Triple Stage Incubator

Author: Brian Iehl
Hoffman Estates IL
brian@dls.net

OVERVIEW

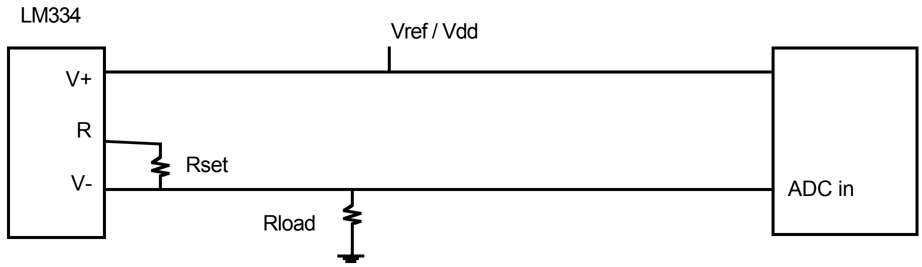
This project is a triple stage incubator. Three separate incubators are simultaneously controlled by one microcontroller. Each incubator stage has its own programmable active temperature range, which is needed if there are different temperature requirements for each stage. This circuit senses the temperature of the environment in each incubator and then controls the corresponding heat lamp in each incubator to keep the temperature within the active temperature range.

Each incubator has a temperature sensor which is connected to the microcontroller. The microcontroller scans the sensors and determines the temperature of each stage. If the temperature for any stage falls below its lower threshold, the microcontroller turns on a heat lamp in the corresponding incubator. If the temperature of any stage rises above its upper threshold, the microcontroller turns the heat lamp in the corresponding incubator off.

By using the PIC12C671 microcontroller, with its built in analog to digital converter (ADC), this design can control all three stages with one microcontroller at a lower cost that designs that either require an external ADC or use digital interface temperature sensors.

APPLICATION OPERATION

Since the temperature sensors will not be located close to the microcontroller, the LM334 current mode sensor is used instead of a simpler voltage mode sensor. A current mode sensor has more noise immunity and is useful if the temperature sensor is located physically far away from the ADC. This temperature sensor passes a small current that varies with temperature. With the component values used here, the current is one microamp (uA) per degree Kelvin (K). Kelvin is an absolute scale; 0 corresponds to absolute zero. To convert K to C, just subtract 273. In order to measure this tiny current we pass it through the 5K resistor at the ADC input, which converts it to a voltage. The spreadsheet shown below calculates the resolution. That is, how many degrees does one ADC bit correspond to. For this design we see that the resolution is 1.3 K per bit.

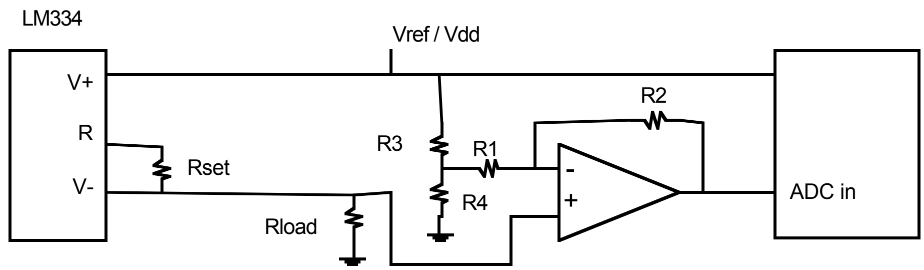


Current mode Temperature Sensor Design Equations		
These design equations are for the LM134 current mode temperature sensor. Input the ADC Vref used, the ADC bits, the desired Max temperature, and the Rload to be used. These equations will tell you what Rset value to use and what the degrees of resolution will be.This design is constrained by the fact that is measure all the way down to absolute zero and hence reduces the resolution that could be obtained if the temperature range was limited on the low end. Current mode temperature sensors are preferred over voltage mode sensors when the sensor is located physically far away from the ADC input because of better noise immunity.		
Vref, VDD	5.0	V
ADC Bits	8	
Desired Max Temperature (F)	135.0	F
Max Temperature (C) (F-32)*5/9	57.2	C
Max Temperature (K) (C+273.15)	330.4	K
Number of ADC Steps	256	
ADC Step Size = (Vref / Num Steps)	19.5	mV
Required Temp Coeff = (Vref / Max Temp)	15.1	mV / K
Resolution = (ADC Step Size / Temp Coeff)	1.29	K / Bit
Rload	5000	ohms
Iset = (Temp Coeff / Rload)	3.03	uA / K
Rset = (227 uV / K) / Iset	75	ohms
Example Temperature (F)	87	F
Example Temperature (C)	30.6	C
Example Temperature (K)	303.7	K
Example ADC Reading (K / Resolution)	235	Bits

Using this we can convert the desire temperature range into corresponding bit levels. This is then programmed in as high and low levels for each stage.

This design saves a considerable amount of money over those designs that use digital interface temperature sensors and the less expensive PIC12C508. Even though the PIC12C671 is more expensive than the PIC12C508, the difference is much less than the cost of the digital sensors IC's. In addition a part such as the widely used Dallas DS1620 Digital Thermometer needs 3 lines to communicate with it. Two of these could be shared, but that would still only allow a two stage incubator to be built instead of a three stage, with out additional I/O ports. Typical prices (JDR Microdevices single qty) are $6.75 for each DS1620 and $0.89 for each LM334. Even for a dual incubator design that is a cost of $1.78 compared to $13.50, a savings of 11.72 in the sensors alone. This more than offset the slightly higher cost of the PIC12C671.

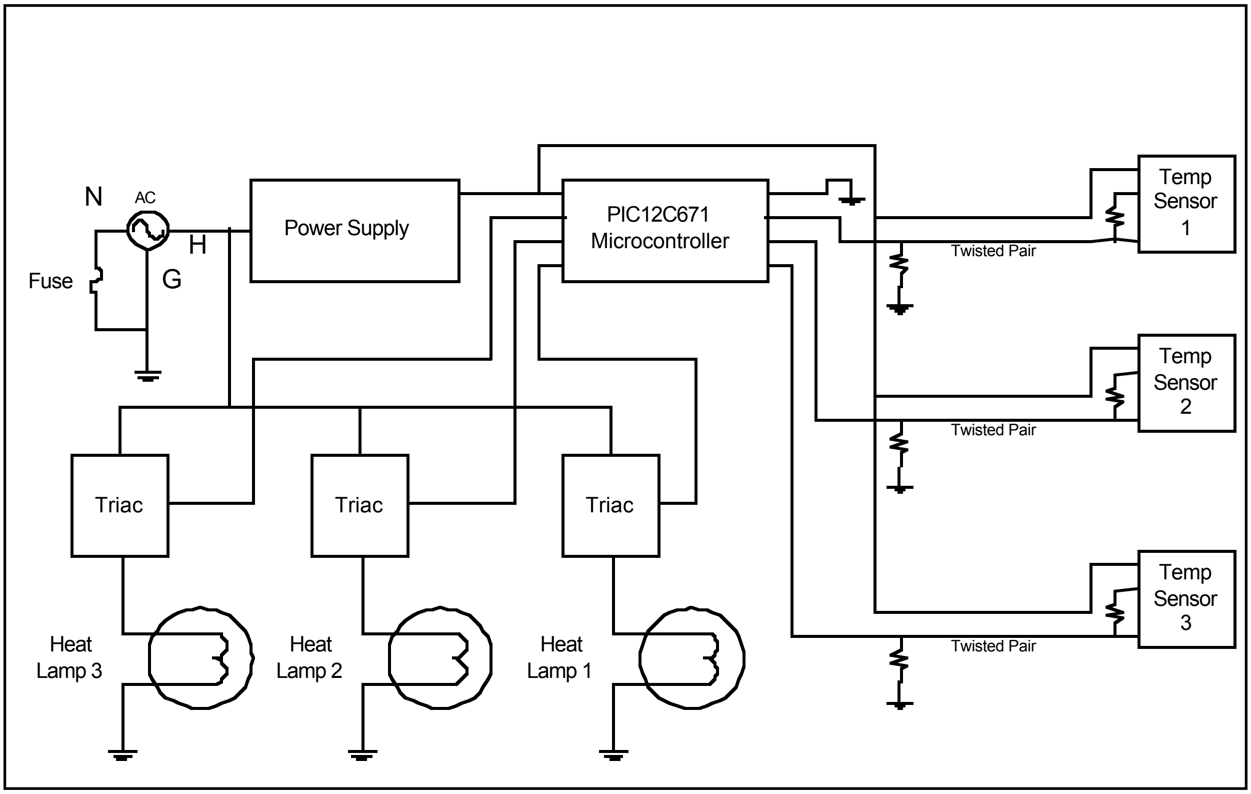
If smaller temperature resolution was require, an op amp could be added in the temperature sensor circuitry that would allow the range to be adjusted to a minimum and maximum temperatures. This keeps the circuit from responding down to absolute zero which uses valuable ADC bits for temperatures that are not used. The op amp adds an offset corresponding to the minimum desired temperature and adds gain to spread the desired temperature range across the full ADC bit range. The spread sheet shown below show the calculations needed to determine the resistor values to use in the circuit and the resolution that is obtained. The spread sheet calculates the optimal circuit that will give 0 ADC reading at the specified minimum temperature and a 255 ADC reading at the specified maximum temperature reading.



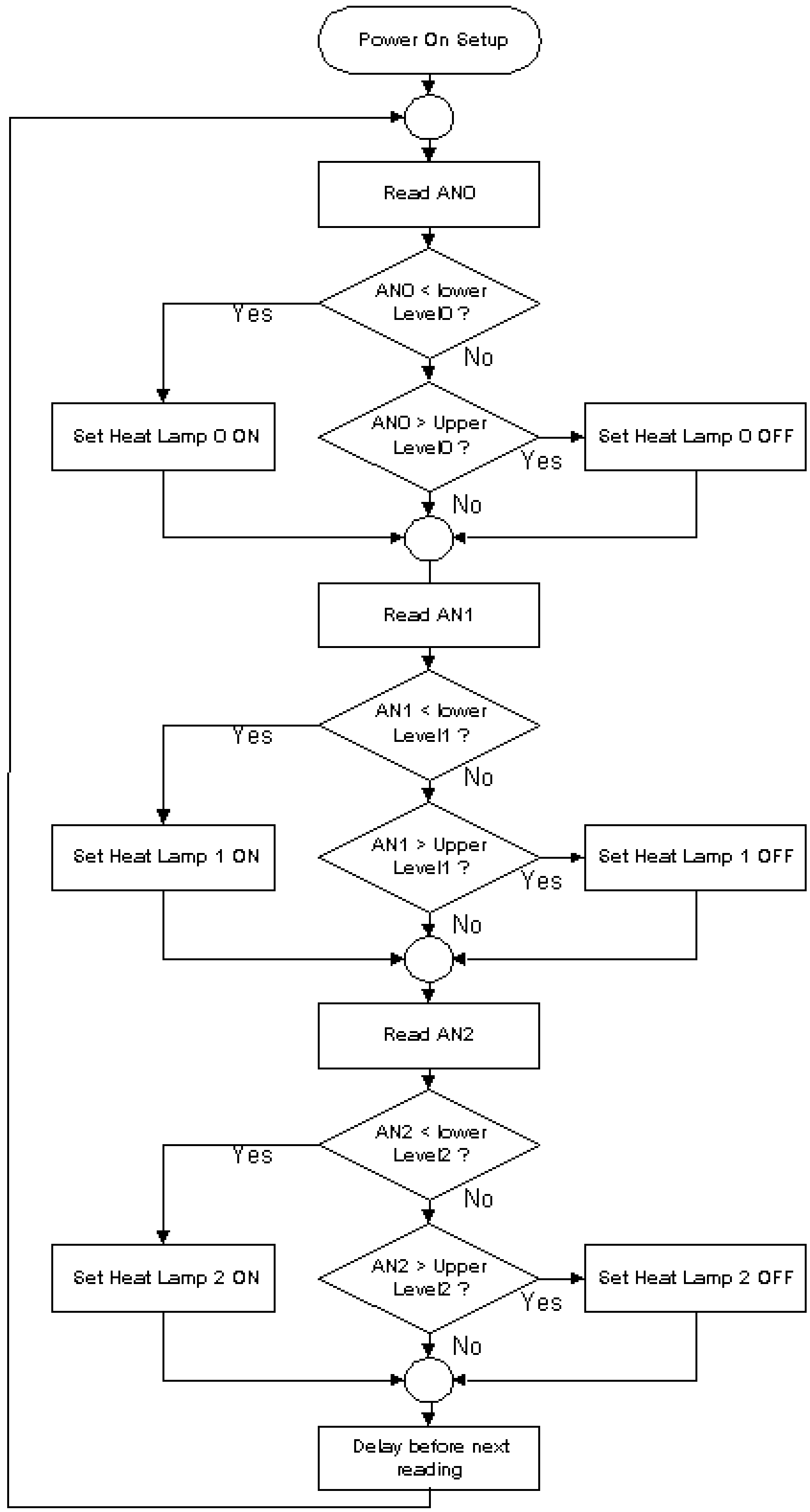
Current mode Temperature Sensor Design Equations		
These design equations are for the LM134 current mode temperature sensor. Input the ADC Vref used, the ADC bits, the desired Max and Min temperatures, the Rload and R1 to be used. These equations will tell you what values to use for R2, R3, R4, and Rset. The op amp in this design is used to provide an offset which will limit the lower range, thus not wasting AD bits on temperatures that will never be used. It also provides gain so that the max desired temperature occures at the max ADC level. This optimizes resolution. Current mode temperature sensors are preferred over voltage mode sensors when the sensor is located physically far away from the ADC input because of better noise immunity.		
Vout = Vin * Gain - Offset		
Vref , Vdd	5.0	V
ADC Bits	8	
Desired Max Temperature (F)	120.0	F
Desired Min Temperature (F)	0.0	F
Offset = (Vref / 2)	2.5	V
Max Temperature (C) (F-32)*5/9	48.9	C
Max Temperature (K) (C+273.15)	322.0	K
Min Temperature (C) (F-32)*5/9	-17.8	C
Min Temperature (K) (C+273.15)	255.4	K
Temperature Range (Max T - Min T)	66.7	K
Number of ADC Steps (2 ^ ADC Bits)	256	
Resolution = (Temp Range /Num ADC Steps)	0.26	K / Bit
ADC Step Size = (Vref / Num Steps)	19.5	mV
Effective Temp Coeff = (Vref / Temp Range)	75.0	mV / K
Gain = 2 / ((Max Temp / Min Temp) - 1)	7.7	
Temp Coeff = (Effective Temp Coeff / Gain)	9.8	mV / K
Rload	10000	ohms
Iset = (Temp Coeff / Rload)	0.98	uA / K
Rset = (227 uV / K) / Iset	232	ohms
R1	30000	ohms
R2 = R1 * Gain -1	229834	ohms
Voffset = (R1/R2)*Offset	0.33	V
R4 = R1 / 10	3000	ohms
R3 = R4 * ((Vdd / Voffset) - 1)	42966.8	ohms
Example Temperature (F)	104	F
Example Temperature (C)	40.0	C
Example Temperature (K)	313.2	K
Example ADC Reading ((K - Min Temp) / Res)	222	Bits

Since the PIC12C671 draws so little current, a very simple, low cost power supply circuit is used which does not need a transformer. This power supply design was taken from the Microchip application note TB008.

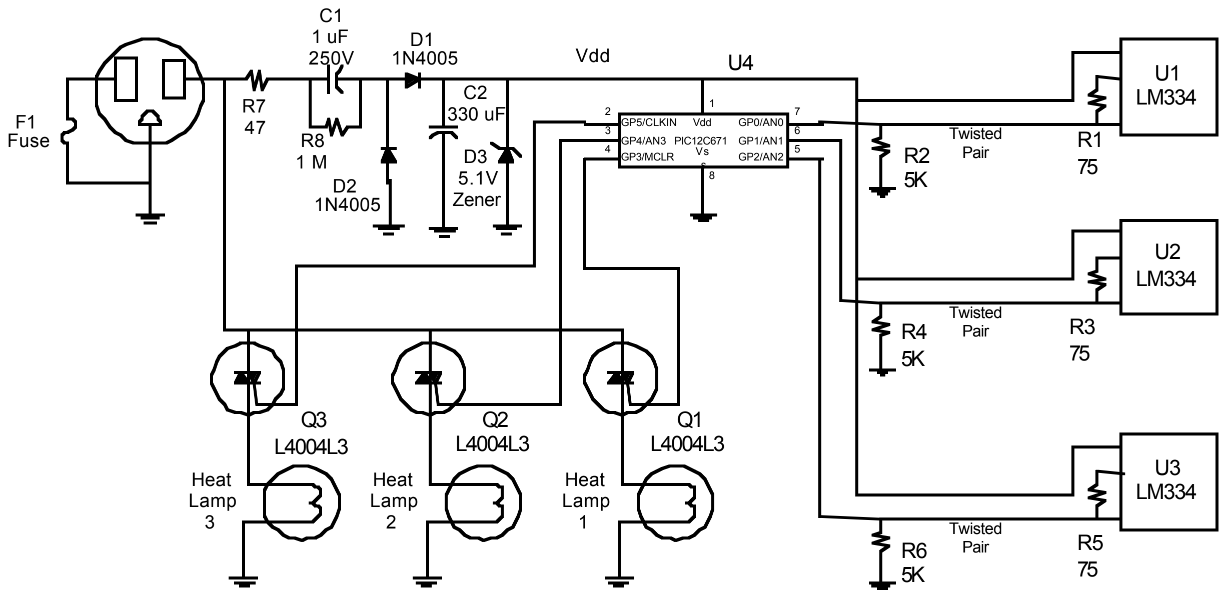
BLOCK DIAGRAM



FLOWCHART


Power On Setup
Read AN0
AN0 < lower Level0 ?
Yes
No
Set Heat Lamp 0 ON
AN0 > Upper Level0 ?
Yes
No
Set Heat Lamp 0 OFF
Read AN1
AN1 < lower Level1 ?
Yes
No
Set Heat Lamp 1 ON
AN1 > Upper Level1 ?
Yes
No
Set Heat Lamp 1 OFF
Read AN2
AN2 < lower Level2 ?
Yes
No
Set Heat Lamp 2 ON
AN2 > Upper Level2 ?
Yes
No
Set Heat Lamp 2 OFF
Delay before next reading

GRAPHICAL HARDWARE REPRESENTATION



BILL OF MATERIALS (BOM)

Ref	Part#	Manufacture
U1,2,3	LM334Z	National Semiconductor
U4	PIC12C671	Microchip Technology
Q1,2,3	L4004L3	Teccor
D3	1N5231BCT	Liteon

MICROCHIP TOOLS USED

Assembler/Compiler version:

MPLAB 3.22.02

APPENDIX A: SOURCE CODE

```
               Title                      "Triple Incubator Controler"
               Subtitle  "Version 1.0"

;              Written by Brian Iehl 7/22/97
;              Last modified 7/27/97

        list p=12C671

               INCLUDE c:\apps\mplab\p12c671.inc

SetIO                   equu              B'00111000'  ; 0 for output, 1 for input
SetADC                  equ               B'00000010'  ; ADC for GP2, GP1, GP0

GPIO0                   equ               0
GPIO1                   equ               1
GPIO2                   equ               2
GPIO3                   equ               3
GPIO4                   equ               4
GPIO5                   equ               5
                                                       ; Program Temp limits here
                                                       ; Temp (Kelvin) / Resolution (K / Bit)
LowRd0                  equ               D'241'       ; 101 F, Port 0 low temp bit reading
HighRd0                 equ               D'243'       ; 104 F, Port 0 High temp bit reading
LowRd1                  equ               D'239'       ;  96 F, Port 1 low temp bit reading
HighRd1                 equ               D'241'       ; 101 F, Port 1 High temp bit reading
LowRd2                  equ               D'235'       ;  87 F, Port 2 low temp bit reading
HighRd2                 equ               D'239'       ;  96 F, Port 2 High temp bit reading

TriacCntl0     equ      GPIO3                          ; Output Triac Control 0
TriacCntl1     equ      GPIO4                          ; Output Triac Control 1
TriacCntl2     equ      GPIO5                          ; Output Triac Control 2
TriacOn                 equ               1            ; Hi to turn Triac on
TriacOff       equ      0                              ; Lo to turn Triac off


ReadDelay      equ      D'15'                          ; S to wait for next reading

ScratchPadRam  equ      0x20
DelayValue     equ      ScratchPadRam+0  ; For mSDelay Macro
SDelayValue    equ      ScratchPadRam+1  ; For SDelay Macro
ADCRead                 equ               ScratchPadRam+2; Save ADC Reading

;********************  Macros *************************************

MOVLF          MACRO    LL,               FF           ; Move Literal to register file
               MOVLW    LL                             ; Load literal
               MOVWF    FF                             ; Store in register file
               ENDM                                    ; end MOVLF

                                                       ; Assumes 4 MHz clock
mSDelay        MACRO    mS                             ; Number of mS to delay up to 255 mS
                                                       ; each clock cycle is 1 uS = .001 mS
               LOCAL    Loop, SetTmr
               MOVLF    mS,               DelayValue   ; store number of mS delay
               CLRWDT                                  ; avoid unitentional reset

SetTmr         BSF      STATUS,           RP0          ; Switch to bank 1 for OPTION
               MOVLF    B'00000111',OPTION_REG         ; Set prescaler to 256,
                                                       :   clear PSA, Clear T0CS
               BCF      STATUS,           RP0          ; Switch to bank 0

               MOVLF    -4,               TMR0         ; 4 * 256 = 1024 uS ~ 1 mS
Loop           MOVF     TMR0,             w            ; force check zero
```

```
                BTFSS    STATUS,          Z             ; w = 0 if same, so Z is set
                goto     Loop                           ; not 0 so loop again
                                                        ; one more mS passed
                DECFSZ   DelayValue,      f             ; count down number of mS
                goto     SetTmr                         ; not done reset timer
                                                        ; if DelayValue = 0 then done
                ENDM                                    ; end mSDelay


SDelay          MACRO    S                              ; Number of Seconds delay up to 63
                LOCAL Loop

                BCF      STATUS,          RP0           ; Switch to bank 0
                MOVLF    S*4,             SDelayValue   ; store number of S delay
Loop            mSDelay  D'250'                         ; Delay 0.25 sec
                DECFSZ   SDelayValue,     f             ; count down number of S
                goto     Loop                           ; not done reset timer
                                                        ; if DelayValue = 0 then done

                ENDM                                    ; end SDelay


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

        org     0x0A       ;start address 0x0A
        goto    Start
        org     0x10

;
Start


Setup           BSF      STATUS,          RP0           ; Switch to bank 1 for TRIS
                MOVLF    SetIO,           TRISIO        ; Load IO configuration byte
                MOVLF    SetADC,          ADCON1        ; Setup ports to be ADC inputs
                BCF      STATUS,          RP0           ; Switch to bank 0

                BCF      ADCON0,          ADCS1         ; Set clock source 8 Tosc
                BSF      ADCON0, ADCS0                  ; 01 = 2.0 uS at 4 MHz clock
                BSF      ADCON0, ADON                   ; Turn ADC module on

                CLRF     DelayValue                     ; Init variables
                CLRF     SDelayValue

MainLoop
                                                        ; Read ADC 0 **********************
                mSDelay  1                              ; Wait before next conversion
                BCF      ADCON0,          CHS1          ; AN0 , CHS1 = 0
                BCF      ADCON0, CHS0                   ; AN0, CHS0 = 0

                mSDelay  1                              ; Wait for acquisition time
                BSF      ADCON0, GO_DONE                ; Start Conversion

ADCLp0          BTFSC    ADCON0, GO_DONE                ; Check to see if conversion done
                GOTO     ADCLp0                         ; Check again

                MOVF     ADRES,           w             ; Read Result
                MOVWF    ADCRead                        ; Store result in ADCRead

                                                        ; Compare to Lower limit
                SUBLW    LowRd0                         ; if ADC <= limit then result pos, C=1
                BTFSC    STATUS,          C             ; if ADC <= limit turn on lamp
                BSF      GPIO,            TriacCntl0    ; Turn Triac on

                MOVF     ADCRead, w                     ; recall ADC Reading
                                                        ; Compare reading to high limit
```



```
                SUBLW     HighRd0                          ; if ADC > limit then result neg, C=0
                BTFSS     STATUS, C                        ; if ADC > limit then turn off lamp
                BCF       GPIO,            TriacCntl0      ; Turn Triac Off


                                                           ; Read ADC 1 **********************
                mSDelay   1                                ; Wait before next conversion
                BCF       ADCON0,          CHS1            ; AN1 , CHS1 = 0
                BSF       ADCON0, CHS0                     ; AN1, CHS0 = 1

                mSDelay   1                                ; Wait for acquisition time
                BSF       ADCON0, GO_DONE                  ; Start Conversion

ADCLp1          BTFSC     ADCON0, GO_DONE                  ; Check to see if conversion done
                GOTO      ADCLp1                           ; Check again

                MOVF      ADRES,           w               ; Read Result
                MOVWF     ADCRead                          ; Store result in ADCRead

                                                           ; Compare to Lower limit
                SUBLW     LowRd1                           ; if ADC <= limit then result pos, C=1
                BTFSC     STATUS,          C               ; if ADC <= limit turn on lamp
                BSF       GPIO,            TriacCntl1      ; Turn Triac on

                MOVF      ADCRead, w                       ; recall ADC Reading
                                                           ; Compare reading to high limit
                SUBLW     HighRd1                          ; if ADC > limit then result neg, C=0
                BTFSS     STATUS, C                        ; if ADC > limit then turn off lamp
                BCF       GPIO,            TriacCntl1      ; Turn Triac Off

                                                           ; Read ADC 2 **********************
                mSDelay   1                                ; Wait before next conversion
                BSF       ADCON0,          CHS1            ; AN2 , CHS1 = 1
                BCF       ADCON0, CHS0                     ; AN2, CHS0 = 0

                mSDelay   1                                ; Wait for acquisition time
                BSF       ADCON0, GO_DONE                  ; Start Conversion

ADCLp2          BTFSC     ADCON0, GO_DONE                  ; Check to see if conversion done
                GOTO      ADCLp2                           ; Check again

                MOVF      ADRES,           w               ; Read Result
                MOVWF     ADCRead                          ; Store result in ADCRead

                                                           ; Compare to Lower limit
                SUBLW     LowRd2                           ; if ADC <= limit then result pos, C=1
                BTFSC     STATUS,          C               ; if ADC <= limit turn on lamp
                BSF       GPIO,            TriacCntl2      ; Turn Triac on

                MOVF      ADCRead, w                       ; recall ADC Reading
                                                           ; Compare reading to high limit
                SUBLW     HighRd2                          ; if ADC > limit then result neg, C=0
                BTFSS     STATUS, C                        ; if ADC > limit then turn off lamp
                BCF       GPIO,            TriacCntl2      ; Turn Triac Off


                SDelay    ReadDelay                        ; Wait before next reading

                goto      MainLoop                         ; Return to top of main loop
                END
```