```
;-------------------
;subroutine library
;-------------------
;   eeClrCounterW                     W=counter address in eePROM
;   eeWriteWtoCounter3                W=counter address in eePROM
;   eeWriteWtoCounter4                W=counter address in eePROM
;   eeIncCounterW                     W=counter address in eePROM
;   ------------------
;   RestoreFactory
;   ------------------
;   eeGetSETPOINT
;   eeGetDCBIAS
;   eeGetByteW
;   eeClearCOILTEMPbuffer
;   eeClearDUTYCYCLEbuffer
;   (eeWriteWbytes)                   W=#bytes
;   eePutSETPOINT       12 msec
;   eePutDCBIAS
;   eePutSPERRORtrip
;   eePutvERROR
;   ------------------
;   eeSaveDUTYCYCLEnew  12 msec
;   eeSaveDCBIAS        12 msec
;   (eeSaveWinDUTYCYCLEbuffer)        W=indirect pointer
;   eeSaveBTNVALUE      12 msec       DEBUG*****
;   eeSaveCOILTEMP      12 msec
;   (eeSaveWinCOILTEMPbuffer)         W=indirect pointer
;   eePutINDIRECTatW                  W=eeADR
;   eeWriteByte
;--------------
;   DisableInterrupts
;   EnableInterrupts
;--------------
;   ReadBTN
;   SetBTNbit
;--------------
;   ReadCOILTEMP        .136 msec
;   AdjustCOILTEMP
;--------------

;**************
; clear byte counter in eePROM,
; W=counter adr in eePROM
;eeClrCounterW
;   banksel EEADR
;   movwf   EEADR           ;set EEPROM adr pointer
;   clrf    EEDAT           ;clear data byte
;   call    eeWriteByte     ;write data byte
;   return
;
; write byte in eePROM counter byte
; W=counter adr in eePROM
;eeWriteWtoCounter3
;   banksel EEADR
;   movwf   EEDAT           ;store data byte
;   movlw   ceeCounter3     ;select 1st counter
;   movwf   EEADR           ;set EEPROM adr pointer
;   call    eeWriteByte     ;write data byte
;   return
;
;eeWriteWtoCounter4
;   banksel EEADR
;   movwf   EEDAT           ;store data byte
;   movlw   ceeCounter4     ;select 1st counter
;   movwf   EEADR           ;set EEPROM adr pointer
;   call    eeWriteByte     ;write data byte
;   return
;
; increment byte counter in eePROM,
; W=counter adr in eePROM
;eeIncCounterW
;   banksel EEADR
;   movwf   EEADR           ;set EEPROM adr pointer
;   bsf     EECON1,RD       ;request EEPROM data
;   incf    EEDAT           ;increment counter
;   call    eeWriteByte     ;save byte
;   return

;--------------
RestoreFactory:
;   from            to
;   ceeFACTORYsp    ceeSETPOINT
;   ceeFACTORYbias  ceeDCBIASadr

    banksel eeFrom
    movlw   ceeFACTORYsp
      movwf eeFrom
    movlw   ceeSETPOINT
      movwf eeTo
    movlw   ceeFACTORYbytes
      movwf eeCnt

    call    DisableInterrupts
rf1:
    ;get byte from eePROM
    banksel eeFrom
    movf    eeFrom,W
    call    eeGetByteW      ;get byte in W

    ;put W(byte) into eePROM
    movwf   EEDAT
    banksel eeTo
    movf    eeTo,W
    banksel EEADR
    movwf   EEADR
    call    eeWriteByte

    ;repeat until done
    banksel eeFrom
    incf    eeFrom
    incf    eeTo
    decfsz  eeCnt
    goto    rf1

    ;signal copy done
    call    EnableInterrupts
    return

;--------------
; set value of SETPOINT from eeSETPOINT (EEPROM)
eeGetSETPOINT:
    banksel SETPOINT
    movlw   cSETPOINTadr
    movwf   FSR

    movlw   ceeSETPOINT
    call    eeGetByteW      ;get LSB
    movwf   INDF            ;save LSB

    incf    FSR

    movlw   ceeSETPOINT+1
    call    eeGetByteW      ;get MSB
    movwf   INDF            ;save MSB

    return

;--------------
; set value of DCBIAS from eeDCBIAS (EEPROM)
eeGetDCBIAS:
    banksel DCBIAS
    movlw   cDCBIASadr
    movwf   FSR

    movlw   ceeDCBIASadr
    call    eeGetByteW      ;get LSB
    movwf   INDF            ;save LSB

    incf    FSR

    movlw   ceeDCBIASadr+1
    call    eeGetByteW      ;get MSB
    movwf   INDF            ;save MSB

    return

;--------------
; set value of SPERRORtrip from eeDCBIAS (EEPROM)
eeGetSPERRORtrip:
    banksel SPERRORtrip
    movlw   cSPERRORtripadr
    movwf   FSR

    movlw   ceeSPERRORtripadr
    call    eeGetByteW      ;get LSB
    movwf   INDF            ;save LSB

    incf    FSR
    movlw   ceeSPERRORtripadr+1
    call    eeGetByteW      ;get MSB
    movwf   INDF            ;save MSB

    return

;-----------------------
;read W byte in EEPROM, return in W
eeGetByteW:
    banksel EEADR
    movwf   EEADR           ;set EEPROM adr pointer
    bsf     EECON1,RD       ;request EEPROM data
    movf    EEDAT,W         ;read byte into W
    return

;-----------------------
;clear eeCOILTEMP ring buffer (112 bytes)
eeClearCOILTEMPbuffer:
    banksel EEADR
    movlw   eeCOILTEMPfirst
    movwf   EEADR
    movlw   0xFF            ;data byte to write
    movwf   EEDAT
    movlw   eeCTBUFbytes    ;init some bytes
    goto    eeWriteWbytes

;-----------------------
;clear DUTYCYCLE ring buffer (112 bytes)
eeClearDUTYCYCLEbuffer:
    banksel EEADR
    movlw   eeDUTYCYCLEfirst
    movwf   EEADR
    movlw   0x55            ;data byte to write
    movwf   EEDAT
    movlw   eeDCBUFbytes    ;init some bytes

eeWriteWbytes:
    movwf   LOOPcnt
    call    DisableInterrupts
eeWrt1:
    call    eeWriteByte
    banksel EEADR
    incf    EEADR
    decfsz  LOOPcnt
      goto  eeWrt1
    call    EnableInterrupts
    return

;-----------------------
; save SETPOINT (2 bytes) in eePROM
; SETPOINT is 12-bit, right justified
eePutSETPOINT:
    banksel SETPOINT
    movlw   cSETPOINTadr
    movwf   FSR
    movlw   ceeSETPOINT
    goto    eePutINDIRECTatW

;-----------------------
; save DCBIAS (2 bytes) in eePROM
eePutDCBIAS:
    banksel DCBIAS
    movlw   cDCBIASadr
    movwf   FSR
    movlw   ceeDCBIASadr
    goto    eePutINDIRECTatW

;-----------------------
; save SPERRORtrip (2 bytes) in eePROM
eePutSPERRORtrip:
    banksel SPERRORtrip
    movlw   cSPERRORtripadr
    movwf   FSR
    movlw   ceeSPERRORtripadr
    goto    eePutINDIRECTatW

;-----------------------
; save SPERRORtrip (2 bytes) in eePROM
eePutvERROR:
    movlw   cvERRORadr
    movwf   FSR
    movlw   ceevERRORadr
    goto    eePutINDIRECTatW

;-----------------------
;write DUTYCYCLEnew to eePROM buffer - 12 msec
;DUTYCYCLE is 10-bit value
eeSaveDUTYCYCLEnew:
    banksel DUTYCYCLEnew
    movlw   cDUTYCYCLEnew
    goto    eeSaveWinDUTYCYCLEbuffer

;-----------------------
;write DCBIAS to eePROM buffer - 12 msec
;DCBIAS is 10-bit value
eeSaveDCBIAS:
    banksel DCBIAS
    movlw   cDCBIASadr

eeSaveWinDUTYCYCLEbuffer:
    ;W is indirect pointer
    movwf   FSR
    ;advance eeDUTYCYCLE buffer pointer by 2
    Advance_eeDUTYCYCLEptr
    ;write 2 byte value
    movf    eeDUTYCYCLEptr,W
    goto    eePutINDIRECTatW

;-----------------------
; write BTNVALUE to COILTEMP eePROM buffer
; BTNVALUE is 10-bit value from AN1
```

```
;eeSaveBTNVALUE:
;   banksel BTNVALUE
;   movlw   cBTNVALUEadr
;   goto    eeSaveWinCOILTEMPbuffer

;-----------------------
; write COILTEMP to eePROM buffer - 12 msec
; COILTEMP is 12-bits extracted from MAX6675 16-bit reading
eeSaveCOILTEMP:
    banksel COILTEMP
    movf    COILTEMPptr,W       ;get pointer to COILTEMP

eeSaveWinCOILTEMPbuffer:
    ;W is indirect pointer
    movwf   FSR                 ;setup indirect pointer
    ;advance eePROM pointer by 2
    Advance_eeCOILTEMPptr
    movf    eeCOILTEMPptr,W     ;load W with eePROM address
    ;goto   eePutINDIRECTatW

    ;drop into write eePROM function

; put 2 INDIRECT bytes into eePROM at W=eeADR
eePutINDIRECTatW:
    call    DisableInterrupts

    banksel EEADR
    movwf   EEADR           ;stuff eePROM address
    movf    INDF,W          ;fetch COILTEMP lo byte
    movwf   EEDAT           ;stuff first byte to write
    call    eeWriteByte     ;save byte 1 (lo)

    banksel EEADR
    incf    EEADR           ;advance eePROM address
    incf    FSR             ;advance pointer to hi byte
    movf    INDF,W          ;fetch COILTEMP hi byte
    movwf   EEDAT           ;stuff second byte to write
    call    eeWriteByte     ;save byte 2 (hi)

    call    EnableInterrupts
    return

;--------------
;execute WRITE eePROM sequence for one byte
;EEDAT = data byte, EEADR = eePROM address
eeWriteByte:
    banksel EECON1
    bsf     EECON1,WREN     ;enable EEPROM write
    ;required WRITE sequence
    movlw   0x55        ;unlock EEPROM write
    movwf   EECON2
    movlw   0xAA        ;signal valid write operation
    movwf   EECON2
    bsf     EECON1,WR   ;initiate write of byte
    ;wait for done
    btfsc   EECON1,WR   ;wait 6 msec (until WR bit clear)
```

```
      goto  $-1
    return

;-----------------------
;disable all interrupts
DisableInterrupts:
    banksel INTCON
    bcf     INTCON,GIE      ;clear global IE to disable INTs
      btfsc INTCON,GIE      ;test IE bit
      goto  $-1             ;wait for IE bit clear
    return

;-----------------------
;enable all interrupts
EnableInterrupts:
    banksel INTCON
    bsf     INTCON,GIE      ;re-enable INTs
      btfss INTCON,GIE      ;test IE bit
      goto  $-1             ;wait for IE bit set
    return


;===== BUTTON =====

;-----------------------
;Run ADC to read BTN input on GP1/AN1
;called from interrupt routine
;interrupts disabled
ReadBTN:

    ;configure AtoD
    ;TRISIO and ANSEL already set
    ;analog input on AN1

    ;prepare AtoD conversion
    banksel ADCON0
    movlw   cADCctrl        ;rjust, Vdd, AN1, Enable
    movwf   ADCON0

    ;start the AtoD conversion
    bsf     ADCON0,GO   ;start conversion

    ;wait for AtoD to complete (11 * 2 usec)
    btfsc   ADCON0,GO   ;is conversion done?
      goto  $-1         ;no, test again

    ;save the button 10-bit AtoD value
    banksel ADRESL
    movf    ADRESL,W    ;read lower 8 bits
    banksel BTNVALUE
    movwf   BTNVALUE    ;save lower 8 bits

    banksel ADRESH
    movf    ADRESH,W    ;read upper 2 bits
    andlw   b'00000011' ;mask 2 bits
    banksel BTNVALUE
    movwf   BTNVALUE+1  ;save upper 2 bits

;   call    eeSaveBTNVALUE  ;DEBUG ****
    return

;set bit in vSENSOR for button press
;Btn1:  2/3 * 1024  top
;Btn2:  1/2 * 1024  middle
SetBTNbit:

    ;save old vSENSOR value
    movf    vSENSOR,W
    movwf   vSENSORold

    ;initialize vSENSOR for button check
    Clear_BUTTON1           ;set if BTN1 > 620
    Clear_BUTTON2           ;set if BTN2 > 465
    Clear_BUTTON_PRESS      ;set if button press

    ;if BTNVALUE > 2/3 then BTN1
    banksel BTNVALUE
    movf    BTNVALUE,W
    sublw   cBTN1minlo
    movf    BTNVALUE+1,W
    Skip_If_CARRY_SET
     incfsz BTNVALUE+1,W
    sublw   cBTN1minhi
    Skip_If_CARRY_SET
      goto  SetBTN1

    ;else if BTNVALUE > 1/2 then BTN2
    movf    BTNVALUE,W
    sublw   cBTN2minlo
    movf    BTNVALUE+1,W
    Skip_If_CARRY_SET
     incfsz BTNVALUE+1,W
    sublw   cBTN2minhi
    Skip_If_CARRY_SET
      goto  SetBTN2

    return

SetBTN1:
;   Increment_Counter ceeCounter1   ;DEBUG ****
    ;test if same BTNpress
    banksel BTNcount
    btfss   vSENSORold,vBtn1
      clrf  BTNcount        ;reset button press counter
    incf    BTNcount        ;count duration of press
    Set_BUTTON1             ;TOP button
    Set_BUTTON_PRESS        ;signal button press
    return
SetBTN2:
;   Increment_Counter ceeCounter2   ;DEBUG ****
    ;test if same BTNpress
    banksel BTNcount
    btfss   vSENSORold,vBtn2
      clrf  BTNcount         ;reset button press counter
    incf    BTNcount         ;count duration of press
    Set_BUTTON2              ;MIDDLE button
    Set_BUTTON_PRESS         ;signal button press
    return

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

;--------------
; set COILTEMP to 2 bytes from MAX6675 (temp)
; COILTEMP      low byte
; COILTEMP+1    high byte
; time = 152 usec = 12 usec + 70 usec + 70 usec
ReadCOILTEMP:
    banksel GPIO
    movf    COILTEMPptr,W    ;get 2-byte storage adr
    movwf   FSR              ;set indirect ptr to COILTEMP
    incf    FSR              ;set ptr to COILTEMP+1 (hi byte)
    bcf     SCLK             ;bring clock low to enable data output
    bcf     CS               ;assert CHIP SELECT
    call    read_8           ;read & save first 8 bits
    decf    FSR              ;set ptr to COILTEMP (lo byte)
    call    read_8           ;read & save second 8 bits
    ;drop CS to restart MAX6675 (<220 msec to get next reading)
    bsf     CS               ;drop CHIP SELECT
    return

    ;clock 8 bits into COILTEMP via INDF
    ;time = 70 usec = 4 usec + 64 usec + 2 usec
read_8:     ;4 usec
    movlw   .8
    movwf   LOOPcnt      ;set loop counter = 8
    bcf     CARRY        ;clear C before 1st rot left
    clrf    INDF         ;clear storage byte
read_1:     ;8 usec * 8 = 64 usec
    rlf     INDF         ;clear LSB
    bsf     SCLK         ;rising edge
    bcf     SCLK         ;falling edge
    btfsc   SDATA        ;read bit, skip if 0
      incf  INDF         ;set LSB if Sdata is 1
    decfsz  LOOPcnt      ;count bit
      goto read_1        ;do 8 bits
    return

;--------------
; Adjust COILTEMP to 12 bits, right justified
AdjustCOILTEMP:
    banksel COILTEMP
    movf    COILTEMPptr,W
    movwf   FSR              ;low byte
    incf    FSR              ;hi byte
    movlw   b'01111111'
    andwf   INDF             ;mask temperature bits

    decf    FSR              ;lo byte
    movlw   b'11111000'
    andwf   INDF            ;mask temperature bits

    ;for i=1 to 3
    movlw   .3
    movwf   LOOPcnt
    Clear_CARRY             ;clear CARRY for first rotate right
adjust_loop:
    incf    FSR             ;hi byte
    rrf     INDF            ;rotate hi byte right into CARRY
    decf    FSR             ;low byte
    rrf     INDF            ;rotate CARRY into lo byte
    decfsz  LOOPcnt
      goto  adjust_loop
    return

; end   *****
```