

Digital Thermometer Project

This lab will illustrate use of PICKIT 2 and MPLAB in the process of assembly, execution and debug of an LCD 1x16 Character module with a thermal sensor to build a digital thermometer that display temperature.

Part 1-Hardware Configuration

A. Integrate a LCD with PICKIT2 module as shown—this should have been retained from an earlier LAB.

Make sure to use potentiometer provided as contrast control

Before power up show construction to instructor. Remove jumper JP1 from PICKIT.

Power up and adjust contrast for readability

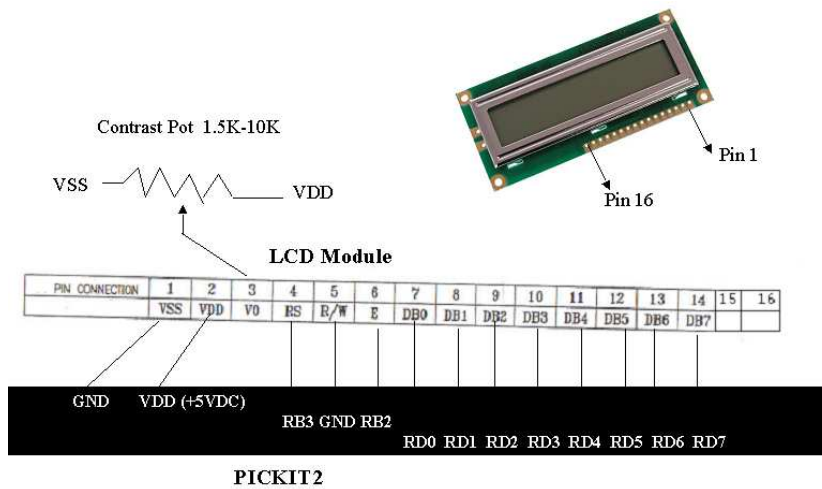


Figure 1 Hook-up Instructions

B. Retain the LCD configuration and add a LM34 Precision Fahrenheit Temperature Sensor to AN1 input of the PICKIT as shown

Connect to VDD, GND and AN1 as shown.

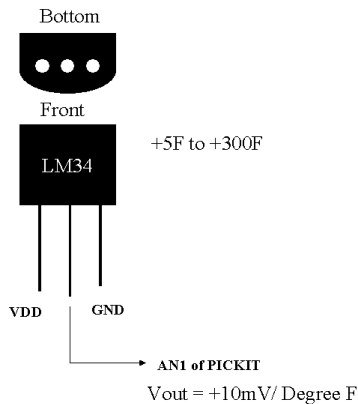


Figure 2 LM34 Thermal sensor

Part 2 Digital Thermometer Requirements

Display temperature of +5 Degree F to +300 Degree Continuously on to an LCD display. An Accuracy of one degree F is acceptable.

Block Diagram

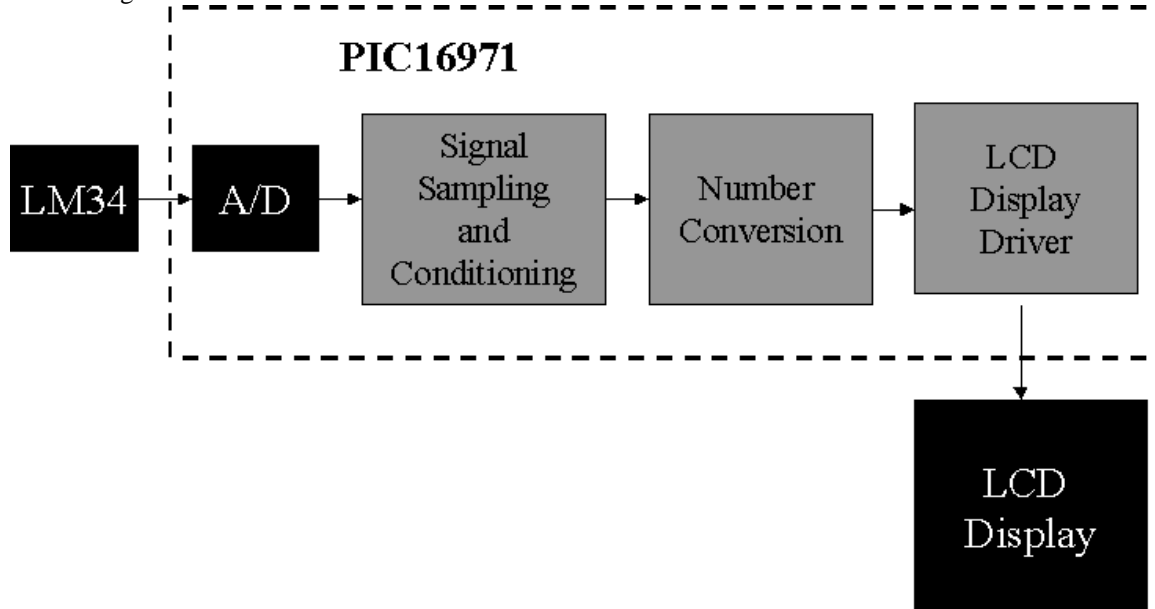


Figure 3 Block Diagram

LM34 –Digital sensor connected to Analog to Digital Converter of PIC16791. PIC16791 is also connected to LCD Display.

Software processes

1. Initialization
 - a. Initializes all I/o including PORTD, PORTA, A/D and variables
 - b. Initializes LCD
2. Signal Sampling and Conditioning
 - a. Properly samples the converter, scales value and adds correction factor
3. Number Conversion
 - a. Converts byte to decimal and then ACSII – three characters worth
4. LCD Display Driver
 - a. LCD8CMD- commands LCD for display size, cursor on/off, display on/off, data transfer size
 - b. LCD8DATA - transfers ASCII data to LCD display
5. Miscellaneous
 - a. Delay subroutines used for LCD timing and loop delays-short and long.

Part 3 Exercise 1 –Flowchart

Review code; identify different segments as described about and then flowchart.

Part 4 Build and Test

Use MPLAB, integrate software with hardware and confirm correct operation. Software source given below.

```
; *****
; PICKit 2 Lesson 4 - "Thermo4"
; Digital Thermometer
; The A2D is referenced to the same Vdd as the device, which
; is provided by the USB cable and nominally is 5V. The A2D
; returns the ratio of the voltage on Pin RA0 to 5V. The A2D
; has a resolution of 10 bits, with 1023 representing 5V and
; 0 representing 0V.

#include <p16F917.inc>
__config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF & _MCLRE_OFF &
_CP_OFF & _CPD_OFF & _BOD_OFF & _IESO_OFF & _FCMEN_OFF)
#define E PORTB, 2
#define RS PORTB,3
    cblock 0x20
Delay1      ; Assign an address to label Delay1
Delay2
Display     ; define a variable to hold the display
Temp
Tempx

Hundreds
Tens
Ones       ; Number ASCII Variables

    endc

    org 0
Start:
    bsf    STATUS,RP0    ; select Register Page 1
    movlw  0xFF
    movwf  TRISA        ; Make PortA all input
    clrf   TRISD        ; Make PortD all output
    movlw  0x01
    movwf  TRISB        ; Make RBO pin input (switch) and
                                ; all rest outputs

    movlw  0x10        ; A2D Clock Fosc/8
    movwf  ADCON1
    movlw  0xFF        ; we want all Port A pins Analoga
    movwf  ANSEL
    bcf    STATUS,RP0  ; back to Register Page 0
    movlw  0x00        ; turn off all leds
    movwf  PORTD
    movwf  PORTB

    movlw  0x85
```

```

    movwf  ADCON0      ; configure A2D for Channel 1 (RA1), right justified, and turn on the A2D
module
    clrf Temp

```

```

clrf Tempx

```

```

    movlw '0'          ; Initialize each of the Variables to "0"
    movwf Hundreds
    movwf Tens
    movwf Ones

```

```

; Initialize the LCD
LCDINIT:

```

```

    call DelayL      ; good long delay to start
    call DelayL
    movlw 0x38       ;enable cursor display
    call LCD8CMD

```

```

    movlw 0x0e       ;set cursor direction
    call LCD8CMD

```

```

    movlw 0x06       ;enable cursor display
    call LCD8CMD

```

```

    movlw 0x01       ;clear display
    call LCD8CMD

```

```

;
    movlw 0x10       ;enable display cursor
    call LCD8CMD

```

```

    movlw 0x03       ;return cursor home
    call LCD8CMD

```

```

    movlw 'T'
    call LCD8Data

```

```

    movlw 'E'
    call LCD8Data

```

```

    movlw 'M'
    call LCD8Data

```

```

    movlw 'P'
    call LCD8Data

```

```

    movlw '='
    call LCD8Data

```

```

    movlw ' '
    call LCD8Data

```

```

    movlw ' '
    call LCD8Data

```

```

    movlw ' '
    call LCD8Data

```

```

MainLoop:          movlw 0xc0
                  call LCD8CMD

nop                ; wait 5uS for A2D amp to settle and capacitor to charge.
nop                ; wait 1uS
nop                ; wait 1uS
nop                ; wait 1uS
nop                ; wait 1uS
nop                ; wait 1uS
bsf    ADCON0,GO_DONE ; start conversion
btfss  ADCON0,GO_DONE ; this bit will change to zero when the conversion is complete
goto   $-1
    bsf    STATUS,RP0 ; select Register Page 1
    movf   ADRESL,w    ; Copy the display to the LEDs

    bcf    STATUS,RP0 ; select Register Page 0
    movwf  Tempx ;
    movlw  0x0a;
    subwf  Tempx
    bcf    STATUS,C
    rrf    Tempx
    movf   Tempx,w
;convert TEMPx to ASCII

    movlw  '0'          ; Initialize each of the Variables to "0"
                    movwf  Hundreds
                    movwf  Tens
                    movwf  Ones

                    call bintoASCII
                    movf  Hundreds,w
                    call LCD8Data

                    movf  Tens,w
                    call LCD8Data

                    movf  Ones,w
                    call LCD8Data
call DelayL
    call DelayL
    call DelayL
    call DelayL
    call DelayL
    call DelayL
    call DelayL
    goto   MainLoop
; Command to LCD
LCD8CMD: bcf RS
        movwf PORTD
        bsf E
        bcf E
        call DelayL
        return

;data to LCD
LCD8Data: bsf RS
        movwf PORTD
        bsf E

```

```

        bcf E
        call DelayS
        return

; Delay Function. Enter with number of 771uS delays in Wreg
DelayL: movlw .20          ; 20 msec
        goto Delay

DelayS: movlw .1          ;771 uss
Delay:
        movwf Delay2      ;
DelayLoop:
        decfsz Delay1,f    ; Waste time.
        goto DelayLoop    ; The Inner loop takes 3 instructions per loop * 256 loopss = 768 instructions
        decfsz Delay2,f    ; The outer loop takes and additional 3 instructions per lap * 256 loops
        goto DelayLoop    ; (768+3) * 256 = 197376 instructions / 1M instructions per second = 0.197 sec.
        ; call it two-tenths of a second.

        return
bintoASCII:
        ; Use temp to convert binary to ASCII
movf Tempx,w
        movwf Temp
        ; movf Temp,w

HundredsLoop:          ; Return Here Each Time to Test for Temp >= 100
        movlw .100       ; Is Temp >= 100?
        subwf Temp, w
        btfsz STATUS, C   ; If STATUS.C Set, then "yes"
        goto TensLoop
        movwf Temp        ; Save the Result
        incf Hundreds, f  ; Increment the Number of 100s
        goto HundredsLoop

TensLoop:              ; Return Here Each Time to Test for Temp >= 10
        movlw .10        ; Is Temp >= 10?
        subwf Temp, w
        btfsz STATUS, C   ; If STATUS.C Set, then "yes"
        goto DoOnes
        movwf Temp        ; Save the Result
        incf Tens, f      ; Increment the Number of 10s
        goto TensLoop

DoOnes:                ; Remainder is the Number of Ones
        movf Temp, w
        addwf Ones, f

        return

end

```