अभियंता

Think outside the ‘bots

Digital Temperature Sensor & PIC

–

Submission File:

Team A:

Devanshi Chauhan
Jainam Shah
Harsh Maghnani

Table Of Content

No.	Topic	Page No.
1	Overview	2
2	Components used	2
3	Softwares used	2
4	PIC16F877	2
5	LM35(Temperature Sensor)	4
6	Interrupts	5
7	Polling v/s Interrupt	6
8	Timer	6
9	ADC (Analog to Digital Converter)	9
10	Task 1.1	12
11	Task 1.2	13
12	Task 1.3	15
13	Conclusion	17
14	References	18

1. Overview

The task is about interfacing a temperature sensor(LM35) with a PIC microcontroller(PIC16f877). PIC microcontrollers are one of the most widely used microcontrollers in industries and education fields[1]. LM35 is a temperature sensor that can be used for various purposes like measuring temperature in boilers, measuring and maintaining ambient room temperature etc.

2. Components used

1. PIC16F877 MCU
2. LM35(Temperature Sensor)
3. 7 Segment Display
4. LEDs
5. Resistors

3. Softwares used

1. MPLAB X IDE(XC8 compiler)
2. PROTEUS

4. PIC16F877

PIC stands for Peripheral Interface Controller which is a family of microcontrollers made by "Microchip Technology". It follows "Harvard architecture" for internal data transfer. Even though there are many types of microcontrollers available, the reason to choose PIC is that it is cheaper, faster, consumes less power and can be easily available[2]. Further, PIC MCUs are classified on the basis of series number, memory, etc. and selection of an MCU that has good online community support and wide applications is a good choice[3]. PIC16F877A and PIC18F4520 are two such MCUs. So, based on our requirements, here PIC16F877 is selected which consists of 40 pins having a variety of functions as shown in Fig.1.0.

Specifications of PIC16F877[4]:

- 8-bit microcontroller with 8k x 14 words of flash program memory, 368 x 8 bytes of data memory and 256 x 8 bytes of EEPROM data memory.
- 20MHz maximum CPU operating speed (200ns instruction cycle).
- Operating voltage range 2.0-5.6 Volts and sink/source current of 25mA.
- Support for 14 interrupts and different types of addressing modes.
- Consists of Power up timer(PWRT) and also 3 timers (either 8-bit or 16-bit).
- 10-bit multi-channel Analog to Digital converter (i.e. 10 bit resolution).
- Universal synchronous asynchronous receiver transmitter (USART).
- Support for I2C and SPI.
- Consists of Brown out reset (BOR) and Watchdog timer.

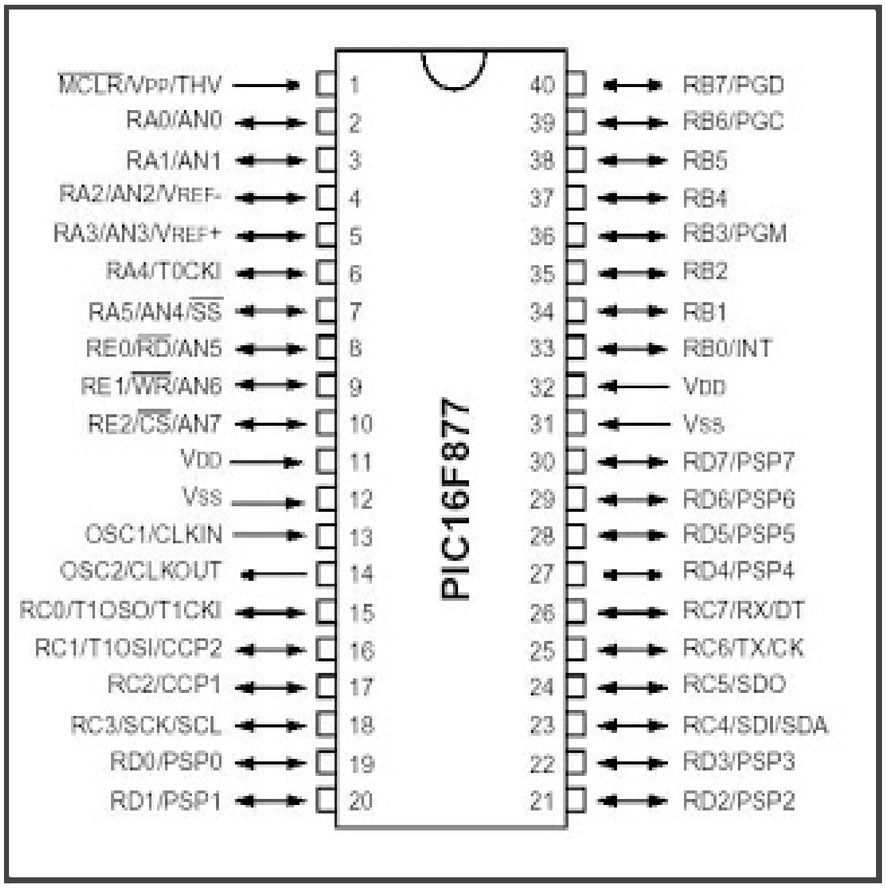


Fig 1.0 Pin Diagram of PIC16F877

5. LM35 (Temperature Sensor)

There are many temperature sensors available out of which LM35 is chosen due to its several advantages over others. The advantage of LM35 over the thermistor is that no external calibration is required and accuracy is higher. Other advantages are that it is protected from self-heating due to the presence of coating, low cost, greater accuracy, low output impedance, linear output and precise inherent calibration makes interfacing to readout or control circuitry easy. Also it is calibrated in Celsius so there is no need of subtracting large numbers. The output voltage in LM35 is linearly proportional to the temperature in centigrade. LM35 has 3-pins namely Vs, ground and Vout as shown in Fig.2[5]. Specifications of LM35 are as mentioned in Table 1.0.

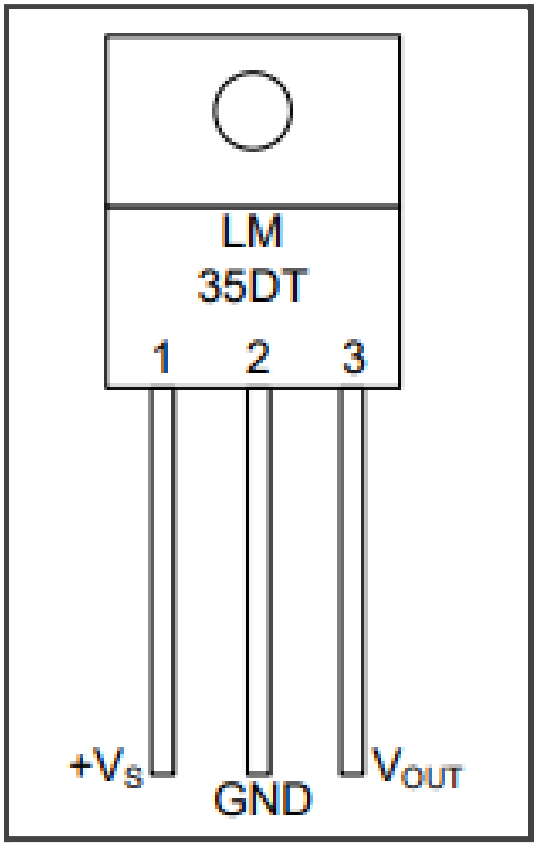


Fig 2.0 LM35 Temperature sensor

Parameter	Value
Sensitivity	10mV/˚C
Accuracy at 25˚C	±0.5˚C
Temperature Range	-55˚C to 150˚C
Accuracy from -55˚C to 150˚C	±1˚C
Output impedance for 1mA load	0.1Ω
Operating Voltage Range	4V to 30V
Non-linearity	±1/4˚C
Output Voltage Range	-1V to 6V

Table 1.0 Specifications of LM35

6. Interrupts

Interrupt is a section of hardware on the microcontroller which is capable of monitoring the state of a microcontroller and can interrupt the microcontroller in its current processes. Whenever an interrupt occurs, a microcontroller calls respective ISR (Interrupt Service Routine), which is nothing but a small function which directs the microcontroller about how to handle it. Interrupts are of two types; hardware and timer. Here we will look upon timer interrupts.

7. Polling v/s Interrupt

There are two methods by which we can communicate between the microcontroller and the external system.

Polling:

The normal processes or work going on in a microcontroller is said to be in polling mode. In this mode, the CPU needs to continuously monitor the overflow flag and take the necessary action when it overflows. During that time period the CPU cannot perform any other task. So the main drawback of polling is that the microcontroller needs to wait and review whether new information has arrived and thus is a waste of time.

Interrupt:

It is a signal sent to a microcontroller to mark the event that requires immediate attention. In order to handle certain conditions properly in major real-time processes, the actual task must be halted or paused for some time so that the microcontroller takes required action and after that it must return to the main task. For executing such types of programs, interrupts are necessary[6].

8. Timer

A timer is a counter which counts till the count reaches the maximum value of timer/counter register and generates an interrupt on its overflow (returns to 0) as shown in Fig.3.0. Using timers we can build very precise time delays which will be based on the system clock and allow us to achieve our desired time delay. Also we can generate delay through timer interrupt with the help of prescaler. Prescaler divides the clock frequency, which helps in defining the execution speed as per user's requirement. There are a total three timers: Timer 0, Timer 1 and Timer 2 out of which Timers 0 and 2 are 8-bit timers and Timer 1 is a 16-bit timer.

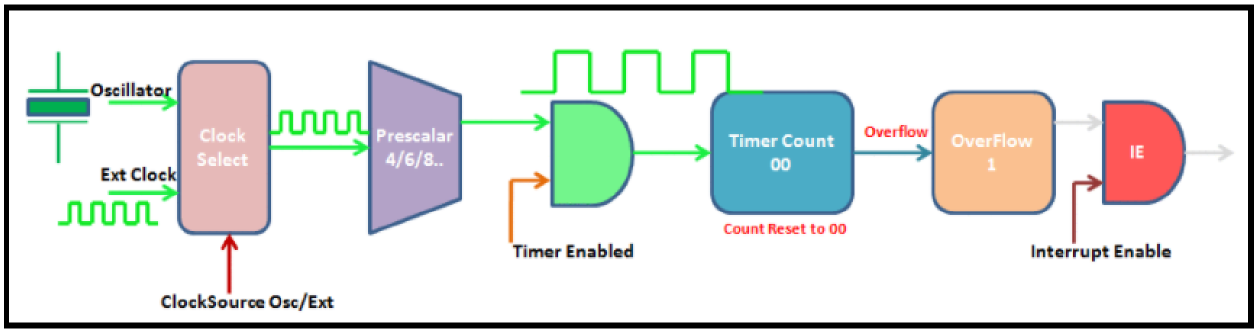


Fig 3.0 Timer block diagram

Registers in Timer-0 module[5]

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
01h,101h	TMR0	Timer0 Module Register							
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Table 2.0 Registers in Timer0

1. **OPTION_REG:** This register is used to configure timer-0 prescaler, clock source etc as shown in Table.3.0.
2. **TMR0:** It holds a timer count which increments on every prescaler count.
3. **INTCON:**This register contains global, peripheral input enable pins along with timer 0 overflow flag and interrupt enable flag bit.

OPTION_REG register[5]:

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Table 3.0 OPTION_REG register bits

Bit 7 $\overline{RBPU}$**:**active low port B bit

1 = pull up disabled

0 = pull up gets activated for port B

Bit 6 INTEDG:enable interrupt at clock edge

Bit 5 T0CS:TMR0 clock source select bit

1 = Transition on T0CKI pin

0 = Internal instruction cycle clock (CLKOUT)

Bit 4 T0SE:TMR0 source edge select bit

1 = increment on high to low transition on T0CKI pin (i.e. on falling edge)

0 = increment on low to high transition on T0CKI pin (i.e. on rising edge)

Bit 3 PSA:prescaler assignment bit

1 = prescaler is assigned to the WDT(watchdog timer)

0 = prescaler is assigned to the timer0 module

Bit 2 PS2:PS0: prescaler rate select bits as shown in Table 4.0

BIT	TMR0 RATE
000	1 : 2
001	1 : 4
010	1 : 8
011	1 : 16
100	1 : 32
101	1 : 64
110	1 : 128
111	1 : 256

Table 4.0 Prescaler rate selection bits

9. ADC (Analog to Digital Converter)

Analog to digital conversion is a process in which we will translate analog values of the signals i.e. temperature,humidity,current,voltages into the digital values.

Types of analog to digital converter:

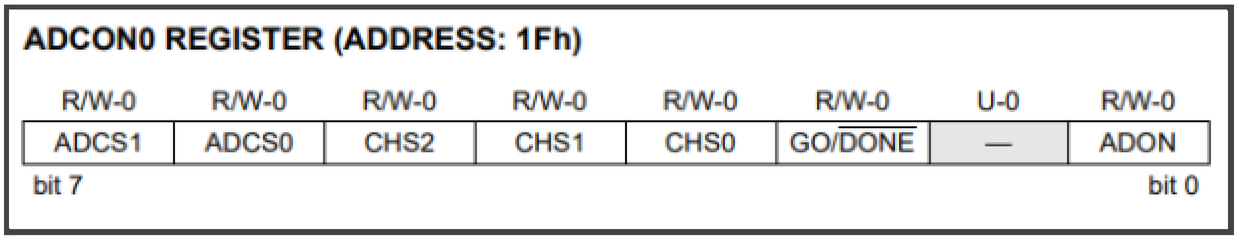
- Successive Approximation(SAR) ADC
- Delta-Sigma ADC
- Dual Slope ADC
- Pipelined ADC
- Flash ADC

ADC in PIC16F877:

In PIC16F877 there are 8 analog input pins available as shown in Fig.4. Analog input signals will charge a sample and hold capacitors. The output of the capacitor will be input of a converter which generates digital result of input via successive approximation. We can select Vdd, Vss or RA2, RA3 pins as reference voltages by configuring registers.

Registers used for A/D module in PIC16F877:

ADCON0 :



ADCON0 REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

Fig 4.0 Register Configuration of ADCON0

ADCS[1:0] (Bit 6 & 7): To select clock frequency

00 : Fosc/2

01 : Fosc/8

10 : Fosc/32

11 : FRC (clock derived from the internal A/D module RC oscillator)

CHS[2:0] (Bit 3,4 & 5):To select Analog channel

000 = channel 0, (RA0/AN0)

001 = channel 1, (RA1/AN1)

010 = channel 2, (RA2/AN2)

011 = channel 3, (RA3/AN3)

100 = channel 4, (RA5/AN4)

101 = channel 5, (RE0/AN5)

110 = channel 6, (RE1/AN6)

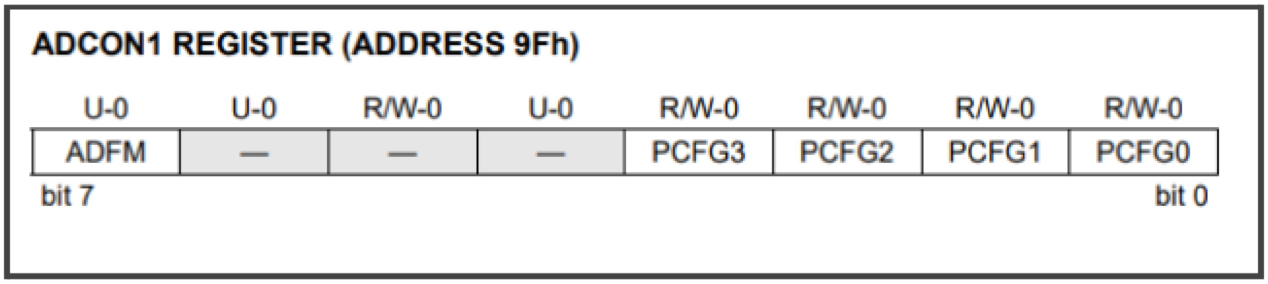
111 = channel 7, (RE2/AN7)

Go/Done (Bit 2):To start A/D conversion

Initially it is 0. But when we assign it 1 our ADC module will start conversion of pin value.

ADON = 1: To operate A/D module

ADCON1:



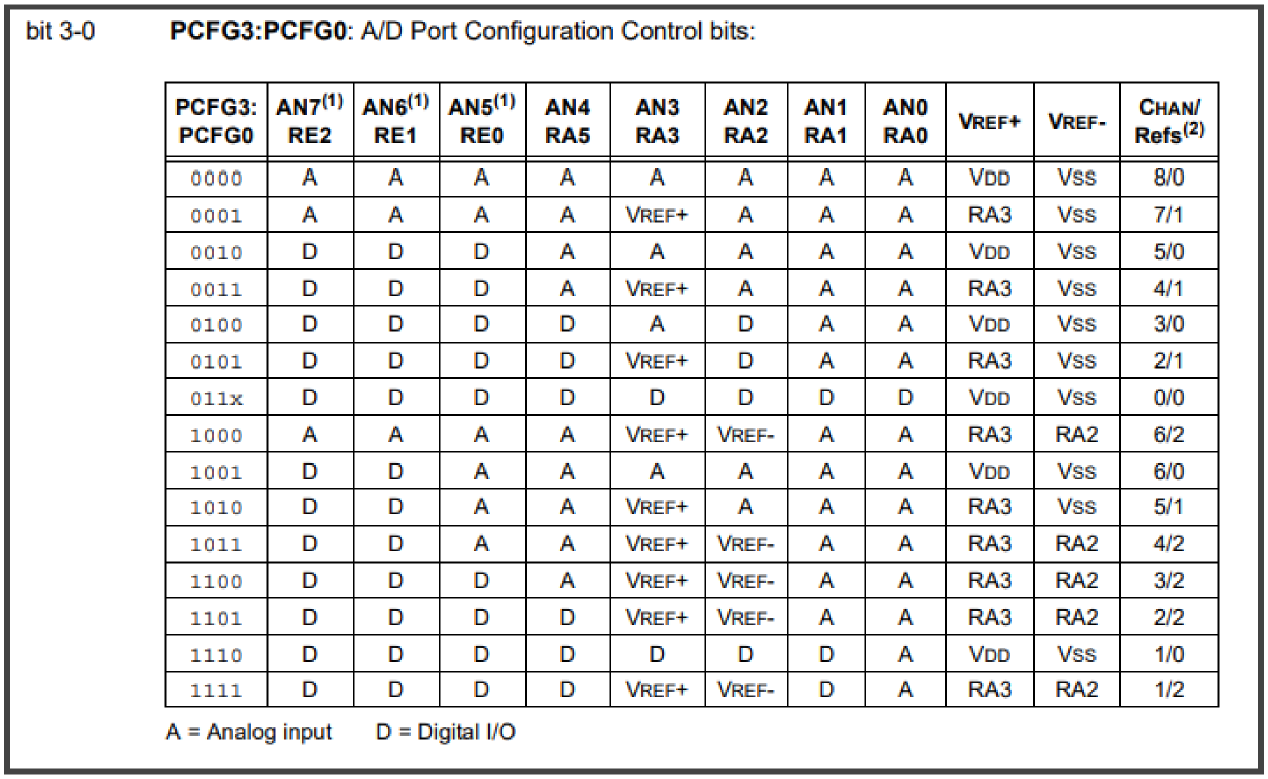
ADCON1 REGISTER (ADDRESS 9Fh)

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Fig 5.0 Register Configuration of ADCON0

ADFM (Bit 7):For generation of ADC result as shown in Fig 5.0

PCFG[3:0] (Bit 0,1,2&3):For assigning analog input pins and to select reference voltages as shown in Table 5.0.



bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN7[1] RE2	AN6[1] RE1	AN5[1] RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs[2]
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Analog input D = Digital I/O

Table 5.0 (Bit selection for different values)

ADRESH and ADRESL:

ADRESH and ADRESL are 8 bits high and low registers respectively from which we can get the result of ADC in form of 10 bits digital value of an analog input. The 10 bits result will store as 16 bits values depending on the value of ADRF as shown in Fig.6.0.

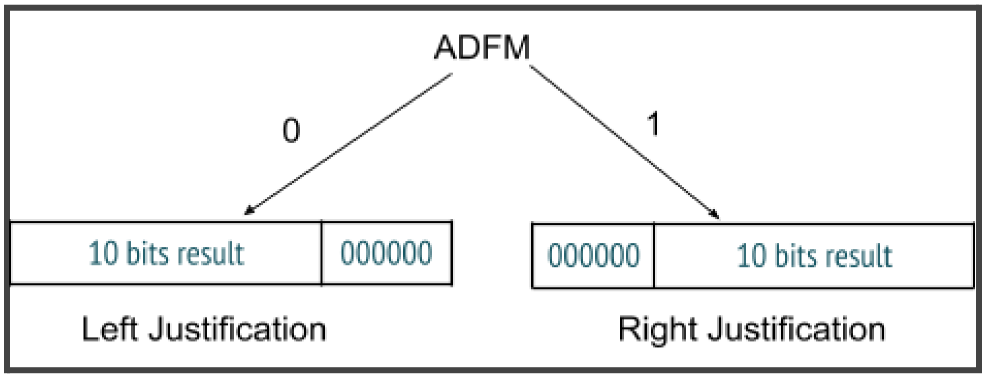


FIG 6.0 ADC result generation

PIE1(ADIE):

To Enable Analog-Digital Converter Peripheral Interrupt.

PIR1(ADIF):

To get the result of ADC converter as a digital value when the value of ADIF is one.We have to assign the value of ADIF as 0 to get in the main code.

10. Task 1.1

Aim:

(A): To blink LEDs in Proteus by referring the datasheet of PIC16F877 and implementing a simple sequential display of three LEDs in pattern as follows:

1,2,3

3,2,1

1,2,3

3,2,1…

(B): To interface SSD with PIC16f877 in order to display any number on it. (Without Using Encoder and any inbuilt library)

Implementation:

Approach for task A:

- In this task the goal was to blink the 3 LEDs in a specific pattern that was 1,2,3,3,2,1 and should keep following this pattern for infinite times.
- So we approached the task by deciding the speed at which the program execution should take place and set crystal frequency to 20MHz.
- Then we decided the output port and connected the 3 LEDs to PIC16F877 in Proteus.
- In order to blink the LEDs, we used the inbuilt macro __delay_ms() to generate delay between the LEDs.
- Later we replaced the inbuilt macro with user defined methods for generating delay.

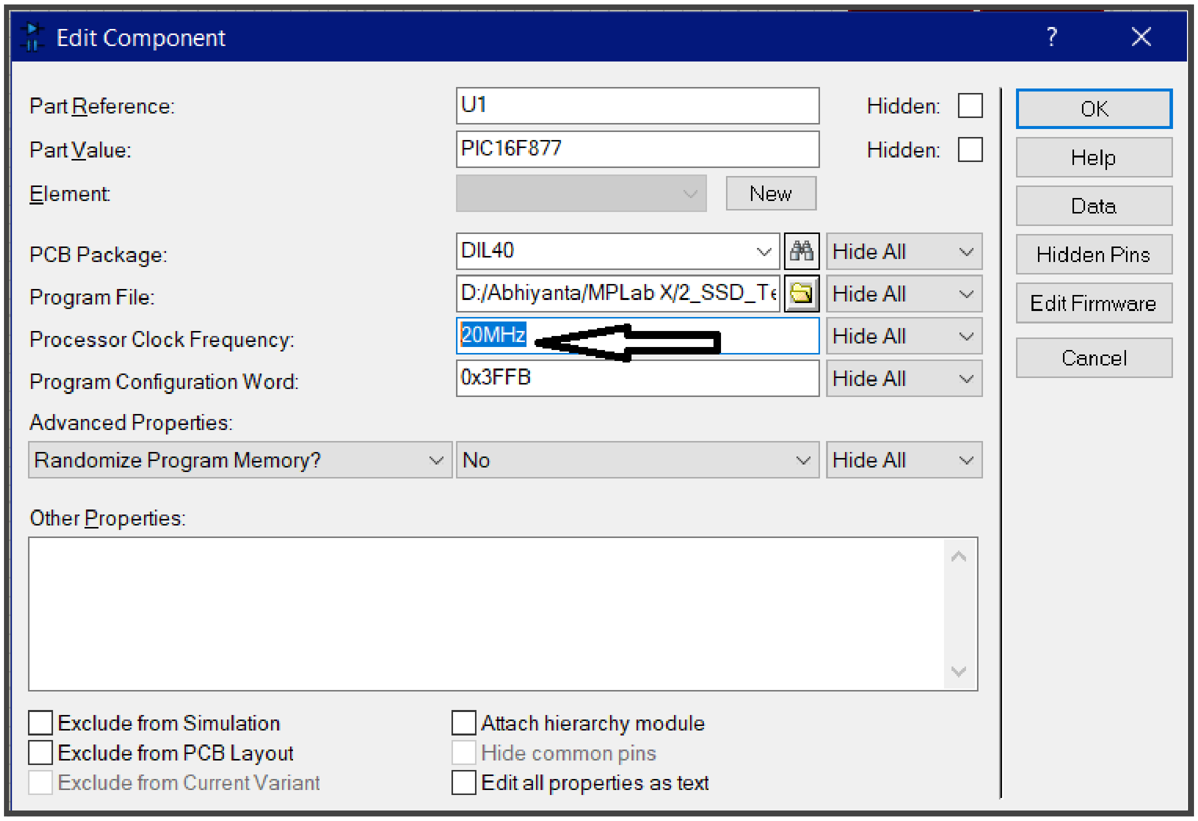
Approach for task B:

- In this task, for interfacing SSD with pic16f877 microcontroller, first we connected it to port B of MCU and assigned it as output port.
- Then we set the internal clock frequency as 20MHz and to display count we converted the count values from 0-9 in binary or hexadecimal form and loaded the value on the output port.

- In an infinite while loop we assigned values to the respective LEDs of the SSD and with the inbuilt macro __delay_ms() or else user defined way, generated delay between the counts to be displayed on SSD.

Troubleshooting:

We were not getting delay as defined in the argument of function __delay_ms(arg). We then set the on board internal frequency of PIC16F877 from proteus as written in code. E.g. We had given frequency of 20MHz in our code so then we set it in proteus as shown in Fig.7.0.


Edit Component

Part Reference:	U1	Hidden: ☐	
Part Value:	PIC16F877	Hidden: ☐	
Element:		New	
PCB Package:	DIL40		Hide All
Program File:	D:/Abhiyanta/MPLab X/2_SSD_Te		Hide All
Processor Clock Frequency:	20MHz		Hide All
Program Configuration Word:	0x3FFB		Hide All
Advanced Properties:			
Randomize Program Memory?	No		Hide All

Other Properties:

☐ Exclude from Simulation
☐ Exclude from PCB Layout
☐ Exclude from Current Variant
☐ Attach hierarchy module
☐ Hide common pins
☐ Edit all properties as text

Fig 7.0 PIC16F877 Properties in Proteus

11. Task 1.2

Aim:

(A): To increment the count in SSD by using timer based interrupts with a delay of 1 sec.

(B): To extend Task 1.2(A) by adding a blinking led pattern as given in Task 1.1(A) with the use of polling mode.

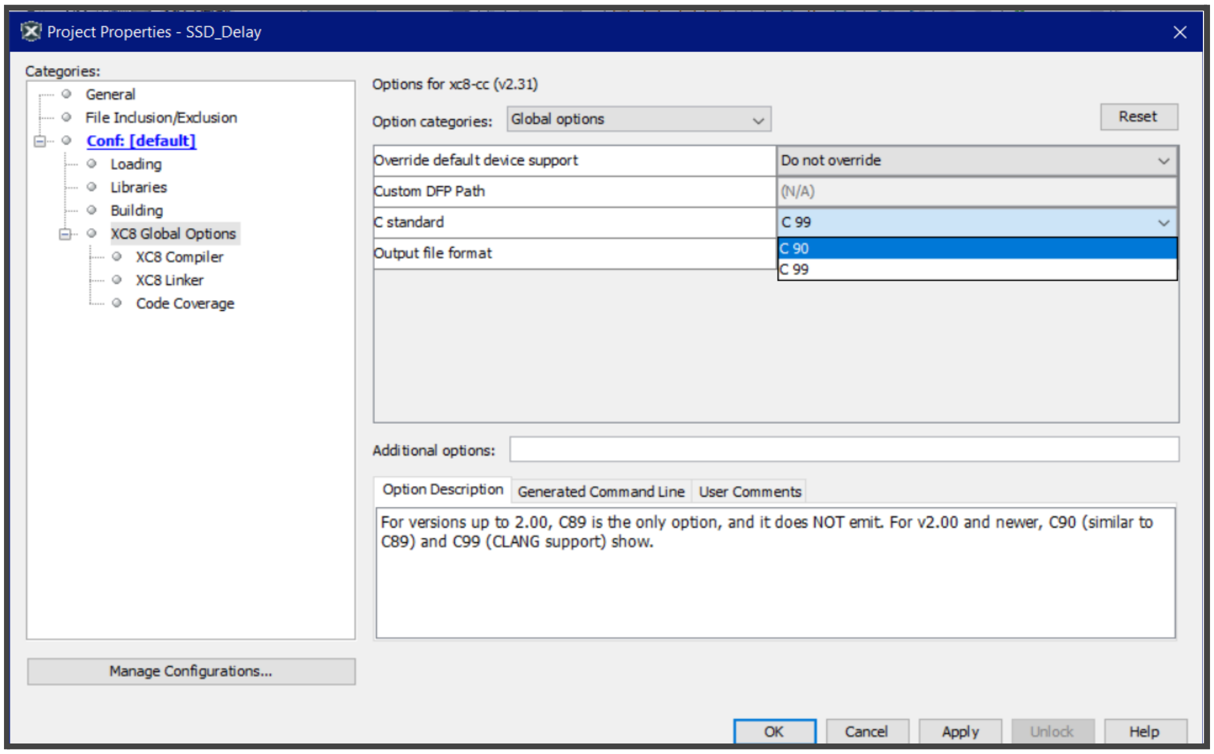
Implementation:

Approach for task (A) & (B):

- Here we used timer 0 which is 8 bit timer.
- Control register for timer0:- OPTION_REG
- Count register:- TMR0
- The maximum delay timer 0 can generate is 13.107ms, so for generating 1sec delay we went for 10ms delay 100 times i.e. 100 x 10ms = 1sec.
- The prescaler is selected as 1:256
- Time delay = (255 -TMR0)*prescaler*machine cycle
- Machine cycle = 4/20MHz
- The required delay is 10ms and at prescaler of 256 the count to be loaded in TMRO comes out to be 59, that means it will take 10ms to reach from 59 to 255 and then will overflow.
- We connected a seven segment display at the port B, and set 5V input at pin 1(VCC to microcontroller) of the microcontroller.
- We defined a counter variable which counts 10ms delay, a total of 100 times in order to generate 1sec delay.
- Initially, we kept count to 0 and generated an ISR function which interrupts the execution, changes the flag value and increments the count value. After that in the main function we enabled timer 0 and all the global and peripheral interrupts.
- As soon as the value of the counter variable reaches 100 (i.e after total delay 100*10ms = 1s), the count on seven segment is incremented by 1 and then the counter variable is set to 0. This process is kept in an infinite while loop to keep on counting.
- Likewise, for task (B) we have to keep above connections along with our SSD to run both the LED pattern and the SSD simultaneously.

Troubleshooting:

We can’t use the global variable here by default setting from MPLAB X XC8 compiler. Right click on your project and go to properties. Change C standards as shown in Fig.8.0.


Project Properties - SSD_Delay

Categories:
- General
- File Inclusion/Exclusion
- Conf: [default]
  - Loading
  - Libraries
  - Building
  - XC8 Global Options
    - XC8 Compiler
    - XC8 Linker
    - Code Coverage

Options for xc8-cc (v2.31)

Option categories: Global options

Override default device support	Do not override
Custom DFP Path	(N/A)
C standard	C 99
Output file format	

C 90
C 99

Additional options:

Option Description | Generated Command Line | User Comments

For versions up to 2.00, C89 is the only option, and it does NOT emit. For v2.00 and newer, C90 (similar to C89) and C99 (CLANG support) show.

Fig.8.0 Properties of project in MPLAB X

12. Task 1.3

Aim:

(A): To interface LM35 temperature sensor in order to observe following output:

Output	Condition
LED 1 (Blinking)	35 ˚C< Temperature < 45˚C
LED 2 (Blinking)	Temperature < 20˚C
All the Leds should be ON	If the temperature reaches either of the extremes

(B):To display the temperature (0-99) using 2 SSD connected with MCU using only 9 digital I/O pins. If the value of the temperature does not fall in this range, allocate a specific led which will turn on.

Implementation

Approach for task (A) & (B):

- LM35 gives analog output with respect to the temperature, to understand the proper working we have to refer to the datasheet of LM35.
- In the circuit we will apply Vcc=5V to LM35 and connect the digital output pin to input of RA0/AN0.
- Now we will configure our registers as our requirement

```
ADCON0 = 0x41;     //8-bits = 01000001
ADCON1 = 0x8E;     //8-bits = 10001110
```

Eg.1 Code for configuration of register

- CHS2:CHS0 = 000 //select channel in PIC RA0/AN0.
- We will assign the value of ADCON0 register = 0x41(To turn on ADC) to 0x45(To start ADC) as discussed in ADC.
- ADCON1 register is configured for ADC process and output generation. ADCON1 = 0x8E, so AN0/RA0 pin considered as analog input and Vcc - Vss are considered as reference Voltages.ADFM = 1 in ADCON1 register for generating the result of ADC in 10 bits as right justification.
- After completion of the ADC process ADIF value will become 1. So we can use interrupt by enabling ADIE =1.After it we have to make sure that the value of ADIF will become 0.

```
void interrupt ADC(){
   if(PIR1bits.ADIF == 1){
      digital = (ADRESH<<8);
      digital = digital + ADRESL;
      PIR1bits.ADIF = 0;
   }
}
```

Eg 2.0 Code For A/D interrupt

- From this Digital value we can measure temperature according to our scaling factor.
- Calculation for Temperature
   1. In this case step value of 10 bit resolution is = 5 V/(1024-1) bits= 4.8875mV
   2. LM35 will change its O/P 10mVC.
   3. So ADC to temp. conversion factor = 10/4.8875= 2.046
   4. Our Temperature = ADC value/2.046
- Once we get the value of temperature we can use it in various applications.

Troubleshooting:

We were unable to turn on leds at extreme temperatures i.e. -55˚C & 150˚C.

13. Conclusion

In this task we learnt about two basic components that is PIC16F877 microcontroller and LM35 temperature sensor. In PIC16F877 we learnt about some of the features of it like the timers, ADC module, etc. and used these concepts for several applications. There are three timers in PIC16F877 and are 8 bits or 16 bits. Also it consists of 10 bits of ADC module, having resolution of 4.8875mV, performs analog to digital conversion using successive approximation method. We also learnt about LM35 temperature sensor that has sensitivity of 10mV/C due to which it provides higher accuracy in temperature measurement. LM35 can measure temperature in range of -55˚C to 150˚C. The resolution of device is 0.48875 °C.

Calculation for resolution of device:

Step value for ADC = 4.8875mV

Vout change for LM35 per 1 °C = 10mV

Resolution : 10mV → 1 °C temp. Change

4.8875mV → ?

Minimum temp. detect by device = 0.48875 °C

14. References

01. PIC as one of the most widely used MCU

02. Reason to choose PIC over other controllers

03. Reason to choose PIC16F877

04. PIC16F877 Datasheet

05. LM35 Datasheet

06. Interrupt

07. Timers

08. http://www.microcontrollerboard.com/pic_interrupt.html

09. https://www.exploreembed1ded.com/wiki/PIC16f877a_Timer

10. https://www.sciencedirect.com/topics/engineering/prescaler

11. https://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

12. https://microcontrollerslab.com/timers-pic-microcontroller-delay/

13. https://openlabpro.com/guide/pic16f877a-timer/