```
; HtrAndPlsCtrl

;Need to detect switch turning off so that multi-presses can be faster thab auto-repeat
;Need to set up heater output.
;19/11/2010 19:39 with scaling, byte 0 of the eeprom says add/subtract N/256ths to the reading.
;19/11/2010 19:40 dimming when displaying voltage
;
;
;#define EMULATE
;                noexpand
                list    n=0             ;lines per page
                list    st=on           ;symbol table
                expand
                radix   dec             ;decimal radix
                list    mm=on           ;memory map
                list    n=0             ;no listing paging
                list    x=off           ;no macro expansion
;                errorlevel -302         ;suppress bank warnings
#define _TimeDate "27/12/2010 16:51"
#define _Version "C"

        include P16F873.INC             ;standard register definitions
;
; macros
                nolist
mOrg            macro   Location
_CurrentLocation set    Location
                endm
mRes            macro   Lable,Size
Lable           equ     _CurrentLocation
_CurrentLocation set    _CurrentLocation+Size
                endm
;*******************************************************************************
; mResBlw        macro   Symbol,Size,UpperLimit
;
; Purpose:    Reserve memory ("register file(s)")
; Function:   Checks space below upper limit of register bank, then reserves space for Symbol
; Parameters: Symbol is the variable name;
;             Size is the number of register files (bytes);
;             UpperLimit is the 1 past the last available location in this register bank
; Example:        ResBelow        STACKW,1,CurLimit   ;Psuedo-Stack
;
                nolist
mResBlw         macro   Symbol,Size,UpperLimit
                nolist
                if      $+Size>=UpperLimit
                list
                  error "ResBelow: Can't put symbol within range"
                nolist
                else
                list
                mRes     Symbol,Size
                nolist
                endif
                list
                endm
                list
;
BankFor         macro   Location
                nolist
                if ((Location<0x70)||(Location>0x7f)||((CurrentBank&0x8000)!=0))
                  if (((CurrentBank&0x8000)!=0)||((Location&0x100)!=(CurrentBank&0x100)))
                    if (Location&0x100)!=0
                list
                       bsf    STATUS,RP1
                nolist
                    else
                list
                       bcf    STATUS,RP1
                nolist
                    endif
                  endif
                  if (((CurrentBank & 0x8000)!=0)||((Location&0x80)!=(CurrentBank&0x80)))
                    if (Location & 0x80)!=0
                list
                       bsf    STATUS,RP0
                nolist
                    else
                list
                       bcf    STATUS,RP0
                nolist
                    endif
                  endif
;                list
CurrentBank       set     Location
                endif
                list
                endm
```

```
                list
CurrentBank     set     0x8000
                list
;*****************************************************************************
; BankFrc         macro   Location
;
; Purpose:    Ensure correct register file paging
; Function:   Forces register bank selection (for whenever the assembler will not know
;             the actual current bank)
; Comment:    CurrentBank will remember the new location
; Parameters: Location - the register to be paged in for access
; Example:        BankFrc offset
;
                nolist
BankFrc         macro   Location
                BankSel Location
                nolist
CurrentBank     set     Location
                list
                endm
                list

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                __CONFIG        _WDT_OFF & _LVP_OFF & _RC_OSC
;
PicClkIn        equ     4705882         ;not really this accurate!
;
; Has 2 1/2 sgment display, normally displays volts (19.9 fs)
;
; On power-up (from ignition)
;  Normally displays voltage;
;   Waits 250ms then puts out 250mS pulse to power relay to set hand gearschange operation
;
; Then looks for two switch inputs to raise or lower heated vest control mark/space.
;   momentarily displays heated vest control instead of voltage (say 1 second).
MaxHeatSetting  equ     8               ;maximum heat setting
MaxBrightnessSetting equ 8              ;brightness heat setting (range ..8)
MinBrightnessSetting equ 1              ;normal brightness
FirstBrightness equ     8

DebounceTime    equ     30              ;debounce time
ShowHeatTime    equ     3000            ;time to show heat setting
BrightnessWaitTime equ  ShowHeatTime+2000 ;total time for bright setting
RepeatTime      equ     400             ;auto-repeat inteval
SegTestTime     equ     100              ;segment test time during starting sequence
;
kVoltageTooLow  equ     120*0x400/200   ;raw for voltage (volts*10)*ADCfsd/(voltage_for_fsd*10)
kLowVoltDefault equ     120
;
PrePulseTime    equ     150       ;milliseconds prior to pulse
PulseTime       equ     200       ;milliseconds pulse duration
AdcDisplayTime  equ     256             ;approx milliseconds between display updates
AdcAverageCount equ     64              ;number of adc readings to average
AdcTime         equ     AdcDisplayTime/AdcAverageCount ;time between adc conversions
SwitchTime      equ     500             ;auto-reprat time for switches
;
;ports
;
; port usage
;
HSDPort         equ     PORTA
HSDTris         equ     TRISA
HSDsegs         equ     5
;
MSDPort         equ     PORTC
MSDTris         equ     TRISC           ;LED output
LSDPort         equ     PORTB
LSDTris         equ     TRISB           ;LED output
;
SegAbit         equ     0
SegBbit         equ     1
SegCbit         equ     2
SegDbit         equ     3
SegEbit         equ     4
SegFbit         equ     5
SegGbit         equ     6
;
SegsFor0        equ     (1<<SegAbit|1<<SegBbit|1<<SegCbit|1<<SegDbit|1<<SegEbit|1<<SegFbit)
SegsFor1        equ     (1<<SegBbit|1<<SegCbit)
SegsFor2        equ     (1<<SegAbit|1<<SegBbit|1<<SegDbit|1<<SegEbit|1<<SegGbit)
SegsFor3        equ     (1<<SegAbit|1<<SegBbit|1<<SegCbit|1<<SegDbit|1<<SegGbit)
SegsFor4        equ     (1<<SegBbit|1<<SegCbit|1<<SegFbit|1<<SegGbit)
SegsFor5        equ     (1<<SegAbit|1<<SegCbit|1<<SegDbit|1<<SegFbit|1<<SegGbit)
SegsFor6        equ     (1<<SegAbit|1<<SegCbit|1<<SegDbit|1<<SegEbit|1<<SegFbit|1<<SegGbit)
SegsFor7        equ     (1<<SegAbit|1<<SegBbit|1<<SegCbit)
SegsFor8        equ     (1<<SegAbit|1<<SegBbit|1<<SegCbit|1<<SegDbit|1<<SegEbit|1<<SegFbit|1<<SegGbit)
SegsFor9        equ     (1<<SegAbit|1<<SegBbit|1<<SegCbit|1<<SegDbit|1<<SegFbit|1<<SegGbit)
SegsForA        equ     (1<<SegAbit|1<<SegBbit|1<<SegCbit|1<<SegEbit|1<<SegFbit|1<<SegGbit)
SegsForB        equ     (1<<SegCbit|1<<SegDbit|1<<SegEbit|1<<SegFbit|1<<SegGbit)
```

```
SegsForC        equ     (1<<SegAbit|1<<SegBbit|1<<SegEbit|1<<SegFbit)
SegsForD        equ     (1<<SegBbit|1<<SegCbit|1<<SegDbit|1<<SegEbit|1<<SegGbit)
SegsForE        equ     (1<<SegAbit|1<<SegDbit|1<<SegEbit|1<<SegFbit|1<<SegGbit)
SegsForF        equ     (1<<SegAbit|1<<SegEbit|1<<SegFbit|1<<SegGbit)
SegsForP        equ     (1<<SegAbit|1<<SegBbit|1<<SegEbit|1<<SegFbit|1<<SegGbit)
SegsBlank       equ     (0)
SegsForPrePMs   equ     (1<<SegDbit)
SegsForPrePLs   equ     (1<<SegDbit)
SegsForPMs      equ     (1<<SegAbit|1<<SegBbit|1<<SegCbit|1<<SegEbit|1<<SegFbit)
SegsForPLs      equ     (0);(1<<SegDbit)
SegsForH        equ     (1<<SegBbit|1<<SegCbit|1<<SegEbit|1<<SegFbit|1<<SegGbit)
SegsForL        equ     (1<<SegDbit|1<<SegEbit|1<<SegFbit)
;
DpPort          equ     PORTC
DpBit           equ     7
DpTris          equ     TRISC
;
PulsePort       equ     PORTA
PulseTris       equ     TRISA
PulseBit        equ     3
HeatPort        equ     PORTB
HeatTris        equ     TRISB
HeatBit         equ     7
;
SwitchUpPort    equ     PORTA
SwitchUpTris    equ     TRISA
SwitchUpBit     equ     4
SwitchDownPort  equ     PORTA
SwitchDownTris  equ     TRISA
SwitchDownBit   equ     2
;
; RAM
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
Page0Limit      set     0x7f            ;limit of 1st page RAM
UpperLimit      set     Page0Limit

                mOrg     0x20            ;start of RAM

        mResBlw w_temp,1,UpperLimit    ;irq w save
        mResBlw status_temp,1,UpperLimit ;status ditto
        mResBlw fsr_temp,1,UpperLimit  ;fsr ditto
        mResBlw pclath_temp,1,UpperLimit  ;pclath ditto
;
        mResBlw disp_temp,1,UpperLimit  ;temp during dispay update
        mResBlw EepromData,1,UpperLimit ;data to write to eeptom
        mResBlw EepromAdr,1,UpperLimit  ;address to write in eeprom
        mResBlw EeWrIntcon,1,UpperLimit ;used in EEPROM access
        mResBlw FlagWord1,1,UpperLimit ;1st flag word
PulseFlagWord   equ     FlagWord1       ;}Flag when on says doing pulse stuff
PulseFlagBit    equ     0               ;}
DisplaySwitchFlagWord equ FlagWord1     ;}Flag when on says displaying heat setting
DisplaySwitchFlagBit equ 1              ;}
VoltageTooLowFlagWord equ FlagWord1     ;}Flag when on says voltage too low for heating
VoltageTooLowFlagBit equ 2              ;}
HadPulseFlagWord equ FlagWord1          ;}Flag when on says we've done the pulse
HadPulseFlagBit equ 3                   ;}
        mResBlw HeatSetting,1,UpperLimit ;Holds heat setting, 0..MaxHeatSetting
        mResBlw HeatSettingOld,1,UpperLimit ;Holds heat setting during brightness change
        mResBlw AdcAverageCounter,1,UpperLimit ;Adc averaging counter
        mResBlw AdcResult,2,UpperLimit ;ADC result
        mResBlw AdcFraction,1,UpperLimit ;fractional part of adcresult when scaling
        mResBlw Fraction,2,UpperLimit ;ADC result used as fraction for scaling
        mResBlw ScaleFactor,1,UpperLimit ;Number of fractional parts to add/subtract
        mResBlw Accum,2,UpperLimit ;Accumulator for adc scaling
        mResBlw BuildDec,2,UpperLimit ;build decimal from binary
        mResBlw Brightness,1,UpperLimit ;brightness setting, 1..max
        mResBlw BrightnessTimer,1,UpperLimit ;brightness setting, 1..max
        mResBlw HSDigitSegmentImage,1,UpperLimit ;segment image
        mResBlw MSDigitSegmentImage,1,UpperLimit ;segment image
        mResBlw LSDigitSegmentImage,1,UpperLimit ;segment image
;
;
        mResBlw MillisecondsCounter,2,UpperLimit ;counts up system milliseconds
DoubleTimers    equ     _CurrentLocation
        mResBlw Second,2,UpperLimit     ;counts down mS, reset every second
        mResBlw AdcTmr,2,UpperLimit     ;when to update reading
        mResBlw PulseTimer,2,UpperLimit ;for pulse timing when to update reading
        mResBlw DebounceTimer,2,UpperLimit ;switch debounce time
        mResBlw RepeatTimer,2,UpperLimit ;switch auto-repeat time
        mResBlw ShowHeatTimer,2,UpperLimit ;show heat time
        mResBlw HeatCompareTimer,2,UpperLimit ;mark-space timer
        mResBlw BrightnessWaitTimer,2,UpperLimit ;times out switch hold for brightness setting
DoubleTimersEnd equ     _CurrentLocation
SingleTimers    equ     _CurrentLocation
        mResBlw SingleTimer,1,UpperLimit     ;dummy
SingleTimersEnd equ     _CurrentLocation
;
```

```
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; eeprom
EE_Size         equ     128
;  org  0x2100            ;The absolue address 2100h is
;                          ;mapped to the 0000 location of
;                          ;EE data memory.
EE_Start        equ     0x2100

                mOrg    EE_Start               ;start of eeprom
EE_UpperLimit    equ     EE_Start+EE_Size
;
        mResBlw EE_Scale,1,EE_UpperLimit        ;add +127..-128 256ths of adc reading as calibration
;                                               ;currently set to 0
        mResBlw EE_Offset,1,EE_UpperLimit       ;add +127..-128 256ths to adc reading (unimlpimented)
;                                               ;currently set to 0
        mResBlw EE_MinVoltage,1,EE_UpperLimit   ;Voltage for no heat (1/10ths of a volt)
;                                               ;currently set to 0x82 (13.0 volts)
        mResBlw EE_AdcReading,2,EE_UpperLimit   ;One reading from the adc
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
                org     0               ;start of program
                goto    main
;
                ORG     0x004           ;interrupt vector location
                movwf   w_temp          ;save current W register
                swapf   STATUS,w        ;save status register ...
                BankFrc status_temp     ;bank set to 0
                movwf   status_temp     ;
                movf    FSR,w           ;save FSR
                movwf   fsr_temp
                movf    PCLATH,w
                movwf   pclath_temp
;irq stuff here
                btfsc   PIR1,TMR2IF     ;timer2 irq?
                call    Timer2IrqHandler        ;yes
; any other sources of interrupt here
                BankFrc pclath_temp     ;ensure bank 0
                movf    pclath_temp,w
                movwf   PCLATH
                movf    fsr_temp,w      ;restore FSR
                movwf   FSR
                swapf   status_temp,w   ;restore STATUS register
                movwf   STATUS          ;
                swapf   w_temp,f        ;restore W register contents
                swapf   w_temp,w        ; without affecting status
                retfie                  ;return from interrupt
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DecToSeg
                andlw   0x0f
                addwf   PCL,f
                retlw   SegsFor0
                retlw   SegsFor1
                retlw   SegsFor2
                retlw   SegsFor3
                retlw   SegsFor4
                retlw   SegsFor5
                retlw   SegsFor6
                retlw   SegsFor7
                retlw   SegsFor8
                retlw   SegsFor9
                retlw   SegsForA
                retlw   SegsForB
                retlw   SegsForC
                retlw   SegsForD
                retlw   SegsForE
                retlw   SegsForF
highdollar equ high $
#if high $ != high DecToSeg
 error DecToSeg table crosses boundary
#endif
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main
;
                BankFrc 0

                clrf    STATUS          ;initialise
; sets IRP & RPn to bank 0
                clrf    INTCON
;

; clear all ram
                movlw   0x20
                movwf   FSR
ClrRamLoop0
                clrf    INDF
```

```
                incf    FSR,f
                btfss   FSR,7
                goto    ClrRamLoop0
                movlw   0xa0
                movwf   FSR
ClrRamLoop1
                clrf    INDF
                incf    FSR,f
                btfsc   FSR,7
                goto    ClrRamLoop1
;
;set up ports
                call    InitAdc
                call    InitDisplays
                call    InitSwitchPort
                call    InitHeater
                call    InitPulse
                call    InitTimer2
;
;
;
                BankFor 0               ;revert to normal
;
; enable interrupts
                BankFor INTCON
                bsf     INTCON,PEIE     ;enable peripheral interrupts
                bsf     INTCON,GIE      ;enable any interrupts
                nop
;
;                call    StartSegTestAndPulse
; and set up BrightnessWaitTimer
                movlw   low BrightnessWaitTime
                movwf   BrightnessWaitTimer+0
                movlw   high BrightnessWaitTime
                movwf   BrightnessWaitTimer+1
;
                BankFor 0
 nop
mainloop
                nop
;
                call    CheckPulse      ;deal with the pulse
                call    CheckSwitch     ;note order
                call    CheckAdc        ;note order
                call    CheckBrightnessLevel


                goto    mainloop
;
CheckPulse
;HadPulseFlagWord equ FlagWord1          ;}Flag when on says we've done the pulse
;HadPulseFlagBit equ 3                   ;}

; if neither flag on set timer and set pre-pulse
                btfsc   HadPulseFlagWord,HadPulseFlagBit ;skip if not had pulse
                goto    CPexit                  ;nothing left
; pulse on?
                btfsc   PulsePort,PulseBit      ;skip if pulse not on
                goto    CPpuleIsOn
;timed out?
                movf    PulseTimer+1,w
                iorwf   PulseTimer+0,w
                btfss   STATUS,Z                ;skip if timed out
                goto    CPexit                  ;waiting
; put pulse on
                bsf     PulsePort,PulseBit      ;pulse on
;set up pulse time
                movlw   low PulseTime
                movwf   PulseTimer+0
                movlw   high PulseTime
                movwf   PulseTimer+1
                goto    CPexit                  ;done here
;
CPpuleIsOn
                movf    PulseTimer+1,w
                iorwf   PulseTimer+0,w
                btfss   STATUS,Z                ;skip if timed out
                goto    CPexit                  ;waiting
; take pulse off
                bcf     PulsePort,PulseBit      ;pulse off
                bsf     HadPulseFlagWord,HadPulseFlagBit ;say had pulse
CPexit
                return                          ;exit


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CheckSwitch
; see if debounce timer is running
```

```
                movf    DebounceTimer+1,w
                iorwf   DebounceTimer+0,w
                btfss   STATUS,Z                ;skip if not running
                goto    CSdone                  ;nothing to do
                btfsc   SwitchUpPort,SwitchUpBit ;skip if Up switch active
                goto    CScheckDown
; set debounce timer
                call    SetDebounceTimer
; see if display timer running (do we already know what the setting is?)
                movf    ShowHeatTimer+1,w
                iorwf   ShowHeatTimer+0,w
                btfsc   STATUS,Z                ;skip if running
                goto    CSdisplayHeat1          ;go set brightness timer then go display
; see if repeat timer running, don't increment if so
                movf    RepeatTimer+1,w
                iorwf   RepeatTimer+0,w
                btfss   STATUS,Z                ;can repeat
                goto    CSdone                  ;can't do anything
; increment if we can
                movf    HeatSetting,w
                sublw   MaxHeatSetting          ;at maximum?
                btfss   STATUS,Z                ;skip if so
                incf    HeatSetting,f
                goto    CSdisplayHeat           ;go display
CScheckDown
                btfsc   SwitchDownPort,SwitchDownBit ;skip if Down switch active
                goto    CSneitherSwitch
                call    SetDebounceTimer
; see if display timer running (do we already know what the setting is?)
                movf    ShowHeatTimer+1,w
                iorwf   ShowHeatTimer+0,w
                btfsc   STATUS,Z                ;skip if running
                goto    CSdisplayHeat1          ;go set brightness timer then go display
; see if repeat timer running, don't decrement if so
                movf    RepeatTimer+1,w
                iorwf   RepeatTimer+0,w
                btfss   STATUS,Z                ;can repeat
                goto    CSdone                  ;can't do anything
; decrement if we can
                movf    HeatSetting,w
                btfss   STATUS,Z                ;skip if zero
                decf    HeatSetting,f           ;decrement
;
CSdisplayHeat1


CSdisplayHeat
;set up to show setting
                movlw   low ShowHeatTime
                movwf   ShowHeatTimer+0        ;this'll hold off the adc display
                movlw   high ShowHeatTime
                movwf   ShowHeatTimer+1
; and set up BrightnessWaitTimer
                movlw   low BrightnessWaitTime
                movwf   BrightnessWaitTimer+0
                movlw   high BrightnessWaitTime
                movwf   BrightnessWaitTimer+1
                bsf     DisplaySwitchFlagWord,DisplaySwitchFlagBit
                call    DispClearMsDigit
; see if voltage too low
                btfss   VoltageTooLowFlagWord,VoltageTooLowFlagBit ;skip if voltage too low
                goto    CSdHok
                movlw   SegsForL
                call    DispLowerDigit
                movlw   0
                goto    CSgoSetting
CSdHok
                movlw   SegsForH
                call    DispLowerDigit
                movf    HeatSetting,w
CSgoSetting
                call    DecToSeg
                call    DispMiddleDigit
; set repeat timer
                movlw   low RepeatTime
                movwf   RepeatTimer+0        ;this'll hold off the adc display
                movlw   high RepeatTime
                movwf   RepeatTimer+1
                goto    CSdone
;
CSneitherSwitch
; if repeat timer is running then set debounce timer
                movf    RepeatTimer+1,w
                iorwf   RepeatTimer+0,w
                btfss   STATUS,Z                ;skip if not running
                call    SetDebounceTimer
; clear repeat timer
CSneitherClearRepeat
                clrf    RepeatTimer+1
```

```
                clrf    RepeatTimer+0
;
CSdone
;check heat display timer
                movf    ShowHeatTimer+1,w
                iorwf   ShowHeatTimer+0,w
                btfsc   STATUS,Z                ;skip if still running
                bcf     DisplaySwitchFlagWord,DisplaySwitchFlagBit ;clear flag
;
                BankFor 0
                return
;
SetDebounceTimer
                movlw   low DebounceTime
                movwf   DebounceTimer+0
                movlw   high DebounceTime
                movwf   DebounceTimer+1
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
WaitPulseTime
                BankFrc PulseTimer
                movwf   PulseTimer+0
                movlw   high PulseTime
                movwf   PulseTimer+1
; wait until timer timed out
PulseLoop
                movf    PulseTimer+1,w
                iorwf   PulseTimer+0,w
                btfss   STATUS,Z
                goto    PulseLoop
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CheckBrightnessLevel
; if BrightnessWaitTimer non-zero set full brightness else set dim
                movf    BrightnessWaitTimer+0,w
                iorwf   BrightnessWaitTimer+1,w
                movlw   MinBrightnessSetting
                btfss   STATUS,Z        ;skip if timer was zero
                movlw   MaxBrightnessSetting
                movwf   Brightness
                BankFor 0
                return

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       mResBlw HSDigitSegmentImage,1,UpperLimit ;segment image
;       mResBlw MSDigitSegmentImage,1,UpperLimit ;segment image
;       mResBlw LSDigitSegmentImage,1,UpperLimit ;segment image
DispClearMsDigit
                BankFrc HSDigitSegmentImage
                clrf    HSDigitSegmentImage
                BankFor 0
                return
DispSetHsDigit
                BankFrc HSDigitSegmentImage
                bsf     HSDigitSegmentImage,HSDsegs
                BankFor 0
                return

DispClear
                call    DispClearMsDigit
                clrf    MSDigitSegmentImage
                clrf    LSDigitSegmentImage
                BankFor 0
                return
DispMiddleDigit
                BankFrc MSDigitSegmentImage
                movwf   MSDigitSegmentImage
                BankFor 0
                return
DispLowerDigit
                BankFrc LSDigitSegmentImage
                movwf   LSDigitSegmentImage
                BankFor 0
                return
;
DispDecimalPointOn
                BankFrc  MSDigitSegmentImage
                bsf    MSDigitSegmentImage,DpBit
                BankFor 0
                return

DispDecimalPointOff
                BankFrc  MSDigitSegmentImage
                bcf    MSDigitSegmentImage,DpBit
                BankFor 0
                return
;;;;;;;;;;;;
```

```
CheckAdc
;
; if switch displayed then do nothing
                btfsc   DisplaySwitchFlagWord,DisplaySwitchFlagBit ;skipi f not
                goto    CheckAdcEnd     ;do nothing

; if AdcTmr is zero then new adc reading
                movf    AdcTmr+0,w
                iorwf   AdcTmr+1,w
                btfss   STATUS,Z        ;skip if zero
                goto    CheckAdcEnd
                BankFor ADCON0
                bsf     ADCON0,GO
; set up timer, gives adc time to work
                movlw   low AdcTime
                BankFor AdcTmr+0
                movwf   AdcTmr+0
                movlw   high AdcTime
                BankFor AdcTmr+1
                movwf   AdcTmr+1
CAwaitDone
                btfsc   ADCON0,NOT_DONE
                goto    CAwaitDone
                BankFor ADRESL
                movf    ADRESL,w
; movlw 0x33
                BankFor AdcResult+0
                addwf   AdcResult+0,f
                BankFor ADRESH
                movf    ADRESH,w
; movlw 1
                btfsc   STATUS,C
                addlw   1
                BankFor AdcResult+1
                addwf   AdcResult+1,f
; if eeprom not got a reading in it ...
                movlw   low ((EE_AdcReading+1) & 0xffff)
                movwf   EepromAdr
                call    EepromRead
                sublw   0xff
                btfss   STATUS,Z
                goto    CAeeReadingDone
                movf    AdcResult+1,w
                movwf   EepromData
                call    EepromWrite
                decf    EepromAdr,f
                movf    AdcResult+0,w
                movwf   EepromData
                call    EepromWrite
CAeeReadingDone
;
                incf    AdcAverageCounter,f
                movf    AdcAverageCounter,w
                addlw   -AdcAverageCount
                btfss   STATUS,Z
                goto    CheckAdcEnd
;
; put in correction factor, allow factors with 1 in 256 resolution,
; so we add or subtract up to 128 256ths
;
; get the factor
                movlw   (EE_Scale & 0xff)
                movwf   EepromAdr
                call    EepromRead
                BankFrc ScaleFactor
                movwf   ScaleFactor
                movf    ScaleFactor,f           ;see if zero
                btfsc   STATUS,Z
                goto    ScalingDone
;
; get adc value/256
                movf    AdcResult+0,w
                movwf   Fraction+0
                movf    AdcResult+1,w
                movwf   Fraction+1
                clrf    AdcFraction             ;ready for fractional part
;
                btfsc   ScaleFactor,7           ;skip if positive
                goto    ScaleFactorNegative
; scale factor positive, add count to value
ScalePositiveLoop
                movf    Fraction+0,w
                addwf   AdcFraction,f
                movf    Fraction+1,w
                btfsc   STATUS,C
                addlw   1
                addwf   AdcResult+0,f
                btfsc   STATUS,C
                incf    AdcResult+1,f
```

```
                decfsz  ScaleFactor,f
                goto    ScalePositiveLoop
                goto    ScalingDone
ScaleFactorNegative
ScaleNegativeLoop
                movf    Fraction+0,w
                subwf   AdcFraction,f
                movf    Fraction+1,w
                btfss   STATUS,C                ;skip if carry = no borrow
                addlw   1                       ;borrow
                subwf   AdcResult+0,f
                btfss   STATUS,C                ;skip if carry = no borrow
                decf    AdcResult+1,f
                incf    ScaleFactor,f
                btfss   STATUS,Z
                goto    ScaleNegativeLoop

ScalingDone

; going to divide by 64 to display, but remember if to go past the half ...
                movlw   0               ;assume less than half
                btfsc   AdcResult+0,5
                movlw   1
; move down by 6 bits
                bcf     STATUS,C
                rrf     AdcResult+1,f   ;1
                rrf     AdcResult+0,f
                bcf     STATUS,C
                rrf     AdcResult+1,f   ;2
                rrf     AdcResult+0,f
                bcf     STATUS,C
                rrf     AdcResult+1,f   ;3
                rrf     AdcResult+0,f
                bcf     STATUS,C
                rrf     AdcResult+1,f   ;4
                rrf     AdcResult+0,f
                bcf     STATUS,C
                rrf     AdcResult+1,f   ;5
                rrf     AdcResult+0,f
                bcf     STATUS,C
                rrf     AdcResult+1,f   ;6
                rrf     AdcResult+0,f
                addwf   AdcResult+0,f
                btfsc   STATUS,C
                incf    AdcResult+1,f
; converg to 1/10ths of a volt
; result is in AdcResult[1:0], 0x3ff=199 (0x400=200)
; need to multiply result by 200 and divide by 0x400
; or multiply by 50 and divide by 256
;
; 50=x32+x16+x2
;
                BankFrc Accum
                clrf    Accum+0
                clrf    Accum+1
; get x2
                bcf     STATUS,C
                rlf     AdcResult+0,f   ;}x2 in AdcResult
                rlf     AdcResult+1,f   ;}
                movf    AdcResult+0,w
                movwf   Accum+0
                movf    AdcResult+1,w
                movwf   Accum+1
                bcf     STATUS,C
                rlf     AdcResult+0,f   ;}x4 in AdcResult
                rlf     AdcResult+1,f   ;}
                bcf     STATUS,C
                rlf     AdcResult+0,f   ;}x8 in AdcResult
                rlf     AdcResult+1,f   ;}
                bcf     STATUS,C
                rlf     AdcResult+0,f   ;}x16 in AdcResult
                rlf     AdcResult+1,f   ;}
                movf    AdcResult+0,w
                addwf   Accum+0,f
                movf    AdcResult+1,w
                btfsc   STATUS,C
                addlw   1
                addwf   Accum+1,f
                bcf     STATUS,C
                rlf     AdcResult+0,f   ;}x32 in AdcResult
                rlf     AdcResult+1,f   ;}
                movf    AdcResult+0,w
                addwf   Accum+0,f
                movf    AdcResult+1,w
                btfsc   STATUS,C
                addlw   1
                addwf   Accum+1,f

;
```

```
;if voltage is too low, don't allow heater
;
; get limit
                movlw   low EE_MinVoltage
                movwf   EepromAdr
                call    EepromRead
; if > 200 then stupid or unprogrammed
                movlw   200
                subwf   EepromData,w
                btfss   STATUS,C
                goto    NotSillyValue
                movlw   kLowVoltDefault
                movwf   EepromData

NotSillyValue
                movf    Accum+1,w         ;get value
                subwf   EepromData,w
                btfsc   STATUS,C        ;skip if ok
                goto    CAtooLow
                bcf     VoltageTooLowFlagWord,VoltageTooLowFlagBit
                goto    CAtestTooLowDone
CAtooLow
                bsf     VoltageTooLowFlagWord,VoltageTooLowFlagBit
; if the heat request level >0, we want top periodically display "0L".
                movf    HeatSetting,f
                btfsc   STATUS,Z        ;skip if not zero
                goto    CAtestTooLowDone
; we'll display "0L" (say) 1/2 second every 4 seconds (or near offer),
; if (MillisecondsCounter[ms] & 0x06) == 0 then display "0L" else normal voltage display
                movf    MillisecondsCounter+1,w
                andlw   0x0e
                btfss   STATUS,Z        ;skip if in "0L" period
                goto    CAtestTooLowDone ;else go display adc
;
                call    DispClearMsDigit
                movlw   SegsFor0
                call    DispMiddleDigit
                movlw   SegsForL
                call    DispLowerDigit
                call    DispDecimalPointOff
                goto    CAdisplayDone
;
CAtestTooLowDone
;
                call    DisplayAdcResult
;
CAdisplayDone
                clrf    AdcAverageCounter;ready for next time
                clrf    AdcResult+1
                clrf    AdcResult+0
CheckAdcEnd
                BankFrc 0
                return

DisplayAdcResult
;

; now convert to decimal
;
;100's
                movlw   -1
                movwf   BuildDec
Build100Loop
                incf    BuildDec,f
                movlw   -100
                addwf   Accum+1,f
                btfsc   STATUS,C
                goto    Build100Loop
; done 100s, correct accum
                movlw   100
                addwf   Accum+1,f
; put up ist digit (can only be 0 or 1)
                movf    BuildDec,w
                btfss   STATUS,Z        ;skip if zero
                goto    Build100notzero
                call    DispClearMsDigit
                goto    Build100Done
Build100notzero
                call    DispSetHsDigit
Build100Done
; tens
                movlw   -1
                movwf   BuildDec
Build10Loop
                incf    BuildDec,f
                movlw   -10
                addwf   Accum+1,f
                btfsc   STATUS,C        ;skip if overflow
                goto    Build10Loop
```

```
                movlw   10              ;correct
                addwf   Accum+1,f
                movf    BuildDec,w
                call    DecToSeg
                call    DispMiddleDigit
                movf    Accum+1,w       ;whats left
                call    DecToSeg
                call    DispLowerDigit
                call    DispDecimalPointOn
                BankFor 0
                return
;



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Timer2IrqHandler
;
                bcf     PIR1,TMR2IF     ;clear the flag
;;;;;;;;; diag stuff ;;;;;;;;;;;;;;
;;;;;;;;; end diag stuff ;;;;;;;;;;;;;;
; if heat compare timer is zero, preset it
                movf    HeatCompareTimer+1,w
                iorwf   HeatCompareTimer+0,w
                btfss   STATUS,Z        ;skipi f zero
                goto    T2HCTnotZero
                movlw   MaxHeatSetting
                movwf   HeatCompareTimer+1
T2HCTnotZero
; increment our milliseconds counter
                incf    MillisecondsCounter+0,f
                btfsc   STATUS,Z
                incf    MillisecondsCounter+1,f
;; deal with dual timers
                movlw   DoubleTimers
                movwf   FSR             ;fsr: [ls]
T2DecDblLoop
; check if zero
                movf    INDF,w          ;[ls]
                incf    FSR,f           ;[ls-ms]
                iorwf   INDF,w          ;[ms]
                btfsc   STATUS,Z        ;[ms]skip if not zero
                goto    T2DblNext       ;[ms]
                decf    FSR,f           ;[ls-ls]point to ls byte
                decf    INDF,f          ;[ls]decr ls byte
                comf    INDF,w          ;[ls]see if 0xff
                btfss   STATUS,Z        ;[lsskip if so
                goto    T2DecDblNoOverflow ;[ls]
                incf    FSR,f           ;[ls-ms]point down to ms
                decf    INDF,f          ;[ms]
T2DblNext
                decf    FSR,f           ;[ms-ls]point back for commion stuff
T2DecDblNoOverflow
                incf    FSR,f           ;[ls-ms]
                incf    FSR,f           ;[ms-next ls] point to next counter
                movlw   DoubleTimersEnd
                subwf   FSR,w
                btfss   STATUS,Z
                goto    T2DecDblLoop
; single byte timers
                movlw   SingleTimers
                movwf   FSR
T2DecSngLoop
                movf    INDF,f          ;zero?
                btfss   STATUS,Z        ;skip if zero
                decf    INDF,f          ;else decrement
                incf    FSR,f           ;bump pointer
                movlw   SingleTimersEnd ;at the end?
                subwf   FSR,w           ;
                btfss   STATUS,Z        ;skip if at end
                goto    T2DecSngLoop    ;else loop
; if seconds timer is zero then initialise
                movf    Second+0,w
                iorwf   Second+1,w
                btfss   STATUS,Z        ;skip if zero
                goto    T2SecNotZero
                movlw   low 1000
                movwf   Second+0
                movlw   high 1000
                movwf   Second+1
T2SecNotZero
; now for the compare timer thingy
; if voltage too low, take it off
                btfss   VoltageTooLowFlagWord,VoltageTooLowFlagBit ;skip if too low
                goto    T2VnotTooLow
                bcf     HeatPort,HeatBit ;take heat off
                goto    T2HtrDone
T2VnotTooLow
; now if heat setting higher than ms byte of timer then turn heat on else turn heat off
```

```
; so take timer from setting; if borrow then turn off
                movf    HeatSetting,w
                subwf   HeatCompareTimer+1,w
                btfss   STATUS,C        ;skip if no borrow
                goto    T2Vborrow
                bcf     HeatPort,HeatBit ;turn heat on
                goto    T2HtrDone
T2Vborrow
                bsf     HeatPort,HeatBit ;turn heat on
T2HtrDone
; brightness. images held in:
; goto DisplayOn
; if brightness setting higher than its timer then turn display on else turn display off
;
; increment timer
                incf    BrightnessTimer,f
                movf    BrightnessTimer,w
                sublw   MaxBrightnessSetting
                btfsc   STATUS,Z
                clrf    BrightnessTimer
; if setting > timer turn display on
                movf    Brightness,w
                btfsc   STATUS,Z                ;mustn't be zero
                incf    Brightness,f            ;we'll pick it up next time
                subwf   BrightnessTimer,w
                btfsc   STATUS,C                ;skip if borrow, brightness > timer
                goto    T2DispOff
DisplayOn
; put display on
                movf    HSDPort,w
                iorlw   (1<<HSDsegs)            ;take high to turn off
                xorwf   HSDigitSegmentImage,w   ;take low any on segments
                movwf   HSDPort
                movf    MSDPort,w
                iorlw   SegsFor8 | 1<<DpBit     ;take high to turn off
                xorwf   MSDigitSegmentImage,w   ;take low any on segments
                movwf   MSDPort
                movf    LSDPort,w
                iorlw   SegsFor8               ;take high to turn off
                xorwf   LSDigitSegmentImage,w   ;take low any on segments
                movwf   LSDPort
                goto    T2DispDone
T2DispOff
                movf    HSDPort,w
                iorlw   (1<<HSDsegs)
                movwf   HSDPort
                movf    MSDPort,w
                iorlw   SegsFor8 | 1<<DpBit
                movwf   MSDPort
                movf    LSDPort,w
                iorlw   SegsFor8
                movwf   LSDPort
T2DispDone
; and exit
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
InitTimer2
; set up timer2 for 1mS irq
; Interval time of 1000uS
; if 4MHz crystal, want division on Timer2 of 4e6/4/4/1000=250,
; so set prescaler to 4, post-scaler to 1.
; alarm tone output we'll do "by hand" from the itq handler
; if 12MHz crystal, set prescaler to 4, post-scaler to 3
                movlw   2<<TOUTPS0|1<<T2CKPS0 ;postscale to 3, prescale to 4
                BankFor T2CON
                movwf   T2CON
Timer2Preset    equ     PicClkIn/4/4/3000-1
                movlw   Timer2Preset
                BankFor PR2
                movwf   PR2
                BankFor TMR2
                clrf    TMR2
                BankFor PIE1
                bsf     PIE1,TMR2IE             ;enable timer2 irqs
                BankFor PIR1
                bcf     PIR1,TMR2IF             ;clear timer2 flag
                BankFor T2CON
                bsf     T2CON,TMR2ON            ;enable timer2
;
                movlw   1
                movwf   Second+0                ;start near zero
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
InitAdc
; bit 0 used as voltage input
; all others digital io
                movlw   0xe<<PCFG0 | 1<<ADFM    ;ch0 adc & right justrify
```

```
                BankFrc ADCON1
                movwf   ADCON1
; adcclock:
;                         8Tos     CH0
                movlw   1<<ADCS0|0<<CHS0
                BankFor ADCON0
                movwf   ADCON0
; enable
                bsf     ADCON0,ADON
;
; start conversion
                bsf     ADCON0,GO
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
InitDisplays
; init port:
                BankFrc HSDTris
                bcf     HSDTris,HSDsegs
                BankFor HSDPort
                bcf     HSDPort,HSDsegs ;on for now
                call    DispSetHsDigit          ;on
                movlw   low (~(SegsFor8 | 1<<DpBit))
                BankFor MSDTris
                andwf   MSDTris,f       ;outputs
                movlw   ~SegsFor8
                BankFor LSDTris
                andwf   LSDTris,f       ;outputs
                movlw   SegsFor8
                call    DispLowerDigit
                movlw   SegsFor8 | 1<<DpBit
                call    DispMiddleDigit
; brightness to maximum
                movlw   MaxBrightnessSetting
                BankFor Brightness
                movwf   Brightness
;
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
InitHeater
                BankFrc HeatTris
                bcf     HeatTris,HeatBit
                BankFor HeatPort
                bcf     HeatPort,HeatBit
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
InitSwitchPort
                BankFrc SwitchUpTris
                bsf     SwitchUpTris,SwitchUpBit
                BankFor SwitchDownTris
                bsf     SwitchDownTris,SwitchDownBit
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
InitPulse
                BankFrc PulseTris
                bcf     PulseTris,PulseBit
                BankFor PulsePort
                bcf     PulsePort,PulseBit
; and pre-pulse time
                movlw   low PrePulseTime
                BankFor PulseTimer+0
                movwf   PulseTimer+0
                movlw   high PrePulseTime
                BankFor PulseTimer+1
                movwf   PulseTimer+1
                BankFor 0
                return
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EepromRead
; read a data byte from the EEPROM, the address in EepromAdr
; puts data in Eepromdata and in w

                BankFrc EepromAdr
                movfw   EepromAdr
                BankFor EEADR
                movwf   EEADR           ;write address
                BankFor EECON1
                bcf     EECON1,EEPGD    ;point to data memory
                bsf     EECON1,RD       ;initiate read
                BankFor EEDATA
                movfw   EEDATA          ;read data
                BankFor EepromData
                movwf   EepromData
                BankFor 0
                return                  ;exit, data in w
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
EepromWrite
; write data byte in EepromData to eeprom address in EepromAdr

                BankFrc EECON1
EWwaitWrite
                btfsc   EECON1,WR       ;skip if previous write finished
                goto    EWwaitWrite
                BankFor EepromAdr
                movfw   EepromAdr       ;get address
                BankFor EEADR
                movwf   EEADR           ;put address
                BankFor EepromData
                movfw   EepromData      ;get data
                BankFor EEDATA
                movwf   EEDATA          ;put data
                BankFor EECON1
                bcf     EECON1,EEPGD    ;point to data memory
                bsf     EECON1,WREN     ;enable writes
                movfw   INTCON          ;get old intcon status
                BankFor EeWrIntcon
                movwf   EeWrIntcon      ;save old interrupt status
                bcf     INTCON,GIE      ;freeze interrupts
                BankFor EECON1
                movlw   0x55
                movwf   EECON2          ;arm write
                movlw   0xaa
                movwf   EECON2          ;arm write
                bsf     EECON1,WR       ;start write
                BankFor EeWrIntcon
                movfw   EeWrIntcon      ;get old interrupt status
                movwf   INTCON          ;restore interrupt status
                BankFor EECON1
                bcf     EECON1,WREN     ;disable writes
                BankFor EECON1
EWwaitWrite1
                btfsc   EECON1,WR       ;skip if previous write finished
                goto    EWwaitWrite1
                BankFor 0
                return

;
                end
```