

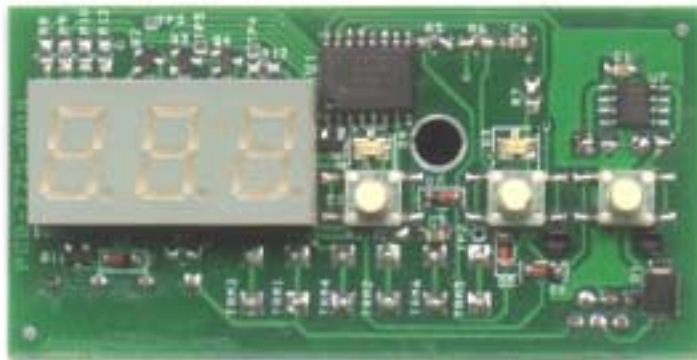
ProjectName: Refrigeration Control

ProjectNumber: F161

Abstract:

The standard mechanical thermostat used in many refrigerators designed for use in RV's normally has no reference to actual operating temperature. This results in setting the thermostat to an arbitrary value and hoping that the food in the refrigerator won't freeze and any frozen foods in the freezer section won't thaw. Usually within a few days after adjusting the thermostat setting the right operating temperature is achieved. This up/down adjustment of the thermostat can be avoided with this refrigeration control.

The three digit display is used to show the icebox temperature during normal operation. Three push buttons are used to turn the refrigerator on or off, modify the set point and change between Fahrenheit and Celsius. Pressing either the up or down buttons once forces the set point to be displayed for 15 seconds. Pressing either again modifies the set point. Pressing both at the same time toggles between Fahrenheit and Celsius. The control can hold the ice box temperature within 1°C.



Software description:

At power on reset the watchdog timer is cleared and the configuration registers are set. The IO ports are then initialized to force all leds and the compressor off. An initial a2d conversion is made to allow the temperature to be displayed.

The timer is setup to generate an interrupt every 10 milliseconds. This is used as a time base for any delays and as a signal to sample the thermistor (used to monitor the temperature of the ice box or refrigerator), scan the buttons, and update the compressor relay. While waiting for the timer interrupt to occur the temperature display and leds are refreshed.

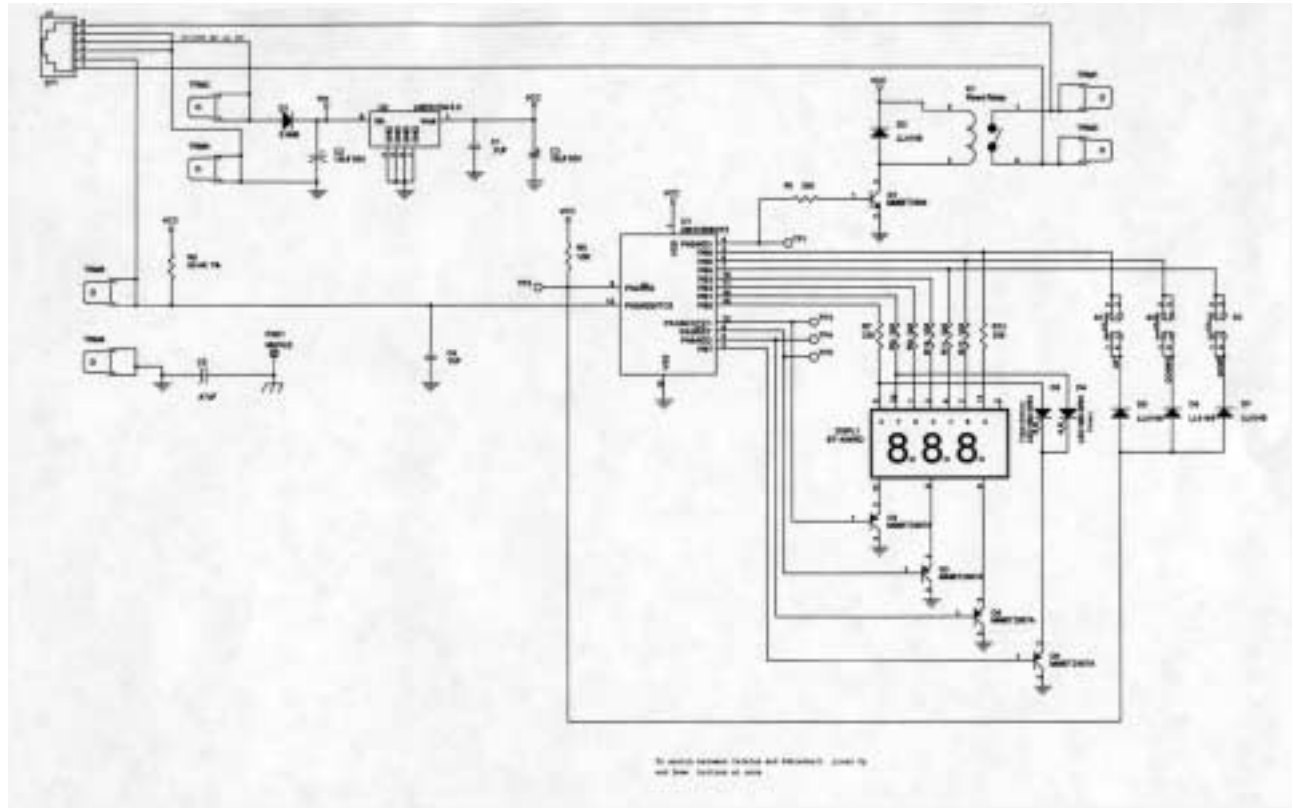
The Clock routine adjusts the holdoff timer (which prevents the compressor from restarting within 5 minutes of being turned off to reduce the chance that the compressor could stall from a high freon pressure condition during starting) and the view timer (used to override the ambient temperature display in favor of the setpoint) if activated.

DebA2D samples the thermistor once every 10 milliseconds and adds the reading to SampleSum. After 256 samples the high byte of the sum is used as the average temperature. The a2d reading is then saved as raw data and SampleSum is cleared in preparation of the next conversion samples. The a2d reading is not converted to a temperature at this point because negative numbers would be difficult to keep track of.

Once the thermistor has been taken care of the buttons are scanned and debounced. If a button or combination of buttons are pressed for 100 milliseconds the unit will decode the button and then take the necessary action. This may entail displaying the setpoint if it was not already being displayed or changing the setpoint. Pressing the up and down buttons simultaneously toggles between Fahrenheit and Celsius readings. The third button allows the system to be turned off. The brightness of the displayed can be adjusted by pressing the down and power buttons. There are three brightness levels for the display. Each time a change is made to any of the parameters it is immediately written to flash. Since a refrigerator is normally set and then taken for granted the rest of it's productive life it is not necessary to use a routine that can rotate the locations used to store this information to maximize the number of changes made to the flash.

The display routine checks if the setpoint or temperature should be displayed then determines if it should be shown in Celsius or Fahrenheit. At this point the raw a2d value is converted to a real temperature and processed for display by the main loop.

CycleCtrl gets the raw a2d reading for the temperature, converts it to Fahrenheit and compares it to the setpoint. If the temperature is above the setpoint and the compressor is off and no holdoff delay is active the compressor is turned on. If the holdoff delay is active no action is taken until the holdoff delay expires. If the compressor is running and the temperature is lower then the setpoint the compressor is turned off and the holdoff delay is initialized.



Title Refrigeration Control

```

        .PAGE0
PortA: DS      1           ;port a
thermistor: EQU    0
Hund:         EQU    1
BCom:         EQU    2
Ten:          EQU    3
Unit:         EQU    4
Relay: EQU     5
PortAInit:    EQU    (2**Hund)^(2**Ten)^(2**Unit)^(2**Relay)

PortB: DS      1           ;port b
sg:           EQU    0
sf:           EQU    1
se:           EQU    2
sd:           EQU    3
sc:           EQU    4
sb:           EQU    5
power: EQU     5
sa:           EQU    6
Comp:         EQU    6
LEDs:         EQU    7
PortBInit:    EQU    (2**LEDs)

DDRA:         EQU    $04           ;data direction register
PortADD:      EQU    (2**Hund)^(2**Ten)^(2**Unit)^(2**Relay)
DDRB:         EQU    $05           ;data direction register
PortBDD:      EQU    0FFH
PTAPUE: EQU    $B
PTBPUE: EQU    $C

KBSCR         EQU    $001a        ; Keyboard interrupt control/status
KBIER        EQU    $001b

INTSCR EQU    $001d        ; IRQ status/control
Model: EQU    0           ;edge/level control
IMASK1: EQU    1           ;interrupt mask bit 1=disabled
ACK1:         EQU    2           ;
IRQF1: EQU    3           ;

CONFIG2:      EQU    $001e        ; System configuration
CONFIG1:      EQU    $001f

TSC           EQU    $0020        ; Timer 1
TOF:          EQU    7
TOIE:         EQU    6
TSTOP: EQU     5
TRST:         EQU    4
PS2:          EQU    2
PS1:          EQU    1
PS0:          EQU    0
TCNTH         EQU    $0021
TCNTL         EQU    $0022
TMODH         EQU    $0023
TMODL         EQU    $0024
TSC0          EQU    $0025
CH0MAX: EQU    0           ;
TOV0:         EQU    1           ;

```

```

ELS0A: EQU 2 ;
ELS0B: EQU 3 ;
MS0A: EQU 4 ;
MS0B: EQU 4 ;
CH0IE: EQU 6 ;
CH0F: EQU 7 ;
TCH0H EQU $0026
TCH0L EQU $0027
TSC1 EQU $0028
CH1MAX: EQU 0 ;
TOV1: EQU 1 ;
ELS1A: EQU 2 ;
ELS1B: EQU 3 ;
MS1A: EQU 4 ;
CH1IE: EQU 6 ;
CH1F: EQU 7 ;
TCH1H EQU $0029
TCH1L EQU $002a

OSCSTAT: EQU $0036
ECGST: EQU 0
ECGON: EQU 1
OSCTRIM: EQU $0038

ADSCR EQU $003C ; A to D converter
ADCO: EQU 5
ADIE: EQU 6
COCO: EQU 7
ADR EQU $003E
Result: EQU ADR
ADICLK EQU $003F
ADIV0: EQU 5
ADIV1: EQU 6
ADIV2: EQU 7

RamStart: EQU 080H
EORAM: EQU $0FF

.CODE
SBSR EQU $fe00 ; System integration
SRSR EQU $fe01
BRKAR: EQU $fe02
BFCR EQU $fe03
INT1 EQU $fe04 ; Interrupt status
INT2 EQU $fe05
INT3 EQU $fe06
;FLTCCR EQU $fe07 ; Flash test/programming
FLCR EQU $fe08
HVEN: EQU 3
MASS: EQU 2
ERASE: EQU 1
PGM: EQU 0
BRKH EQU $fe09 ; Hardware breakpoint
BRKL EQU $fe0a
BRKSCR EQU $fe0b
LVISR EQU $fe0c ; Low voltage detect
FLBPR EQU $ff7E ; Flash block protect register

```

```
InOSCTRIM: EQU $ffc0
COPCTL EQU $ffff ; COP (Computer operating properly) control
```

```
RomStart EQU $EE00
```

```
VectorTbl EQU $FFDE
```

```
VectorPage EQU $FFC0
```

```
Stack: EQU EORAM-42
```

```
.PAGE0
```

```
;ram equates
```

```
ORG RomStart
```

```
SampleCnt: DS 1 ;number of samples to take
```

```
SampleSum: DS 2 ;sum of samples
```

```
Thermistr: DS 1 ;debounced a2d reading for thermistor
```

```
Status: DS 1
```

```
Flash: EQU 1 ;force display to flash
```

```
ViewSP: EQU 2
```

```
AdjBright: EQU 3
```

```
Dbneced: EQU 4
```

```
Tic: EQU 5
```

```
ViewTimer: DS 1
```

```
Debounce: DS 1
```

```
Tics: DS 1
```

```
Destination DS 2
```

```
LastScan: DS 1
```

```
Hundrds: DS 1 ;segment data for hundreds digit
```

```
Tens: DS 1 ;segment data for tens digit
```

```
Units: DS 1 ;segment data for units digit
```

```
;WrCnt: DS 1
```

```
;FlashDest: DS 2
```

```
Seconds: DS 1
```

```
Holdoff: DS 1
```

```
PwrBtn: EQU 4 ;value for power button
```

```
UpBtn: EQU 2
```

```
DownBtn: EQU 1
```

```
Intensity: EQU 5 ;combination of down and power buttons
```

```
Euro: EQU 3 ;combination of up and down buttons
```

```
BfrStrt: DS 4 ;reserve maximum number of byte to temporarily store  
;data to be written to flash
```

```
ExeRam: EQU $ ;executable ram
```

```
Minus: EQU 01000000B
```

```
.CODE
```

```
ORG RomStart
```

```
Power: DB 0 ;if =0 unit is turned off =1 unit is turned on
```

```
Setpoint: DB 38 ;setpoint for icebox
```

```

Fahrenheit:  DB    0FFH          ;if FF then displai temperature in Fahrenheit else celsius
Brightness:  DB     4            ;brightness control range 1-4 with 4 full brightness

```

```

*****
*                               Initialization                               *
*****

```

```

    BLKB    (($/64)+1)*64-$,09DH    ;fill page with NOP's

```

```

Start:

```

```

    CLRA
    STA    COPCTL          ;reset watchdog
    STA    CONFIG1        ;set configuration registers
    STA    CONFIG2

```

```

    MOV    #PortAInit,PortA ;initialize ports
    MOV    #PortADD,DDRA
    MOV    #PortBInit,PortB
    MOV    #PortBDD,DDRB

```

```

    LDA    InOSCTRIM      ;trim oscillator
    STA    OSCTRIM

```

```

    MOV    #.HIGH.32000,TMODH    ;with a 3.2mhz internal bus frequency
    MOV    #.LOW.32000,TMODL     ;set timer for 10msec interrupts
    BCLR   TSTOP,TSC           ;start timer
    BSET   TOIE,TSC           ;enable overflow interrupt

```

```

?a:

```

```

    LDHX   #EORAM           ;clear all ram registers
    CLR    ,X
    DECX
    CPX    #RamStart
    BHI    ?a

```

```

    MOV    #20H,ADICLK      ;initialize A2D convertor set A2D clock frequency
    MOV    #20H,ADSCR      ;turn on a2d convertor and make first conversion
    BRCLR  COCO,ADSCR,$    ;wait for conversion to finish
    LDA    ADR              ;read A2D result

```

```

    MOV    #20H,ADSCR      ;read thermistor voltage
    BRCLR  COCO,ADSCR,$    ;wait for conversion to finish
    LDA    ADR              ;save thermistor a2d value

```

```

    STA    Thermistr
    JSR    Temperature     ;convert a2d reading to temperature
    TSTA
    BMI    ?b              ;test if tempeature is negative

```

```

    JSR    HexToBCD        ;convert number to bcd for display
    STA    Units
    STX    Tens
    PSHH
    PULA

```

```

    STA    Hundrds
    BRA    ?c

```

```

?b:

```

```

    NEGA          ;make positive for conversion to bcd
    JSR    HexToBCD ;convert number to bcd for display
    STA    Units
    STX    Tens
    MOV    #Minus,Hundrds

```

```

?c:      BCLR  Relay,PortA      ;initialize cycle status= off
        MOV   #100,Tics       ;set 10 msec counter

MainLoop: LDA   Power
        BEQ   ?a
        LDA   Hundrds        ;get hundreds digit for display
        ORA   #80H           ;combine with discrete led driver
        STA   PortB          ;output segment data
        BCLR  Hund,PortA     ;turn on hundreds drive transistor
        JSR   Delay          ;wait
        BSET  Hund,PortA     ;turn off hundreds drive transistor
        ORA   #80H           ;combine with discrete led driver
        STA   PortB          ;output segment data
        BCLR  Ten,PortA      ;turn on tens drive transistor
        JSR   Delay
        BSET  Ten,PortA      ;turn off tens drive transistor
        ORA   #80H           ;combine with discrete led driver
        STA   PortB          ;output segment data
        BCLR  Unit,PortA     ;turn on units drive transistor
        JSR   Delay
        BSET  Unit,PortA     ;turn off units drive transistor
?a:      LDA   Power          ;show operating status
        AND   #2*power       ;set bit for power led
        BEQ   ?b
        BRCLR Relay,PortA,?b
        ORA   #2*Comp
?b:      STA   PortB          ;output segment data
        BCLR  LEDs,PortB     ;turn on discrete led drive transistor
        JSR   Delay
        BSET  LEDs,PortB     ;turn off discrete led drive transistor
?c:      STA   COPCTL         ;reset watchdog
        BRCLR Tic,Status,?c   ;wait for 10 msec flag to be set
        BCLR  Tic,Status     ;clear 10 msec flag
        BRA   MainLoop

Clock:   LDA   TSC           ;reset timer overflow flag
        MOV   #(2**TOIE),TSC
        DBNZ  Tics,DebA2D    ;count off 1 second
        MOV   #100,Tics      ;reset counter

        TST   Seconds        ;test if holdoff timer is active
        BEQ   ?a
        DBNZ  Seconds,?a
        TST   Holdoff
        BEQ   ?a
        MOV   #60,Seconds
        DEC   Holdoff

?a:      TST   ViewTimer      ;if not zero decrement view timer
        BEQ   DebA2D
        DEC   ViewTimer
        BNE   DebA2D         ;if timer reaches 0 clear temporary flags
        BCLR  AdjBright,Status ;clear display flags
        BCLR  ViewSP,Status

DebA2D:  MOV   #20H,ADSCR     ;read thermistor voltage

```

```

BRCLR COCO,ADSCR,$           ;wait for conversion to finish
LDA  ADR
ADD  SampleSum+1           ;add to sample sum
STA  SampleSum+1
CLRA
ADC  SampleSum
STA  SampleSum
DBNZ SampleCnt,?a           ;decrement sample counter
LDA  SampleSum           ;get average
STA  Thermistr
CLR  SampleSum           ;clear summing registers
CLR  SampleSum+1
?a: EQU  $

Switches: MOV  #0EFH,PortB       ;strobe power button
LDX  #27
DBNZX $                   ;delay 25usec
BRCLR BCom,PortA,?a       ;sample switch input to set carry
?a: ROLA                   ;rotate carry into accumulator
MOV  #0DFH,PortB         ;strobe down button
LDX  #27
DBNZX $                   ;delay 25usec
BRCLR BCom,PortA,?b       ;sample switch input to set carry
?b: ROLA
MOV  #0BFH,PortB         ;strobe up button
LDX  #27
DBNZX $                   ;delay 25usec
BRCLR BCom,PortA,?c       ;sample switch input to set carry
?c: ROLA
COMA                       ;complement switches to make pressed = 1
AND  #7                   ;mask out unused bits
LDX  LastScan             ;get last scan
STA  LastScan             ;replace LastScan with this scan
TXA
EOR  LastScan
BEQ  ?d
CLR  Debounce             ;reset debounce counter
JMP  Display
?d: LDA  LastScan           ;test if a button is pressed
BNE  ?e
BCLR Dbnced,Status        ;clear button debounced flag
JMP  Display
?e: BRSET Dbnced,Status,?f ;wait for button to be released if it was already
debounced
INC  Debounce             ;increment debounce
LDA  Debounce             ;if not debounced long enough exit
CMP  #10                  ;test if same for 100msec
BHS  Button
?f: JMP  Display

Button: CMP  #PwrBtn        ;test if only the power button is pressed now
BNE  ?up
LDA  Power                ;get power state
BEQ  ?a                   ;test current power state
CLRA                               ;is on now turn off
BRA  ?b
?a: COMA                   ;set new power state to on

```

```

?b:      STA   BfrStrt           ;save in buffer for write
          LDA   Setpoint         ;copy rest of parameters to write buffer
          STA   BfrStrt+1
          LDA   Fahrenheit
          STA   BfrStrt+2
          LDA   Brightness
          STA   BfrStrt+3
          JMP   Erase
?up:     CMP   #UpBtn           ;test if up button is pressed
          BNE   ?down
          BRCLR ViewSP,Status,?d ;allow setpoint to be modified only if it is
displayed
          LDA   Setpoint         ;get setpoint
          CMP   #55              ;test if at maximum temperature
          BHS   ?c
          INCA                    ;increase setpoint
?c:      STA   BfrStrt+1         ;save in buffer
          LDA   Power            ;copy rest of parameters to write buffer
          STA   BfrStrt
          LDA   Fahrenheit
          STA   BfrStrt+2
          LDA   Brightness
          STA   BfrStrt+3
          JMP   ?d
?down:   CMP   #DownBtn         ;test if down button is pressed
          BNE   ?bright
          BRCLR ViewSP,Status,?d ;allow setpoint to be modified only if it is
displayed
          LDA   Setpoint         ;get setpoint
          BEQ   ?e               ;test if at minimum temperature
          DECA                    ;increase setpoint
?e:      STA   BfrStrt+1         ;save in buffer
          LDA   Power            ;copy rest of parameters to write buffer
          STA   BfrStrt
          LDA   Fahrenheit
          STA   BfrStrt+2
          LDA   Brightness
          STA   BfrStrt+3
?d:      BSET  ViewSP,Status     ;force setpoint to be displayed
          MOV   #15,ViewTimer    ;set duration of setpoint displat to 5 seconds
          JMP   Erase
?bright: CMP   #Intensity        ;test if system units combination of buttons are
pressed now
          BNE   ?units
          LDA   Brightness       ;toggle system units
          INCA
          AND   #3                ;test if at maximum brightness
          BNE   ?f
          INCA                    ;set to minimum brightness
?f:      STA   BfrStrt+3
          LDA   Setpoint         ;copy rest of parameters to write buffer
          STA   BfrStrt+1
          LDA   Fahrenheit
          STA   BfrStrt+2
          LDA   Power
          STA   BfrStrt
          JMP   Erase

```

```

?units:      CMP    #Euro           ;test if system units combination of buttons are
pressed now  BNE    Display
             LDA    Fahrenheit      ;get power state
             BEQ    ?g              ;test current power state
             CLRA   ;is on now turn off
             BRA    ?h
?g:          COMA   ;set new power state to on
?h:          STA    BfrStrt+2
             LDA    Setpoint        ;copy rest of parameters to write buffer
             STA    BfrStrt+1
             LDA    Brightness
             STA    BfrStrt+3
             LDA    Power
             STA    BfrStrt

Erase:       LDHX   #EOE-PgErase     ;set loop count for coping erase subroutine to
ram          LDA    PgErase-1,X      ;copy erase subroutine to ram
?a:          STA    ExeRam-1,X
             DBNZX ?a
             JSR    ExeRam

Write:       LDHX   #EORW-RowWrite   ;set loop count for coping rowwrite subroutine
to ram      LDA    RowWrite-1,X      ;copy rowwrite subroutine to ram
?a:          STA    ExeRam-1,X
             DBNZX ?a
             LDA    #4              ;set number of bytes to be written to flash
             JSR    ExeRam

Display:     BRCLR  ViewSP,Status,?a ;test if need to show setpoint
             CLR    Hundrds
             LDA    Setpoint
             LDX    Fahrenheit      ;test if need to show celsius
             BNE    ?b
             SUB    #32             ;convert to celsius
             BPL    ?c
             MOV    #Minus,Hundrds
             NEGA   ;make number positive
?c:          LDX    #5
             MUL
             CLRH
             TXA
             LDX    #9
             DIV
?b:          JSR    HexToBCD
             STA    Units
             STX    Tens
             BRA    CycleCtrl
?a:          LDA    Thermistr
             JSR    Temperature     ;convert a2d reading to temperature
             BMI    ?d
             JSR    HexToBCD       ;convert number to bcd for display
             STA    Units
             STX    Tens
             PSHH

```

```

        PULA
        STA   Hndrds
        BRA   CycleCtrl
?d:    NEGA                ;make positive for conversion to bcd
        JSR   HexToBCD     ;convert number to bcd for display
        STA   Units
        STX   Tens
        MOV   #Minus,Hndrds
CycleCtrl: LDA   Thermistr
        JSR   Temperature  ;convert a2d reading to temperature
        BMI   ?a
        CMP   Setpoint
        BHI   ?c
?a:    BRCLR  Relay,PortA,?b
        MOV   #5,Holdoff
        MOV   #1,Seconds
?b:    BCLR   Relay,PortA
        RTI
?c:    TST   Holdoff
        BNE   dummy
        TST   Seconds
        BNE   dummy
        BSET  Relay,PortA
dummy: RTI

;*****
;      subroutines
;*****
Delay:  LDX   Brightness   ;get brightness level
        LDA   #128         ;setup for a 1msec delay
        PSHA
        LDA   #3
?a:    DBNZ  1,SP,?a
        DBNZ  ?a
        DBNZ  ?a           ;total delay is X*1msec
        RTS

Temperature: CMP   #224     ;test if temp is within Fahrenheit table range
        BHI   ?c
        SUB   #65
        BCS   ?d
        CLRH
        LDX   Fahrenheit   ;test if display is in Fahrenheit mode
        BEQ   ?a
        TAX
        LDA   FTbl,X       ;move thermistor value to X
        RTS                ;get Fahrenheit reading
?a:    TAX
        LDA   Celsius,X    ;get celsius value
        RTS
?c:    BSET  Flash,Status
        LDA   #-40
        RTS
?d:    BSET  Flash,Status
        LDA   #60
        RTS

```

```

;load number in acc when entering.
;H,X are changed
;BCD digits will be placed in H:X:A(hundreds:tens:units) on exit from subroutine
HexToBCD:   AIS    #-5           ;add variables to stack
            STA    4,SP         ;save value to be converted
            LDA    #8           ;set outer loop count
            STA    5,SP
            CLR    1,SP         ;clear destination regs
            CLR    2,SP         ;clear destination regs
            CLR    3,SP         ;clear destination regs
            LDA    4,SP         ;test if value is 0
            BEQ    ?h
?a         LSL    4,SP         ;shift bit into carry
            LDA    3,SP         ;double orginal value adding the shifted bit or carry
            ADC    3,SP
            CMP    #10          ;test if digit has exceeded 9
            BLO    ?b
            ADD    #6           ;add 6 to correct back to decimal
            AND    #0FH         ;remove upper nibble
            SEC           ;set carry so that overflow to next digit will occur
            BRA    ?c
?b         CLC           ;clear carry to enshure no overflow
?c         STA    3,SP         ;save updated digit
            LDA    2,SP         ;double orginal value adding the shifted bit or carry
            ADC    2,SP
            CMP    #10          ;test if digit has exceeded 9
            BLO    ?d
            ADD    #6           ;add 6 to correct back to decimal
            AND    #0FH         ;remove upper nibble
            SEC           ;set carry so that overflow to next digit will occur
            BRA    ?e
?d         CLC           ;clear carry to enshure no overflow
?e         STA    2,SP         ;save updated digit
            LDA    1,SP         ;double orginal value adding the shifted bit or carry
            ADC    1,SP
            CMP    #10          ;test if digit has exceeded 9
            BLO    ?f
            ADD    #6           ;add 6 to correct back to decimal
            AND    #0FH         ;remove upper nibble
            SEC           ;set carry so that overflow to next digit will occur
            BRA    ?g
?f         CLC           ;clear carry to enshure no overflow
?g         STA    1,SP         ;save updated digit
            DEC    5,SP         ;dec bit counter
            BNE    ?a
?h:        LDA    1,SP         ;get hundreds digit
            PSHA          ;move hundreds to h
            PULH
            LDX    2,SP         ;get tens digit
            LDA    3,SP         ;test if tens digit can be blanked
            AIS    #5
            RTS

PgErase:   LDA    #(2**ERASE)   ;set erase bit
            STA    FLCR
            LDA    FLBPR        ;read flash block protect register
            LDHX   Destination ;get Destination address

```

```

        STA    ,X
        LDA    #11                ;wait 10usec
        DBNZA $
        LDA    #(2**ERASE)^(2**HVEN) ;set erase and hven bits
        STA    FLCR
        LDHX   #2560              ;wait 4msec
?a:     AIX    #-1
        BNE    ?a
        LDA    #(2**HVEN)        ;set erase and hven bits
        STA    FLCR
        LDA    #6                 ;wait 5usec
        DBNZA $
        STA    FLCR
        LDA    #1                 ;wait 1usec
        DBNZA $
        RTS
EOE:    EQU    $

RowWrite: PSHA                   ;save number of bytes to be written
        LDA    #(2**PGM)         ;set program bit
        STA    FLCR
        LDA    FLBPR              ;read flash block protect register
        LDHX   Destination       ;get Destination address
        STA    ,X
        LDA    #11                ;wait 10usec
        DBNZA $
        LDA    #(2**PGM)^(2**HVEN) ;set program and hven bits
        STA    FLCR
        NOP                       ;used to make delay 5usec
?a:     LDHX   Destination       ;get location to be written
        LDA    #32                ;determine which location to write to
        SUB    1,SP
        TAX
        CLRH
        LDA    BfrStrt,X         ;get data to be written
        LDHX   Destination       ;get address to be written to
        STA    ,X
        LDA    #33                ;wait 30usec
        DBNZA $
        INC    Destination       ;adjust address pointer
        DBNZ   1,SP,?a          ;decrement loop counter
        LDA    #(2**HVEN)        ;set and hven bits
        STA    FLCR
        LDA    #6                 ;wait 5usec
        DBNZA $
        STA    FLCR
        LDA    #3                 ;wait 2usec
        DBNZA $
        RTS
EORW:   EQU    $

;     xg fedcba
Segment: DB    00111111B        ;0
        DB    00000110B        ;1
        DB    01011011B        ;2
        DB    01001111B        ;3
        DB    01100110B        ;4

```

```

DB 01101101B ;5
DB 01111101B ;6
DB 00000111B ;7
DB 01111111B ;8
DB 01100111B ;9
DB 01110111B ;A
DB 01111100B ;b
DB 00111001B ;C
DB 01011110B ;d
DB 01111001B ;E
DB 01110001B ;F

```

```

FTbl: DB 58,57,56,55,55,54,53,52,52,51 ;a2d 065-74
DB 50,50,49,48,48,47,46,46,45,44 ;075-84
DB 44,43,42,42,41,40,44,39,39,38 ;085-94
DB 37,37,36,36,35,34,34,33,33,32 ;095-104
DB 31,31,30,30,29,29,28,28,27,26 ;105-114
DB 26,25,25,24,24,23,23,22,22,21 ;115-124
DB 21,20,19,19,18,18,17,17,16,16 ;125-134
DB 15,15,14,14,13,13,12,12,11,11 ;135-144
DB 10,10,9,9,8,8,7,6,6,5 ;145-154
DB 5,4,4,3,3,2,2,1,1,0 ;155-164
DB 0,0,-1,-1,-2,-2,-3,-3,-4,-4 ;165-174
DB -5,-5,-6,-7,-7,-8,-8,-9,-9,-10 ;175-184
DB -11,-11,-12,-12,-13,-14,-14,-15 ;185-192
DB -15,-16,-17,-17,-18,-19,-19,-20 ;193-200
DB -21,-21,-22,-23,-23,-24,-25,-26 ;201-208
DB -26,-27,-28,-29,-29,-30,-31,-32 ;209-216
DB -33,-33,-34,-35,-36,-37,-38,-39 ;217-224

```

```

Celsius: DB 14,13,13,12,12,12,11,11,11,10
DB 10,10,9,8,8,8,7,7,7,6
DB 6,6,5,5,5,4,4,3,3,3
DB 2,2,2,2,1,1,1,0,0,0
DB -1,-1,-1,-1,-1,-1,-2,-2,-2,-3
DB -4,-4,-4,-4,-4,-5,-5,-5,-5,-6
DB -6,-6,-7,-7,-7,-7,-8,-8,-8,-8
DB -9,-9,-10,-10,-10,-10,-11,-11,-11,-11
DB -12,-12,-12,-12,-13,-13,-13,-14,-14,-15
DB -15,-15,-15,-16,-16,-16,-16,-17,-17,-17
DB -17,-17,-18,-18,-18,-18,-19,-19,-20,-20
DB -20,-20,-21,-21,-21,-22,-22,-22,-22,-23
DB -23,-23,-24,-24,-25,-25,-25,-26
DB -26,-26,-27,-27,-27,-28,-28,-28
DB -29,-29,-30,-30,-30,-31,-31,-32
DB -32,-32,-33,-33,-33,-34,-35,-35
DB -35,-36,-36,-37,-37,-38,-38,-39

```

```

iftrue $<0FDFH
BLKB 0FE00H-$,09DH
ELSE
MESSAGE ERROR - Program Size Exceeds Available ROM Space
ENDIF

```

```
org VectorTbl
dw dummy      ; ADC Conversion Complete
dw dummy      ; Keyboard Vector
dw dummy      ;
dw dummy      ;
dw dummy      ;
dw dummy      ;
dw dummy      ;
dw dummy      ;
dw dummy      ;
dw dummy      ;
dw Clock      ; TIM1 Overflow Vector
dw dummy      ; TIM1 Channel 1 Vector
dw dummy      ; TIM1 Channel 0 Vector
dw dummy      ;
dw dummy      ; ~IRQ1 Vector
dw dummy      ; SWI Vector
dw Start      ; Reset Vector
```

```
ORG    0FF7EH
DB     .LOW.((Start)-0C000H)/64      ;set protection for program
```

END