```
;Macro Definitions

;Skip_If_ZERO Skip_If_NOT_ZERO
;Set_CARRY Clear_CARRY Skip_If_CARRY_SET Skip_If_CARRY_CLR
;Skip_If_vRestart Skip_If_NOT_vRestart
;Set_vAbove Clear_vAbove Skip_If_vAbove Skip_If_NOT_vAbove Skip_If_Old_vAbove
Skip_If_Old_NOT_vAbove
;Set_vBelow Clear_vBelow Skip_If_vBelow Skip_If_NOT_vBelow Skip_If_Old_vBelow
Skip_If_Old_NOT_vBelow
;Set_vInc Clear_vInc Skip_If_vInc Skip_If_Old_vInc Skip_If_NOT_vInc
;Set_vDec Clear_vDec Skip_If_vDec Skip_If_Old_vDec Skip_If_NOT_vDec
;Set_vAccel Clear_vAccel Skip_If_vAccel Skip_If_Old_vAccel Skip_If_NOT_vAccel
;Set_vDecel Clear_vDecel Skip_If_vDecel Skip_If_Old_vDecel
;Set_vChgMax Clear_vChgMax Skip_If_vChgMax Skip_If_NOT_vChgMax
;Clear_vTypeK Skip_If_vTypeK Skip_If_NOT_vTypeK
;Set_BUTTON1 Clear_BUTTON1 Skip_If_BUTTON1
;Set_BUTTON2 Clear_BUTTON2 Skip_If_BUTTON2
;Set_BUTTON_PRESS Clear_BUTTON_PRESS
;Set_vDCnew Clear_vDCnew Skip_If_vDCnew
;Set_vDCzero Clear_vDCzero Skip_If_vDCzero Skip_If_NOT_vDCzero
;Set_vDCmax Clear_vDCmax Skip_If_vDCmax Skip_If_NOT_vDCmax
;Set_vDCsp Clear_vDCsp
;Set_vDCsign Clear_vDCsign Skip_If_vDCsign Skip_If_NOT_vDCsign

;   Clear_Bytes (address, count)
;   Clear_Counter (counter_num)
;   Increment_Counter (counter_num)
;   Disable_Interrupts
;   Enable_Interrupts
;   Disable_GPIO
;   Initialize_GPIO
;   Enable_LED
;   Initialize_Oscillator
;   Disable_PWM
;   Initialize_PWM
;   Initialize_PID
;   Initialize_eeDUTYCYCLEptr
;   Advance_eeDUTYCYCLEptr
;   Initialize_SETPOINT
;   Reset_COILTEMPdelay
;   Initialize_DCBIAS

;   Set_LED_delay_count
;   Initialize_LED_WDT
;   Initialize_LED_ZeroReading
;   Initialize_LED_Sensor
;   Initialize_LED_TooHot
;   Initialize_LED_RunTime
;   Initialize_LED_Battery
;   Initialize_LED_vWait

;   Initialize_vSTATE
;   Initialize_vSENSOR
;   Initialize_vERROR
;   Initialize_vCOILTEMP
;   Initialize_vPWM
```

```
;   Initialize_Start_Time
;   Initialize_Restart_Time
;   Clear_eePROM_Buffers
;
;   Save_W_STATUS
;   Restore_W_STATUS
;
;   Set_vSTATE (bit), Clear_vSTATE (bit), Wait_vSTATE (bit)
;   Set_vSENSOR (bit), Clear_vSENSOR (bit), Wait_vSENSOR (bit)
;   Set_vPWM (bit), Clear_vPWM (bit)
;   Set_vERROR (bit), Clear_vERROR (bit)
;   Set_vCOILTEMP (bit), Clear_vCOILTEMP (bit)

;   If_Heartbeat_Interrupt "THEN"
;   Clear_Heartbeat_Interrupt_Flag
;   Initialize_Heartbeat
;   Start_Heartbeat
;   Stop_Heartbeat

;   Initialize_Watchdog
;   If_WDT_Reset_Then_Shutdown
;   Reset_Watchdog
;   If_vRun "THEN"

;   Read_Button
;   Wait_Release_BTN (btn#)
;   Reset_BTNdelay (ticks)
;   Clear_BTNcount
;   Increase_SETPOINT (units)
;   Decrease_SETPOINT (units)
;   If_BUTTON_Adjust_SETPOINT

;   Update_Duty_Cycle
;   Initialize_COILTEMPptr
;   Advance_COILTEMPptr
;   Initialize_eeCOILTEMPptr
;   Advance_eeCOILTEMPptr
;   Read_Coil_Temperature
;   Signal_SETPOINT

;   If_Maximum_Run_Time "THEN_GOTO" addr
;--------------------

Skip_If_ZERO macro
    btfss   ZERO
    endm
Skip_If_NOT_ZERO macro
    btfsc   ZERO
    endm
;--------------------
Set_CARRY macro
    bsf     CARRY
    endm
Clear_CARRY macro
    bcf     CARRY
    endm
```

```
Skip_If_CARRY_SET macro
    btfss   CARRY
    endm
Skip_If_CARRY_CLR macro
    btfsc   CARRY
    endm
;--------------------
Skip_If_vRestart macro
    btfss   vSTATE,vRestart
    endm
Skip_If_NOT_vRestart macro
    btfsc   vSTATE,vRestart
    endm
;--------------------
Set_vAbove macro
    bsf     vCOILTEMP,vAbove
    endm
Clear_vAbove macro
    bcf     vCOILTEMP,vAbove
    endm
Skip_If_vAbove macro
    btfss   vCOILTEMP,vAbove
    endm
Skip_If_NOT_vAbove macro
    btfsc   vCOILTEMP,vAbove
    endm
Skip_If_Old_vAbove macro
    btfss   vCOILTEMPold,vAbove
    endm
Skip_If_Old_NOT_vAbove macro
    btfsc   vCOILTEMPold,vAbove
    endm
;--------------------
Set_vBelow macro
    bsf     vCOILTEMP,vBelow
    endm
Clear_vBelow macro
    bcf     vCOILTEMP,vBelow
    endm
Skip_If_vBelow macro
    btfss   vCOILTEMP,vBelow
    endm
Skip_If_NOT_vBelow macro
    btfsc   vCOILTEMP,vBelow
    endm
Skip_If_Old_vBelow macro
    btfss   vCOILTEMPold,vBelow
    endm
Skip_If_Old_NOT_vBelow macro
    btfsc   vCOILTEMPold,vBelow
    endm
;--------------------
Set_vInc macro
    bsf     vCOILTEMP,vInc
    endm
Clear_vInc macro
```

```
    bcf     vCOILTEMP,vInc
    endm
Skip_If_vInc macro
    btfss   vCOILTEMP,vInc
    endm
Skip_If_Old_vInc macro
    btfss   vCOILTEMPold,vInc
    endm
Skip_If_NOT_vInc macro
    btfsc   vCOILTEMP,vInc
    endm
;--------------------
Set_vDec macro
    bsf     vCOILTEMP,vDec
    endm
Clear_vDec macro
    bcf     vCOILTEMP,vDec
    endm
Skip_If_vDec macro
    btfss   vCOILTEMP,vDec
    endm
Skip_If_Old_vDec macro
    btfss   vCOILTEMPold,vDec
    endm
Skip_If_NOT_vDec macro
    btfsc   vCOILTEMP,vDec
    endm
;--------------------
Set_vAccel macro
    bsf     vCOILTEMP,vAccel
    endm
Clear_vAccel macro
    bcf     vCOILTEMP,vAccel
    endm
Skip_If_vAccel macro
    btfss   vCOILTEMP,vAccel
    endm
Skip_If_Old_vAccel macro
    btfss   vCOILTEMPold,vAccel
    endm
Skip_If_NOT_vAccel macro
    btfsc   vCOILTEMP,vAccel
    endm
;--------------------
Set_vDecel macro
    bsf     vCOILTEMP,vDecel
    endm
Clear_vDecel macro
    bcf     vCOILTEMP,vDecel
    endm
Skip_If_vDecel macro
    btfss   vCOILTEMP,vDecel
    endm
Skip_If_Old_vDecel macro
    btfss   vCOILTEMPold,vDecel
    endm
```

```
;--------------------
Set_vChgMax macro
    bsf     vCOILTEMP,vChgMax
    endm
Clear_vChgMax macro
    bcf     vCOILTEMP,vChgMax
    endm
Skip_If_vChgMax macro
    btfss   vCOILTEMP,vChgMax
    endm
Skip_If_NOT_vChgMax macro
    btfsc   vCOILTEMP,vChgMax
    endm
;--------------------
Clear_vTypeK macro
    bcf     vSENSOR,vTypeK
    endm
Skip_If_vTypeK macro
    btfss   vSENSOR,vTypeK
    endm
Skip_If_NOT_vTypeK macro
    btfsc   vSENSOR,vTypeK
    endm
;--------------------
Set_BUTTON1 macro
    bsf     vSENSOR,vBtn1
    endm
Clear_BUTTON1 macro
    bcf     vSENSOR,vBtn1
    endm
Skip_If_BUTTON1 macro
    btfss   vSENSORold,vBtn1
    endm
Set_BUTTON2 macro
    bsf     vSENSOR,vBtn2
    endm
Clear_BUTTON2 macro
    bcf     vSENSOR,vBtn2
    endm
Skip_If_BUTTON2 macro
    btfss   vSENSORold,vBtn2
    endm
Set_BUTTON_PRESS macro
    bsf     vSENSOR,vBtnON
    endm
Clear_BUTTON_PRESS macro
    bcf     vSENSOR,vBtnON
    endm
;--------------------
Set_vDCnew macro
    bsf     vPWM,vDCnew
    endm
Clear_vDCnew macro
    bcf     vPWM,vDCnew
    endm
Skip_If_vDCnew macro
```

```
    btfss   vPWM,vDCnew
    endm
;--------------------
Set_vDCzero macro
    bsf     vPWM,vDCzero
    endm
Clear_vDCzero macro
    bcf     vPWM,vDCzero
    endm
Skip_If_vDCzero macro
    btfss   vPWM,vDCzero
    endm
Skip_If_NOT_vDCzero macro
    btfsc   vPWM,vDCzero
    endm
;--------------------
Set_vDCmax macro
    bsf     vPWM,vDCmax
    endm
Clear_vDCmax macro
    bcf     vPWM,vDCmax
    endm
Skip_If_vDCmax macro
    btfss   vPWM,vDCmax
    endm
Skip_If_NOT_vDCmax macro
    btfsc   vPWM,vDCmax
    endm
;--------------------
Set_vDCsp macro
    bsf     vPWM,vDCsp
    endm
Clear_vDCsp macro
    bcf     vPWM,vDCsp
    endm
;--------------------
Set_vDCsign macro
    bsf     vPWM,vDCsign
    endm
Clear_vDCsign macro
    bcf     vPWM,vDCsign
    endm
Skip_If_vDCsign macro
    btfss   vPWM,vDCsign
    endm
Skip_If_NOT_vDCsign macro
    btfsc   vPWM,vDCsign
    endm

;******************
;Clear_Counter macro num
;   movlw   num
;   call    eeClrCounterW
;   endm
;
;Increment_Counter macro num
```

```
;   movlw   num
;   call    eeIncCounterW
;   endm
;******************

;------------------
Clear_Bytes macro adr, cnt
    banksel adr
    movlw   cnt
     movwf  LOOPcnt
    movlw   adr
     movwf  FSR
;loop:
    clrf    INDF
    incf    FSR
    decfsz  LOOPcnt
    goto    $-3
    endm

;------------------
; Disable all interrupts
Disable_Interrupts macro
    call    DisableInterrupts
    endm

; Enable all interrupts
Enable_Interrupts macro
    call    EnableInterrupts
    endm

; Disable GPIO
Disable_GPIO macro
    banksel GPIO
    clrf    GPIO        ;set all GPx to '0' level
    banksel ANSEL
    clrf    ANSEL       ;set digital I/O
    banksel TRISIO
    movlw   b'00111111'
    movwf   TRISIO      ;set pins as input
    banksel CMCON0
    clrf    CMCON0      ;disable Comparator
    endm

; Initialize GPIO
Initialize_GPIO macro
    banksel GPIO
    clrf    GPIO        ;set all GPx output to '0' level

    ;disable comparator
    movlw   0x0F        ;only digital I/O on pins
    movwf   CMCON0      ;comparator OFF

    ;set analog input on AN1/GP1
    banksel ANSEL
    clrf    ANSEL       ;set digital I/O
    movlw   b'00010010' ;Fosc/8 & AN1 analog input
```

```
      movwf ANSEL
    ;enable input on GP1, GP3
    movlw   b'00001010' ;- - o o i o i o
      movwf TRISIO      ;GP  5 4 3 2 1 0

    ;make sure MAX6675 chip select (CS) is off (HI)
    banksel GPIO
    bsf     CS          ;disable CS
    endm

; Turn ON the LED pin for OUTPUT
Enable_LED macro
    banksel TRISIO
    bcf     TRISIO, GP0
    endm

;------------------
; Initialize the internal oscillator for 4MHz
Initialize_Oscillator macro
    banksel OSCCON
    movlw   b'01100001' ;default: internal osc @ 4 MHz
    movwf   OSCCON
    endm

;disable PWM
Disable_PWM macro
    banksel GPIO
    bcf     PWM         ;set GP2 to 0
    clrf    T2CON       ;stop Timer2
    clrf    CCP1CON     ;clear duty cycle 2 LSB
    clrf    CCPR1L      ;clear duty cycle 8 MSB

    banksel TRISIO
    ;bsf    TRISIO,GP2  ;set GP2 to input
    clrf    PR2         ;set PWM period to zero
    endm

; Initialize PWM
    ;PR2
    ;T2CON
    ;CCPR1L
    ;CCP1CON<5:4> = LSB
    ;CCPR1L = MSB
    ;frequency = 1000 Hz (1 msec per tick)
    ;T2CON<prescaler> = '01' = 1:4          1000 Hz
    ;T2CON<prescaler> = '11' = 1:16          250 Hz
    ;T2CON<postscaler> = '0000' = 1:1
    ;PR2 = 249
    ;duty register = 0 (0%) to 1000 (100%)
    ;duty step = 0.1%
Initialize_PWM macro
;Setup for PWM Operation from page 82

    ;(1) disable PWM pin - set TRISIO,GP2
    banksel TRISIO
    bsf TRISIO,GP2
```

```
    ;(2) set PWM period by loading PR2
    movlw   .249        ;set TMR2 period
    movwf   PR2

    ;(3) configure CCP for PWM by loading CCP1CON
    banksel CCP1CON
    movlw   b'00001100' ;load PWM duty cycle LSB='00'
    movwf   CCP1CON     ;and configure CCP for PWM

    ;(4) finish setting duty cycle in CCPR1L
    clrf    CCPR1L      ;initial duty cycle set to 0

    ;(5) Configure and start Timer2
    banksel PIE1
    bcf     PIE1,TMR2IE     ;disable Timer2 interrupt
;   bsf     PIE1,TMR2IE     ;enable Timer2 interrupt
    banksel PIR1
    bcf     PIR1,TMR2IF     ;clear Timer2 interrupt flag
;   movlw   b'00000101'     ;-,0000,TMR2 ON, prescaler=1:4  1000 Hz
    movlw   b'00000111'     ;-,0000,TMR2 ON, prescaler=1:16  250 Hz
    movwf   T2CON           ;load prescaler value

    ;(6) Enable PWM after new PWM cycle
init_pwm_1:
    btfss   PIR1,TMR2IF     ;wait until Timer2 overflows
      goto  init_pwm_1
    banksel TRISIO
    bcf     TRISIO,GP2      ;set GP2 to output (PWM)
    endm

;------------------
;initialize PID variables
Initialize_PID macro
    banksel SPERROR
    Clear_Bytes cCOILTEMPstart, cCOILTEMPbytes
    Clear_Bytes cSPERRORadr, .9
    movlw   cSPERRORcnt     ;reset counter
      movwf SPERRORcnt
    endm

;------------------
; Initialize eeDUTYCYCLE buffer pointer
Initialize_eeDUTYCYCLEptr macro
    banksel eeDUTYCYCLEptr
    movlw   eeDUTYCYCLEfirst
    movwf   eeDUTYCYCLEptr
    endm

;advance eeDUTYCYCLE buffer pointer by 2
Advance_eeDUTYCYCLEptr macro
    banksel eeDUTYCYCLEptr
    incf    eeDUTYCYCLEptr
    incf    eeDUTYCYCLEptr
    btfsc   eeDUTYCYCLEptr,.7   ;bit clear means buffer wrap
      goto  $+3
```

```
    ;bit clear: wrap eeDUTYCYCLE pointer
    movlw   eeDUTYCYCLEfirst    ;first address in buffer
    movwf   eeDUTYCYCLEptr
    endm

;retrieve saved SETPOINT from eePROM
Initialize_SETPOINT macro
    call    eeGetSETPOINT
    endm

;set ticks to next COILTEMP reading
Reset_COILTEMPdelay macro
    movlw   .3
    movwf   COILTEMPdelay
    endm

;Get DCBIAS from eePROM (0x007D = 125/1023 = 12.2%)
Initialize_DCBIAS macro
    call    eeGetDCBIAS
    endm

;Get DCBIAS from eePROM (0x007D = 125/1023 = 12.2%)
Initialize_SPERRORtrip macro
    call    eeGetSPERRORtrip
    endm

;==================
;Set LED to flash 'count' times every 'delay' ticks
Set_LED_delay_count macro delay, count
    banksel FLASHdelay0
    movlw   delay
      movwf FLASHdelay0
    movlw   count
      movwf FLASHcount0
    call ResetLED
    endm

Initialize_LED_WDT macro
    Set_LED_delay_count .5, .1
    endm

Initialize_LED_ZeroReading macro
    Set_LED_delay_count .5, .2
    endm

Initialize_LED_Sensor macro
    Set_LED_delay_count .20, .1
    endm

Initialize_LED_TooHot macro
    Set_LED_delay_count .20, .2
    endm

Initialize_LED_RunTime macro
    Set_LED_delay_count .20, .3
    endm
```

```
Initialize_LED_Battery macro
    Set_LED_delay_count .20, .4
    endm

Initialize_LED_vWait macro
    Set_LED_delay_count .30, .3
    endm

;------------------
;clear the vSTATE byte
Initialize_vSTATE macro
    clrf    vSTATE
    endm

;clear the vSENSOR byte
Initialize_vSENSOR macro
    clrf    vSENSOR
    endm

;clear the vERROR byte
Initialize_vERROR macro
    banksel vERROR
    clrf    vERROR
    endm

;clear the vCOILTEMP byte
Initialize_vCOILTEMP macro
    clrf    vCOILTEMP
    endm
;clear the vPWM byte
Initialize_vPWM macro
    clrf    vPWM
    endm

;set the ticks until Shutdown
;set TOKEIT delay
Initialize_Start_Time macro
    banksel RUNTIME
    movlw   cRUNTIMElo
      movwf RUNTIME
    movlw   cRUNTIMEhi
      movwf RUNTIME+1
    movlw   cTOKEITlo           ;delay before green LED on
      movwf TOKEITdelay
    movlw   cTOKEIThi
      movwf TOKEITdelay+1
    endm

Initialize_Restart_Time macro
    banksel RUNTIME
    movlw   cEXTRATIMElo
      movwf RUNTIME
    movlw   cEXTRATIMEhi
      movwf RUNTIME+1
    clrf    TOKEITdelay         ;restart
```

```
    clrf    TOKEITdelay+1
    endm

;write FFs into eePROM
Clear_eePROM_Buffers macro
    call    eeClearCOILTEMPbuffer
    call    eeClearDUTYCYCLEbuffer
    endm

;------------------
;save W and STATUS on interrupt
Save_W_STATUS macro
    movwf   W_save          ;push W
    swapf   STATUS,W        ;move STATUS to W without affecting Z
    movwf   STATUS_save     ;push STATUS
    endm

;restore W and STATUS after interrupt
Restore_W_STATUS macro
    swapf   STATUS_save,W   ;pop STATUS register
    movwf   STATUS          ;restore pre-isr STATUS register contents
    swapf   W_save          ;two instructions to pop W from F ...
    swapf   W_save,W        ;... without changing STATUS,Z like movf does
    endm

;===== Vaporizer State Variable =====

;set a vSTATE bit
Set_vSTATE macro bit
    bsf     vSTATE,bit
    endm

;clear a vSTATE bit
Clear_vSTATE macro bit
    bcf     vSTATE,bit
    endm

Wait_vSTATE macro bit
    btfss   vSTATE,bit
      goto  $-1
    endm

;set a vSENSOR bit
Set_vSENSOR macro bit
    bsf     vSENSOR,bit
    endm

;clear a vSENSOR bit
Clear_vSENSOR macro bit
    bcf     vSENSOR,bit
    endm


;wait for vSENSOR bit
Wait_vSENSOR macro bit
    btfss   vSENSOR,bit
```

```
      goto  $-1
    endm

;set a vERROR bit
Set_vERROR macro bit
    banksel vERROR
    bsf     vERROR,bit
    endm

;clear a vERROR bit
Clear_vERROR macro bit
    banksel vERROR
    bcf     vERROR,bit
    endm

;set a vPWM bit
Set_vPWM macro bit
    bsf     vPWM,bit
    endm

;clear a vPWM bit
Clear_vPWM macro bit
    bcf     vPWM,bit
    endm

;set a vCOILTEMP bit
Set_vCOILTEMP macro bit
    bsf     vPWM,bit
    endm

;clear a vCOILTEMP bit
Clear_vCOILTEMP macro bit
    bcf     vPWM,bit
    endm

;===== Heartbeat =====

;testing for Heartbeat interrupt (TIMER1)
If_Heartbeat_Interrupt macro THEN
    banksel PIR1
    btfss   PIR1, TMR1IF
      goto  intr_exit
    endm

Clear_Heartbeat_Interrupt_Flag macro
    banksel PIR1
    bcf     PIR1,TMR1IF
    endm

;Initialize Heartbeat Timer (Timer1) - 10 ticks per second
; -, ignore gate, 1:2 prescaler (01), LPosc off, -, Fosc/4, OFF
Initialize_Heartbeat macro
    banksel T1CON
    movlw   b'00010000'
    movwf   T1CON
    endm
```

```
; Stop Heartbeat Ticker
Stop_Heartbeat macro
    banksel T1CON
    bcf     T1CON,TMR1ON    ;stop Timer1
    endm

; Start Heartbeat Ticker (reinit counter)
Start_Heartbeat macro
    ;stop Timer1 during setup
    Stop_Heartbeat

    ;initialize Timer1 counter
    banksel TMR1L
    movlw   cHeartbeatLO    ;load counter with 15536
    movwf   TMR1L
    movlw   cHeartbeatHI    ; = overflow in 100.001 msec
    movwf   TMR1H

    ;enable interrupt on rollover
    banksel PIE1
    bsf     PIE1,TMR1IE
    bsf     INTCON,PEIE
    bsf     INTCON,GIE

    ;start the Timer1
    banksel T1CON
    bcf     PIR1,TMR1IF     ;clear interrupt flag
    bsf     T1CON,TMR1ON    ;start Timer1
    endm

;===== Watchdog Timer =====

; Initialize WatchDog Timer to 4.5 min timeout
; WDT enabled in CONFIG so it is always running
;prescaler= 1:128, postscaler= 1:65536
;timeout = .00003226 * 65536 * 128 = 270.6 sec = 4.5 min
Initialize_Watchdog macro
    clrwdt
    banksel WDTCON
    movlw   cWDT5min_pre    ;1:65536 prescaler
    movwf   WDTCON
    banksel OPTION_REG
    movlw   cWDT5min_post   ;1:128 postscaler
    movwf   OPTION_REG
    endm

;test for WDT reset and SHUT DOWN
If_WDT_Reset_Then_Shutdown macro
    btfss   STATUS,NOT_TO   ;~TO = 0 for WDT reset
      goto  Error_WDT       ;WDT timed out!!! "Danger Will Robinson"
    endm

; Reset WatchDog Timer
; clear wdt and reset postscaler for 256 msec timeout
; 16msec (1:512 reset value) * 16 = 256 msec
```

```
Reset_Watchdog macro
    clrwdt                  ;reset prescaler to 1:512 (16msec default)
    banksel OPTION_REG
    movlw   cWDT256ms_post  ;set postscaler to 1:16 (256 msec)
    movwf   OPTION_REG
    endm

If_vRun macro THEN
    btfss   vSTATE,vRun
      goto  intr_done
    endm

;===== Button Press =====

;check for button press every tick
Read_Button macro

    call    ReadBTN
    call    SetBTNbit
    Set_vSENSOR vBtn    ;set ready flag
    endm

;wait for release of BTN=bit
Wait_Release_BTN macro bit

    ;wait for next button reading (each tick)
    Wait_vSENSOR vBtn       ;2 instr
    Clear_vSENSOR vBtn      ;1 instr

    ;test for release of BTN=bit
    btfsc   vSENSOR,bit     ;1 instr
      goto  $-4
    endm

;reset BTN delay for 'ticks' number of ticks
Reset_BTNdelay macro ticks
    movlw   ticks
    movwf   BTNdelay
    endm

;zero BTNcount to count number of ticks for BTN press
Clear_BTNcount macro
    banksel BTNcount
    clrf    BTNcount        ;prepare to time button press
    endm

;===== SETPOINT Adjustment =====

;increase SETPOINT temperature by "units"
;limit to cSPMAXlo,cSPMAXhi
;update eePROM value
Increase_SETPOINT macro units
    banksel SETPOINT
    movlw   units
    addwf   SETPOINT
    Skip_If_CARRY_CLR
```

```
     incf   SETPOINT+1

    ;test for max value
    movf    SETPOINT,W
    sublw   cSPMAXlo        ;lo byte
    movf    SETPOINT+1,W
    Skip_If_CARRY_SET
     incfsz SETPOINT+1,W    ;borrow
    sublw   cSPMAXhi        ;hi byte
    Skip_If_CARRY_CLR       ;C clear = too high
      goto  $+5

    ;limit SETPOINT to SETPOINTmax (cSPMAXlo,cSPMAXhi)
    movlw   cSPMAXlo
      movwf SETPOINT
    movlw   cSPMAXhi
      movwf SETPOINT+1
    endm

;decrease SETPOINT temperature by "units"
Decrease_SETPOINT macro units
    banksel SETPOINT
    movlw   units
    subwf   SETPOINT
    movlw   .0
    Skip_If_CARRY_SET           ;borrow?
     movlw  .1              ;Yes
    subwf   SETPOINT+1

    ;test if SETPOINT < 0
    Skip_If_CARRY_CLR
      goto  $+3
        clrf    SETPOINT
        clrf    SETPOINT+1
    endm

;test for button release after press, then adjust SETPOINT
If_BUTTON_Adjust_SETPOINT macro

    ;debounce BTN press
    btfss   vSENSOR,vBtnON      ;Is button still pressed?
    goto    main_btn_0          ;No, check history

    ;time BTN1 press
    ;if BTN1 > 32 ticks then ...
    ;   ... setup for Factory Default reset
    goto    run_DC              ;done, exit BTN processing

    ;check last button press
main_btn_0:
    btfss   vSENSORold,vBtnON   ;was button pressed before?
    goto    run_DC              ;No, so exit

    ;if BTN1 press - increase temperature by 5°F
main_btn_1:
    Skip_If_BUTTON1
```

```
      goto  main_btn_2
    Increase_SETPOINT cBUTTONinc
    goto    main_btn_3

    ;if BTN2 press - decrease temperature by 4°F
main_btn_2:
    Skip_If_BUTTON2
      goto  run_DC
    Decrease_SETPOINT cBUTTONdec
    goto    main_btn_3

;update SETPOINT in eePROM
main_btn_3:
    btfsc   vSTATE,vRun
      call  eePutSETPOINT
    goto    run_DC
    endm

;===== PWM Duty Cycle =====

;Update PWM's Duty Cycle 10-bit value
Update_Duty_Cycle macro

    ;only after sensor reading
    Skip_If_vTypeK
    goto    udc_exit

    ;... continue ...

    ;save old vCOILTEMP (status flags)
    movf    vCOILTEMP,W
    movwf   vCOILTEMPold

    ;clear sensor ready bit
    Clear_vTypeK

    ;set either vAbove or vBelow (or neither at transition)
    ;save maximum above error and below error
    call    CalcSPERROR

    ;sum the SPERROR
    call    CalcSPERRORsum

    ;set either vInc or vDec (or neither at transition)
    ;save maximum increasing change
    call    CalcSPERRORchg

    ;set either vAccel or vDecel (or neither at transition)
    call    CalcSPERRORacc

    ;calculate the DUTYCYCLEsp value based on DCBIAS
    call    CalcDCnew

    ;test if DUTYCYCLEnew modified
    Skip_If_vDCnew
    goto    udc_exit
```

```
        ;write DUTYCYCLEnew to eePROM ring buffer
        call eeSaveDUTYCYCLEnew

        ;store DUTYCYCLE in PWM registers
        call    PutDCnew

udc_exit:
    endm

;===== COILTEMP =====

;initialize COILTEMPptr
Initialize_COILTEMPptr macro
    movlw   cCOILTEMPstart
    movwf   COILTEMPptr
    endm

;advance COILTEMPptr and wrap at end
;implements a 16-byte (8 value) ring buffer
Advance_COILTEMPptr macro
    movf    COILTEMPptr,W
    addlw   .2
    andlw   b'00001110'     ;save bottom 4 bits
    iorlw   cCOILTEMPstart  ;set top 4 bits
    movwf   COILTEMPptr
    endm

;----------------------
; Initialize eeCOILTEMP buffer pointer
Initialize_eeCOILTEMPptr macro
    banksel eeCOILTEMPptr
    movlw   eeCOILTEMPfirst
    movwf   eeCOILTEMPptr       ;set point to first byte
    endm

;Advance eeCOILTEMP buffer pointer by 2, wrap at end
;use:   0x10 thru 0x7F
Advance_eeCOILTEMPptr macro
    banksel eeCOILTEMPptr
    incf    eeCOILTEMPptr
    incf    eeCOILTEMPptr
    btfss   eeCOILTEMPptr,.7    ;bit set = ptr >= 0x80
      goto  $+3
    ;bit set: wrap eeCOILTEMP buffer pointer
    movlw   eeCOILTEMPfirst     ;first address in buffer
    movwf   eeCOILTEMPptr
    endm

;----------------------
;Read the MAX6675 every 300 msec
;called from HeartBeat interrupt code
Read_Coil_Temperature macro
    decfsz  COILTEMPdelay
      goto  intr_done       ;not time to read MAX6675
```

```
    ;reset temperature read delay (300 msec)
    Reset_COILTEMPdelay

    ;advance COILTEMPptr to next word / wrap to first adr
    Advance_COILTEMPptr

    ;read the MAX6675
    call    ReadCOILTEMP    ;store in COILTEMP, COILTEMP+1
    ;Note:  FSR is pointing to COILTEMP (lo byte) with new reading

    ;test for missing sensor
    btfsc   INDF,vNoSensor      ;test sensor bit
      goto  Error_Sensor        ;MISSING SENSOR -- ABORT!

    ;mask and right justify COILTEMP
    call    AdjustCOILTEMP      ;FSR points to lo byte

    ;test for too hot i.e. COILTEMP > COILTEMPmax
    banksel COILTEMP
    movf    COILTEMPptr,W
    movwf   FSR         ;point to low byte
    movf    INDF,W      ;W = lo(COILTEMP)
    sublw   cCTMAXlo    ;W = lo(COILTEMPmax) - lo(COILTEMP)
    incf    FSR         ;point to hi byte
    movf    INDF,W      ;W = hi(COILTEMP)
    Skip_If_CARRY_SET   ;C clear if lo(COILTEMP) > lo(COILTEMPmax)
     incfsz INDF,W      ;C Clr, do borrow
    sublw   cCTMAXhi    ;W = hi(COILTEMPmax) - hi(COILTEMP)
    Skip_If_CARRY_SET   ;C clear if hi(COILTEMP) > hi(COILTEMPmax)

        goto    Error_TooHot    ;TOO HOT -- ABORT!

    ;save MAX6675 reading in EEPROM data array
    call    eeSaveCOILTEMP      ;12 msec

    ;test for zero value and skip it
    banksel COILTEMP
    movf    COILTEMPptr,W
    movwf   FSR
    movf    INDF,W
    incf    FSR
    iorwf   INDF,W
    Skip_If_NOT_ZERO    ;lo byte = hi byte = 0, abort

        goto    Error_ZeroRead  ;ZERO READING -- ABORT!

    ;set flag to signal reading complete
    Set_vSENSOR vTypeK
    endm

;===== SETPOINT management =====

;turn on GREEN LED if at or just crossed SETPOINT
Signal_SETPOINT macro
    call    SignalSETPOINT
    endm
```

```
;===== Error Checks =====

If_Maximum_Run_Time macro addr
    banksel RUNTIME
    decfsz  RUNTIME
      goto  $+5
    movf    RUNTIME+1
    Skip_If_NOT_ZERO
    goto    addr
    decf    RUNTIME+1
    endm

;end    *****
```