" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        movf    DMB,W
        xorlw   0x02            ;toggle bit 1
        goto    tdb

; @@Ds  Toggle Display Status Bits
        movlw   "s"             ;is char an "s" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        movf    DMB,W
        xorlw   0x40            ;toggle bit 6
        goto    tdb

; @@Dt  Toggle Display TIC direct
        movlw   "t"             ;is char a "t" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        movf    DMB,W
        xorlw   0x01            ;toggle bit 0
        goto    tdb

; @@Dv  Toggle Display Interpolated Value
        movlw   "v"             ;is char a "v" ?
        call    RxChk
        btfss   Zflag
        goto    Chf             ;no valid @@D command, exit
        movf    DMB,W
        xorlw   0x10            ;toggle bit 4
tdb     movwf   DMB
        goto    Chf             ;clear header flags and exit

H4      btfss   Hdr4            ;is header 4 flag set? (@@C rcvd)
        goto    H3              ;no, test for next header

; @@Cd  Disable Autocal Mode, Manual Cal
        movlw   "d"             ;is char a "d" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bcf     ACM             ;clear autocal flag
        goto    Chf             ;clear header flags and exit

; @@Ce  Enable Autocal Mode
```

```
        movlw   "e"             ;is char an "e" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        bsf     ACM             ;set autocal flag
        call    clrSC           ;init peak detectors
        goto    Chf             ;clear header flags and exit

; @@Cp  Print Calibration Values and Duration in HEX
        movlw   "p"             ;is char a "p" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        call    hpCD            ;print Calibration Data
        call    hpCT            ;print Calibration Time
        goto    Prtn            ;clear header flags and exit

; @@Cr  Reset Calibration Peak Detectors
        movlw   "r"             ;is char an "r" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        call    clrSC           ;init peak detectors
        goto    Chf             ;clear header flags and exit

; @@Ct  Set Calibration Time Duration xxxx HEX
        movlw   "t"             ;is char a "t" ?
        call    RxChk
        btfss   Zflag
        goto    Chf             ;no valid @@C command, exit
        movlw   Hcal            ;move indirect address to W
        goto    G2B

H3      btfss   Hdr3            ;header 3 flag set (@@B received)?
        goto    H2              ;no, test for next header

; @@Bc  Set Ch B Center Value xxx HEX
        movlw   "c"             ;is char a "c" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        bsf     LoByt           ;set LoByt for 1 byte
        movlw   HibC
        goto    G2B

; @@Bs  Set Ch B Span Value    xxx HEX
        movlw   "s"             ;is char an "s" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        bsf     LoByt           ;set LoByt for 1 byte
        movlw   HibS
        goto    G2B

; @@Bz  Set Ch B Zero Value    xxx HEX
        movlw   "z"             ;is char a "z" ?
        call    RxChk
        btfss   Zflag
        goto    Chf             ;no valid @@B command, exit
        bsf     LoByt           ;set LoByt for 1 byte
        movlw   HibZ
        goto    G2B

H2      btfss   Hdr2            ;header 2 flag set (@@A received)?
        goto    H1              ;no, test for next header

; @@Ac Set Ch A Center Value xxx HEX
        movlw   "c"             ;is char a "c" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
```

```
        bsf     LoByt           ;set LoByt for 1 byte
        movlw   HiaC
        goto    G2B

; @@As Set Ch A Span Value    xxx HEX
        movlw   "s"             ;is char an "s" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        bsf     LoByt           ;set LoByt for 1 byte
        movlw   HiaS
        goto    G2B

; @@Az  Set Ch A Zero Value    xxx HEX
        movlw   "z"             ;is char a "z" ?
        call    RxChk
        btfss   Zflag
        goto    Chf             ;no valid @@A command, exit
        bsf     LoByt           ;set LoByt for 1 byte
        movlw   HiaZ
G2B     movwf   Dadr            ;store in Dadr
        movlw   0x02            ;get 2 registers (4 bytes)
GBR     movwf   BytCt
        clrf    TSS
        bsf     Nrdy            ;set not ready flag
        bsf     Hdr8            ;set Header 8 flag
        return

H1      btfss   Hdr1            ;header 1 flag set (@@ received)?
        goto    H0              ;no, test for header start

; @@A   Interpolator A Calibration
        movlw   "A"             ;is char an "A" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bsf     Hdr2            ;set Header 2 flag
        return

; @@B   Interpolator B Calibration
        movlw   "B"             ;is char a "B" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bsf     Hdr3            ;set Header 3 flag
        return

; @@C   Calibration Commands
        movlw   "C"             ;is char a "C" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bsf     Hdr4            ;set Header 4 flag
        return

; @@D   Toggle Display Bits
        movlw   "D"             ;is char a "D" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bsf     Hdr5            ;set Header 5 flag
        return

; @@M   Mode Commands
        movlw   "M"             ;is char an "M" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bsf     Hdr6            ;set Header 6 flag
        return
```

```
; @@P   Print value to serial TX port
        movlw   "P"             ;is char a "P" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bsf     Hdr7            ;yes, set header 7 flag
        return

; @@R   Run Command - Reset and Enable Counter
        movlw   "R"             ;is char an "R" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 4           ;no, next header
        bcf     HoldF           ;clear hold flag
        call    clrTC           ;reset counter
        goto    Chf             ;clear header flags and exit

; @@S   Stop Command - Hold Counter
        movlw   "S"             ;is char an "S" ?
        call    RxChk
        btfss   Zflag
        goto    $ + 3           ;no, next header
        bsf     HoldF
        goto    Chf             ;clear header flags and exit

; @@U   Save data to EEPROM
        movlw   "U"             ;is char a "U" ?
        call    RxChk
        btfss   Zflag
        goto    Chf             ;no valid @@ command, exit
        bsf     DUF             ;set data update flag
        goto    Chf             ;clear header flags and exit

H0      movlw   "@"             ;is char a "@" ?
        call    RxChk
        btfsc   Zflag
        goto    $ + 4
        btfsc   Hdr0            ;If @ rcvd but second char not @
        bcf     Hdr0            ;clear header 0 flag and
        return                  ;exit
        btfsc   Hdr0            ;was header 0 flag set?
        goto    $ + 3
        bsf     Hdr0            ;no, set header 0 flag (@ rcvd)
        return                  ;exit
        bsf     Hdr1            ;yes, set header 1 flag (@@ rcvd)
        return                  ;exit

Chf     clrf    cHdr0           ;clear header flags
        clrf    cHdr1
        return

;*****************************************************************
; Check Received byte for match to character in W on call
; Set Zflag if match (used in background so use btmp)

RxChk   movwf   btmp
        movf    RX_BUF,W
        subwf   btmp,W
        return

;*****************************************************************
; Hex Convert - convert ASCII 0-9 or A-F to number 0-15 in W
; converts small a-f to caps by clearing 32 bit first
; so either caps or smalls for a-f work.
;(used in background, so use btmp)

HexC    movf    RX_BUF,W        ;get the byte
        movwf   btmp
        btfss   btmp,6          ;number or char?
        goto    $ + 4
        bcf     btmp,5          ;clear 32 bit (convert small to caps)
```

```
        movlw   D'7'            ;subtract 55 to convert A-F to values 10-15
        subwf   btmp
        movlw   D'48'           ;subtract 48 to value
        subwf   btmp
        movf    btmp,W
        andlw   0x0f            ;discard high bits (if wrong char)
        return

;*********************************************************************
;Get Byte - fetch number of registers in BytCt and store
;at address starting at Dadr, clear Nrdy when finished.
;Revised to use temp storage until full byte is received.
;Clear TSS and set LoByt before call to fetch 1/2 byte.

GetByt  call    HexC            ;convert input to value
        btfsc   LoByt           ;if LoByte set,
        goto    $ + 5
        movwf   TSS             ;load in temp storage
        swapf   TSS             ;move to hi nibble
        bsf     LoByt           ;set low byte flag
        return
        addwf   TSS             ;add to value in storage
        movf    Dadr,W          ;get storage address
        movwf   FSR
        movf    TSS,W           ;move data to W
        movwf   INDF            ;place in register
        clrf    TSS             ;clear temp reg
        bcf     LoByt           ;clear low byte flag
        decfsz  BytCt           ;got all the registers?
        goto    $ + 3
        bcf     Nrdy            ;clear not ready flag
        return
        incf    FSR             ;point to next data
        movf    FSR,W
        movwf   Dadr            ;and store in Dadr
        return

;*********************************************************************
;                           EEPROM Routines
;                         Copyright © 2006
;                        Richard H McCorkle
;*********************************************************************
;                       EEPROM Memory Usage

;  Address             Function                        Registers
;    0 - 1       16 bit Ch A zero                      HiaZ,LiaZ
;    2 - 3       16 bit Ch A Center                    HiaC,LiaC
;    4 - 5       16 bit Ch A span                      HiaS,LiaS
;    6 - 7       16 bit Ch B zero                      HibZ,LibZ
;    8 - 9       16 bit Ch A Center                    HibC,LibC
;    A - B       16 bit Ch B span                      HibS,LibS
;    C - D       16 bit Cal Duration (Samples)         Hcal,Lcal
;    E            8 bit Display Mode Bits              DMB
;    F            8 bit Command Mode Bits              CMB

;*********************************************************************
;While EEPROM's offer unlimited reads, the number of write cycles
;is limited to 1M before errors occur due to memory cell failure.
;To maximize EEPROM life, this program performs EEPROM writes once
;per hour giving a predicted life of > 100 years @ 25C. More frequent
;writes should be avoided. Predicted EEPROM life updating once per
;minute is < 2 years @ 25C !!
;
;Reads EEPROM data byte pointed to by address in Padr on call
;Returns with data in W

PromRd  movf    Padr,W          ; Address to read in Padr
        bcf     STATUS,RP1      ; select Bank 1
        bsf     STATUS,RP0
        movwf   EEADR           ; store read address
        bcf     EECON1,EEPGD    ; Point to Data memory
```

```
        bsf     EECON1,RD       ; EE Read
        movf    EEDAT,W         ; W contains EEDATA
        bcf     STATUS,RP0      ; back to Bank 0
        return

;*********************************************************************
;Writes data byte pointed to by FSR to EEPROM address pointed to by
;Padr on call

PromWr  movf    INDF,W          ;Data Value to write
        bcf     STATUS,RP1      ;select Bank 1
        bsf     STATUS,RP0
        movwf   EEDAT           ;store data value
        bcf     STATUS,RP0      ;select Bank 0
        movf    Padr,W          ;Data Address to write
        bsf     STATUS,RP0      ;select Bank 1
        movwf   EEADR           ;store address
        bcf     EECON1,EEPGD    ;Point to DATA memory
        bcf     INTCON,GIE      ;disable interrupts
        bsf     EECON1,WREN     ;Enable writes
        movlw   0x55
        movwf   EECON2          ;Write 55h
        movlw   0xaa
        movwf   EECON2          ;Write AAh
        bsf     EECON1,WR       ;Set WR bit to begin write
        bcf     EECON1,WREN     ;Disable writes
        bsf     INTCON,GIE      ;enable interrupts
        btfsc   EECON1,WR       ;Wait for write
        goto    $ - 1           ;to complete
        bcf     STATUS,RP0      ;back to bank 0
        btfsc   t1f             ;if TMR1 overflow set
        call    UpdOF           ;update TMR1 overflow counter
        return

;*********************************************************************
; Update PROM with 16 registers of current data

D2prom  bcf     DUF             ;clear update flag
        clrf    Padr            ;move start address to Padr
        movlw   HiaZ            ;load addr of HiaZ as indir addr
        movwf   FSR
        movlw   0x10            ;get 16 bytes at HiaZ -> HiaZ + 15
        movwf   temp
pWr     call    PromWr          ;Write data
        incf    Padr            ;point to next prom address
        incf    FSR             ;point to next register
        decfsz  temp            ;Done?
        goto    pWr             ;no, get next word
        return

;*********************************************************************
; Initialize internal registers from PROM

InitCon clrf    PORTA           ;clear port output latches
        clrf    PORTC
        clrf    INTCON          ;disable all interrupts for now
        clrf    T1CON           ;Stop Timer1
        movlw   0x07            ;set PORTA pins as digital (not comparator inputs)
        movwf   CMCON0
        bsf     CMCON1,T1GSS    ;enable external gate on TMR1
        clrwdt                  ;clear watch dog & prescaler

;initialize bank 1 control regs

        bsf     STATUS,RP0      ;select bank 1
        movlw   0x71            ;select 8MHz internal clock for PIC
        movwf   OSCCON
;       movlw   0x00            ;put the cal value in OSCTUNE
;       movwf   OSCTUNE         ;to calibrate oscillator
;       movlw   0x07            ;int on falling edge of RA2
        movlw   0x47            ;int on rising edge of RA2
```

```
        movwf   OPTION_REG
        movlw   0x03            ;set PORTA pin 0,1 as analog, 2-5 as digital
        movwf   ANSEL
        movlw   0x50            ;set A/D Clock = 16Tosc
        movwf   ADCON1
        movlw   0x3f            ;set PORTA pin 0-5 as inputs
        movwf   TRISA
        movlw   0x37            ;set PORTC 0-2 as inputs, 3 as output, 2 serial pins
        movwf   TRISC
        clrf    PIE1            ;no int on async xmt (tst TRMT instead)

;back to bank 0

        bcf     STATUS,RP0      ;select bank 0
        movlw   0x81            ;Analog on A0, Right Justified, Vdd Reference
        movwf   ADCON0
        bsf     TXSTA,TXEN      ;enable USART xmt (async mode)
        bsf     TXSTA,BRGH      ;set USART hi speed mode
        movlw   D'51'           ;set async rate at 9600 baud (51. for 8 MHz int, BRGH=1)
        movwf   SPBRG
        ifdef   RS232
        bcf     BAUDCTL,SCKP    ;normal transmit polarity if RS-232 comm
        else
        bsf     BAUDCTL,SCKP    ;invert transmit polarity if TTL comm
        endif
        bsf     RCSTA,CREN      ;enable USART rcv (async mode)
        bsf     RCSTA,SPEN      ;enable serial port
        bsf     INTCON,INTE     ;enable interrupt on RA2
        clrf    PIR1            ;clear peripheral interrupt flags
        clrf    TMR1L           ;clear TMR1
        clrf    TMR1H
        clrf    HOFctr          ;clear overflow counter
        clrf    LOFctr
        bcf     t1f             ;clear overflow flag
        clrf    flg0            ;clear all flags
        clrf    cHdr0           ;clear comm headers
        clrf    cHdr1
        clrw                    ;Read 16 registers from PROM
        movwf   Padr            ;move start address to Padr
        movlw   HiaZ            ;load addr of HiaZ as indir addr
        movwf   FSR
        movlw   0x10            ;get 16 bytes at HiaZ -> HiaZ + 15
        movwf   temp
        call    PromRd          ;Read data
        movwf   INDF            ;store data in register
        incf    Padr            ;point to next prom address
        incf    FSR                     ;point to next register
        decfsz  temp            ;Done?
        goto    $ - 5           ;no, get next word
        movlw   0xC6            ;set TMR1 as f_osc/1, async, + gate, ext clk
        btfsc   PSM             ;if prescaler mode
        movlw   0x06            ;set TMR1 as f_osc/1, async, no gate, ext clk
        movwf   T1CON
        movf    HiaC,W          ;init peak detectors to center
        movwf   HiaM
        movwf   HiaO
        movf    LiaC,W
        movwf   LiaM
        movwf   LiaO
        movf    HibC,W
        movwf   HibM
        movwf   HibO
        movf    LibC,W
        movwf   LibM
        movwf   LibO
        clrf    HcalT           ;load cal timer with 60
        movlw   0x3c
        movwf   LcalT
        bsf     FpF             ;set first cal flag
        return
```

```
;*********************************************************************
        de      "Simple PICTIC Rev 1.00, Copyright © Richard H McCorkle 2008"
        de      " "

        END
```