MICROWAVE_OVEN

In Embedded System Programming

Submitted By:-

Trupti Balkrishan Deore
Department of Electronics &Telecommunication.
truptideorepatil18@gmail.com

INDEX

1. Proposed system

➢ Have to Implement Microwave_Oven in Following Basic Mode:-

I. Microwave Mode
II. Grill Mode
III. Convection Mode
IV. Start

All this mode of Microwave Oven are implemented on PIC16f877A.and user can select any mode as per there requirment by pressing keys. Here we are using the simulator called PicSim lab. And the programming lagnguage we have used is Embedde C.

Have to set the working of mode as mentioned bellow:

- Microwave Mode:

In this mode we have to set power level of cooking. In which heating is done by microwave and we have to implement such a system that will set cooking time in micromode and after completion of given time it have to give alert.

- Grill Mode:

In this mode we have to set up timmer for cooking by pressing keys. We have several keys on board so have to set different function to different keys.

- Convection Mode:-

Convection Mode enables you to cook food in the same way as in a traditional oven. The microwave mode is not used.You can set the temperature, as required, in a given range. The maximum cooking time is 180sec. We have keep door close if the door is open it have to give alret and turn on the buzzer and turn off the fan.

Features:

- Proposed system has 4 diffrent modes
- We have time counter for each mode
- We can set temperature and timer manually
- Easy Design

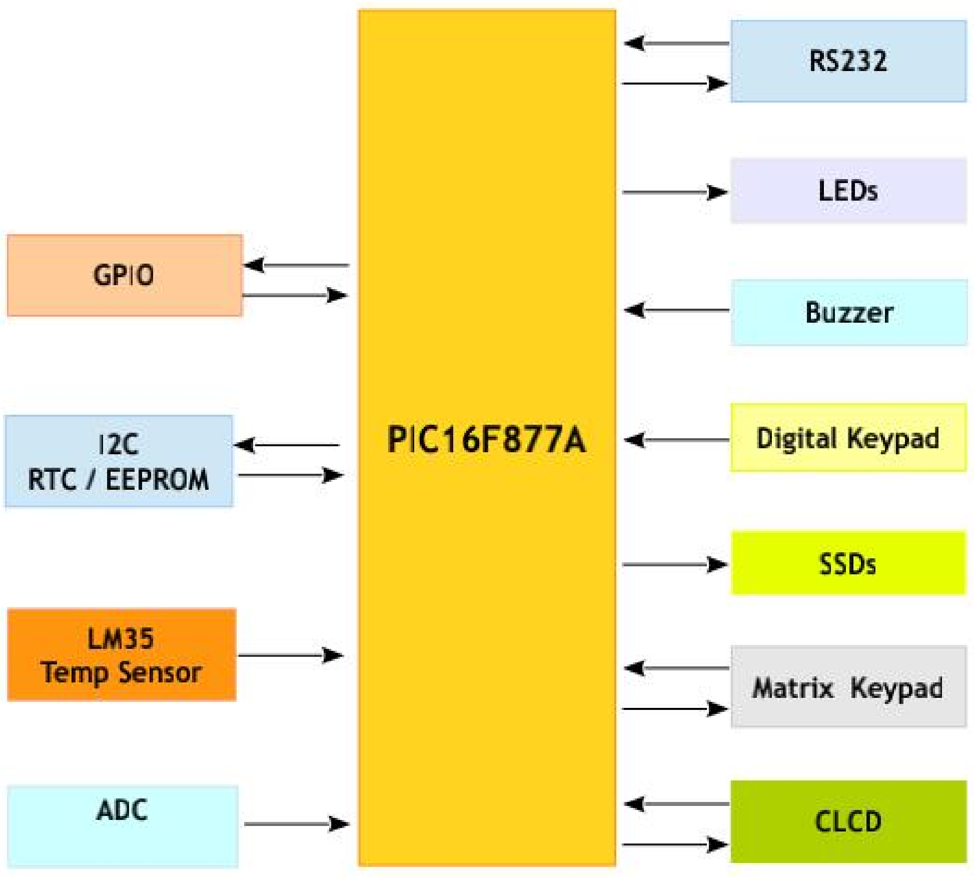
2. Designing

Here we designe a Microwave_Oven using PIC16F877A Microcontroller Board.

2.1 Hardware Logic Design[Architecture]:

2.1.1 Microcontrollers:-

It is good to know what your target board is, what it contains by its architecture. Board architecture generally gives you overview about your board and its peripheral interfaces. In our case we will be using PICSimLab simulator board.



Microcontroller PIC16F877A is one of the PICmicro Family microcontrollers which is popular at this moment, start from beginner until all professionals. Because very easy to use PIC16F877A and use FLASH memory technology so that can be write-erase until thousand times. The superiority this Risc Microcontroller compared to with another microcontroller 8-bit especially at a speed of and his code compression.

The 16F877A is a capable microcontroller that can do many tasks because it has a large enough programming memory (large in terms of sensor and control projects) of 8k words and 368 Bytes of RAM.

Features of PIC16F877A (PIC16F877A Introduction):-

The PIC16F877A CMOS FLASH-based 8-bit microcontroller is upward compatible with the PIC16C5x, PIC12Cxxx, and PIC16C7x devices. It features 200 ns instruction execution, 256 bytes of EEPROM data memory, self-programming, an ICD, 2 Comparators, 8 channels of 10-bit Analog-to-Digital (A/D) converter, 2 capture/compare/PWM functions, an asynchronous serial port that can be configured as either 3-wire SPI or 2-wire I2C bus, a USART, and a Parallel Slave Port.

1. High-Performance RISC CPU
2. Lead-free; RoHS-compliant
3. Operating speed: 20 MHz, 200 ns instruction cycle
4. Operating voltage: 4.0-5.5V
5. Industrial temperature range (-40 to +85C)
6. 15 Interrupt Sources
7. 35 single-word instructions
8. All single-cycle instructions except for program branches (two-cycle)

- **Special Microcontroller Features**

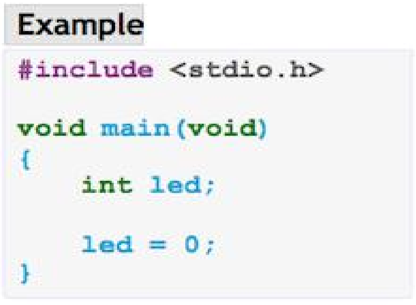
1. Flash Memory: 14.3 Kbytes (8192 words)
2. Data SRAM: 368 bytes
3. Data EEPROM: 256 bytes
4. Self-reprogrammable under software control
5. In-Circuit Serial Programming via two pins (5V)
6. Watchdog Timer with on-chip RC oscillator
7. Programmable code protection
8. Power-saving Sleep mode
9. Selectable oscillator options
10. In-Circuit Debug via two pins

- **Peripheral Features**

1. 33 I/O pins; 5 I/O ports
2. Timer0: 8-bit timer/counter with 8-bit Prescaler
3. Timer1: 16-bit timer/counter with Prescaler
4. Can be incremented during Sleep via external crystal/clock
5. Timer2: 8-bit timer/counter with 8-bit period register, Prescaler, and postscaler
6. Two Capture, Compare, PWM modules
7. 16-bit Capture input; max resolution 12.5 ns
8. 16-bit Compare; max resolution 200 ns
9. 10-bit PWM
10. Synchronous Serial Port with two modes:
11. SPI Master
12. I2C Master and Slave
13. USART/SCI with 9-bit address detection

2.2 Software Logic Design:

Embedded Programming Let’s start understanding software part,
So from the architecture we come to know that the board has few LEDs, So why don't we start with it



Example

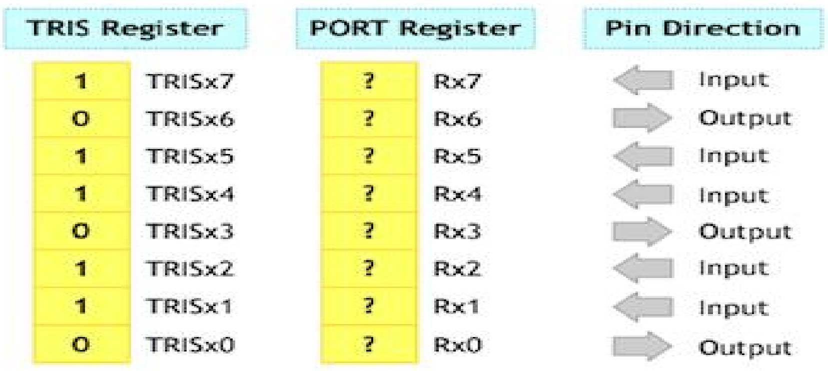
```
#include <stdio.h>

void main(void)
{
    int led;

    led = 0;
}
```

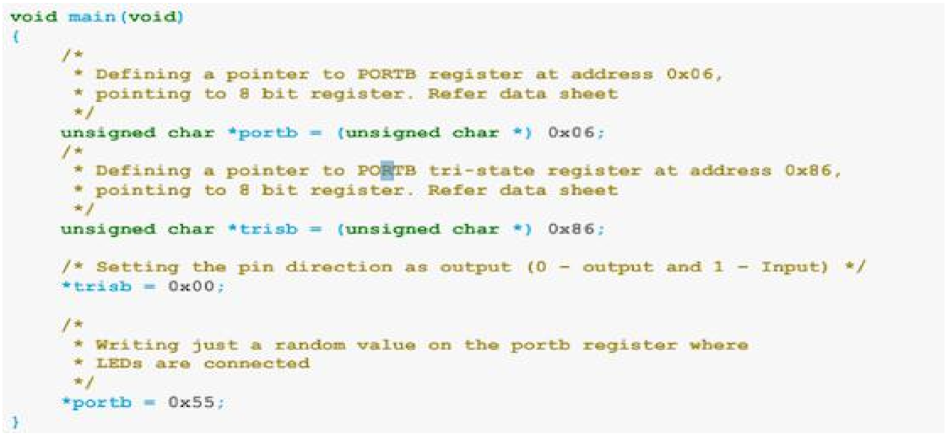
So simple right? Well I hope you know what’s going happen with this code!!
Any C programmer knows that the led is just a integer variable and we write just a value in it, hence no point in this code.

● LED is an external device connected to microcontroller port
● A port is interface between the controller and external peripheral.
● Based on the controller architecture you will have N numbers of ports
● The target controller in PICGenios board is PIC16F877A from Microchip
● The next question arises is how do I know how many ports my target controller has?
– From Microcontroller Architecture which will be detailed in the data sheet provided by the maker
● By reading the data sheet you come to know that there are 5 Ports
● Again a question. Where are the LEDs connected. You need the Schematic of the target board to know this.
● A Schematic is document which provides information about the physical connections on the hardware.
● From the schematic we come to know the the LEDs are connected to PORTB and PORTD
● Port is a peripheral and we need need to know on how to access and address. This info will be available in the data sheet in PORTB, PORTD and Data Memory sections From the section of PORTB it clear that there is 1 more register associated with it named, TRISB
● The TRISB register is very important for IO configuration. The value put in this register would decide pin direction as shown below



TRIS Register		PORT Register		Pin Direction
1	TRISx7	?	Rx7	⇦ Input
0	TRISx6	?	Rx6	⇨ Output
1	TRISx5	?	Rx5	⇦ Input
1	TRISx4	?	Rx4	⇦ Input
0	TRISx3	?	Rx3	⇨ Output
1	TRISx2	?	Rx2	⇦ Input
1	TRISx1	?	Rx1	⇦ Input
0	TRISx0	?	Rx0	⇨ Output

So from previous slide its clear that we have to use the TRIS register to control the pin direction
● LEDs ardriven by external source, so the port direction should be made as output
● In this case the LEDs are connected to the controller and will be driven by it
● Fine, what should write to the port to make it work? It depends on the hardware design.
● By considering all these point we can modify our code as hown in the next slide

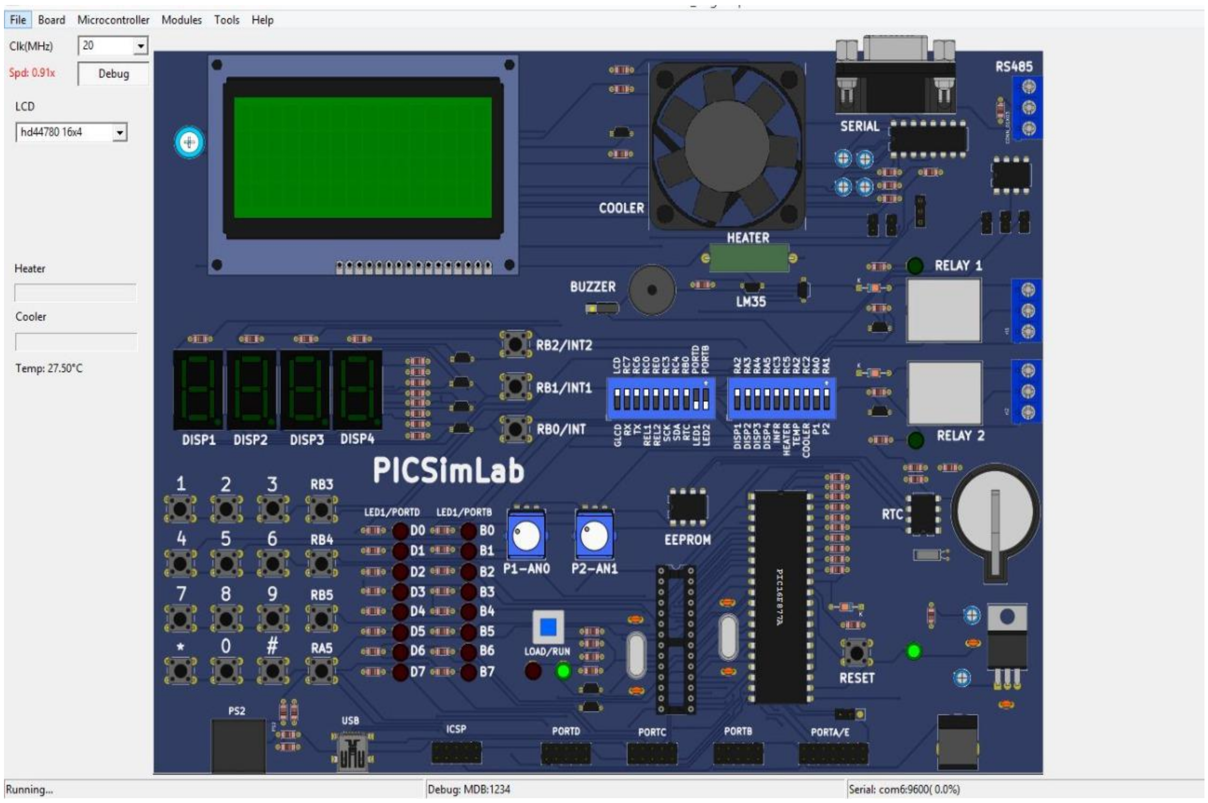


```
void main(void)
{
    /*
     * Defining a pointer to PORTB register at address 0x06,
     * pointing to 8 bit register. Refer data sheet
     */
    unsigned char *portb = (unsigned char *) 0x06;
    /*
     * Defining a pointer to PORTB tri-state register at address 0x86,
     * pointing to 8 bit register. Refer data sheet
     */
    unsigned char *trisb = (unsigned char *) 0x86;

    /* Setting the pin direction as output (0 - output and 1 - Input) */
    *trisb = 0x00;

    /*
     * Writing just a random value on the portb register where
     * LEDs are connected
     */
    *portb = 0x55;
}
```

We wrote our first Embedded C code for our target board
● Come on let's move forward, how do I compile this code? Obviously with a compiler!, Yes but a cross compiler since this code has to run on the target board.
● The target controller, as mentioned, is by Microchip. So we will be using XC8 (Free Version)
● You need to download it and install it in your system – You can use MPLABX

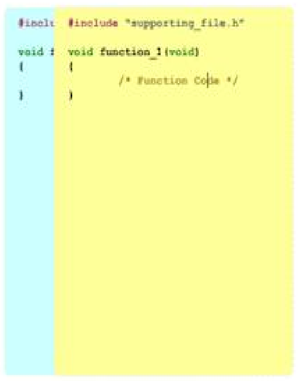


Clk(MHz) 20
Spd: 0.91x
LCD hd44780 16x4
Temp: 27.50°C
Running...
Debug: MDB:1234
Serial: com6:9600(0.0%)

2.3 Project Creation - Code Organization

Let's see how to Orgnize code:

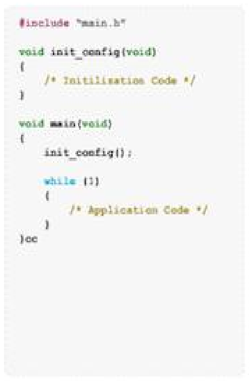
- Please organize the code as shown below to increase productivity and modularity
- Every .c file should have .h file



```c
#include "supporting_file.h"

void function_1(void)
{
        /* Function Code */
}
```

modules.c



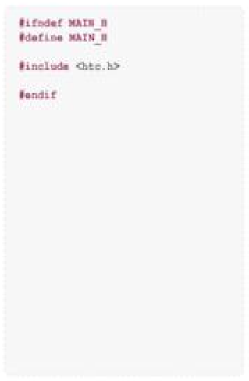
```c
#include "main.h"

void init_config(void)
{
    /* Initilization Code */
}

void main(void)
{
    init_config();

    while (1)
    {
        /* Application Code */
    }
}oc
```

main.c

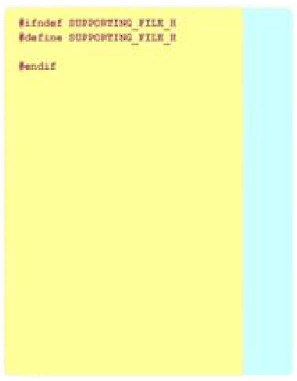


```c
#ifndef MAIN_H
#define MAIN_H

#include <htc.h>

#endif
```

main.h



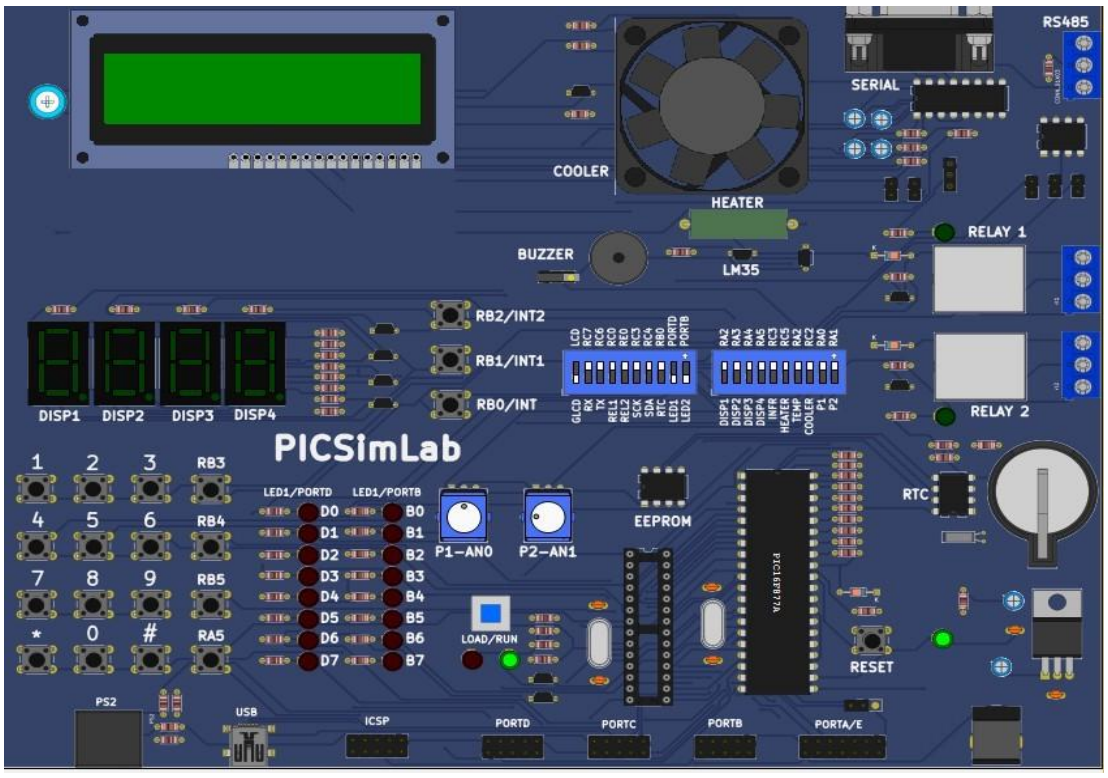
```c
#ifndef SUPPORTING_FILE_H
#define SUPPORTING_FILE_H

#endif
```

modules.h

2.3.1 Interfacing

Lets Role Out on Interfaces



- Following are the important componets let’s see the interfacing of it one-by-one
- LEDs
- Digital Keypad
- Interrupts
- Timers
- Clock I/O
- SSDs
- CLCD
- Matrix Keypad
- Analog Inputs

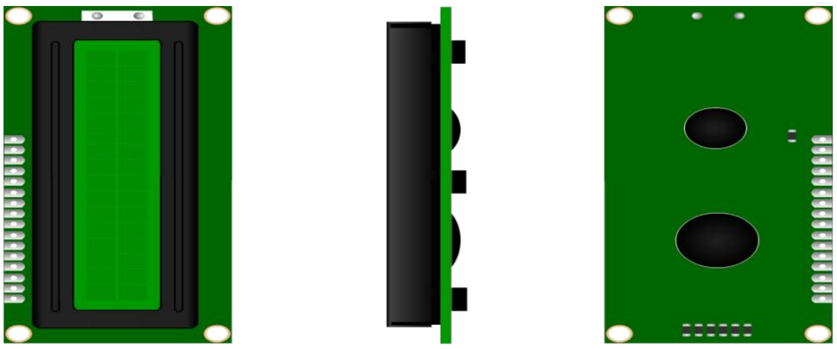
1. **Light Emitting Diodes:**
   Simplest device used in most on the embedded applications as feedback

- Works just like diodes
- Low energy consumptions, longer life, smaller size, faster switching make it usable in wide application

2. **Character Liquid Crystal Display:**
   Most commonly used display ASCII characters

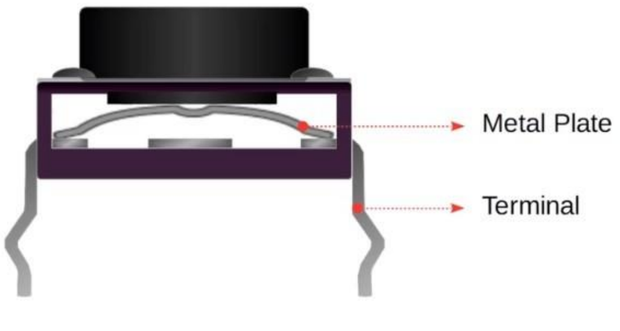
- Some customization in symbols possible
- Communication Modes

– 8 Bit Mode
– 4 Bit Mode



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Vdd	Vdd	Vo	RS	R/W	E	D0	D1	D2	D3	D4	D5	D6	D7	A	K

3. **Tactile Switch:**

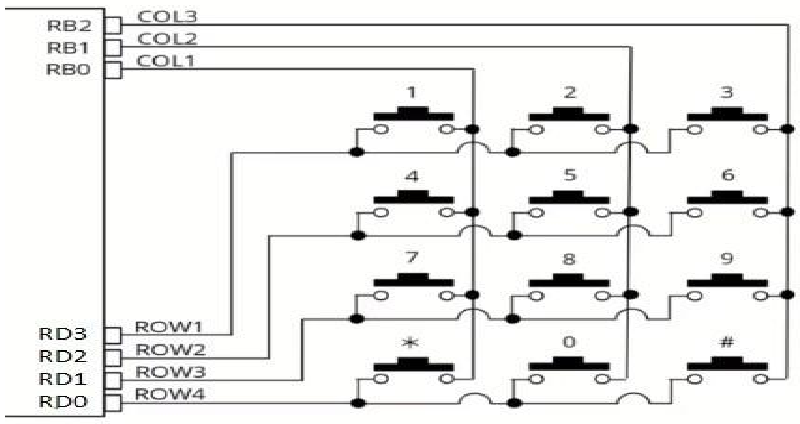


4. **Matrix Keypad:**

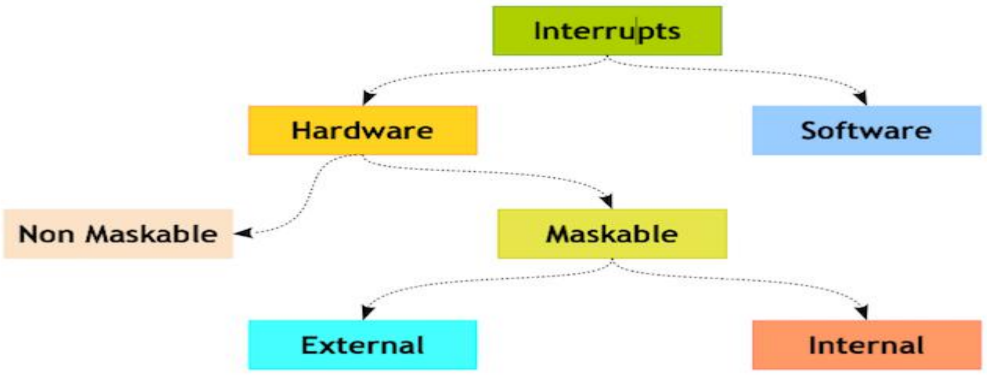
Used when the more number of user inputs is required and still want to save the some controller I/O lines

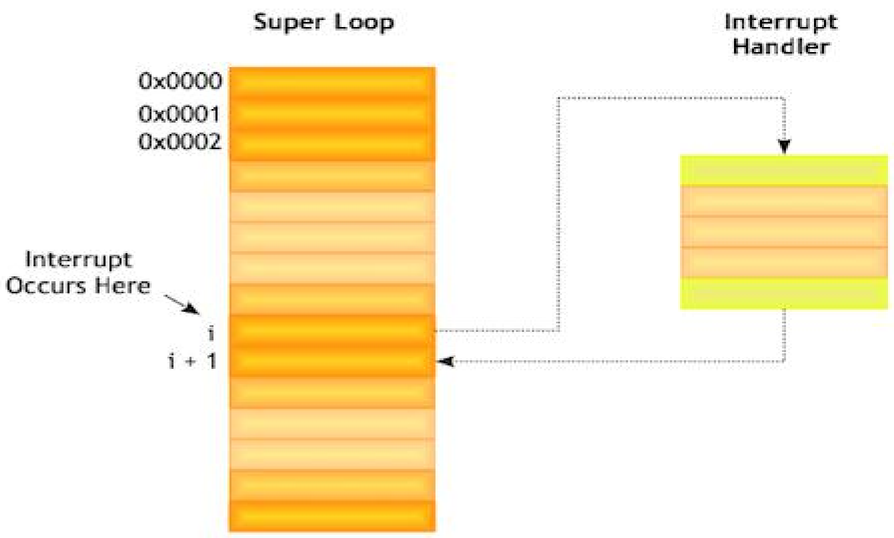
● Uses rows and columns concept

● Most commonly used in Telephonic, Calculators, Digital lockers and many more applications



2.3.2 Interrupts





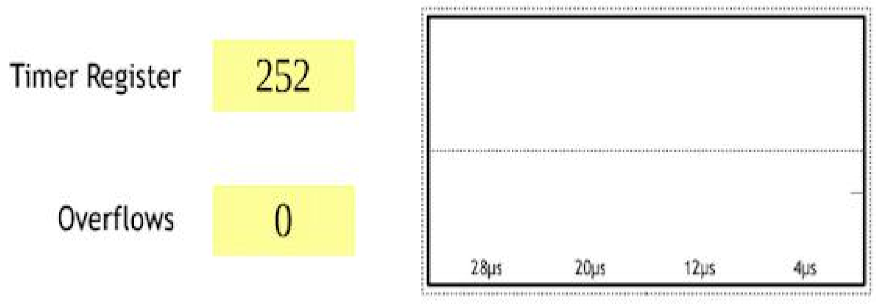
2.3.3 Timers

- Resolution:-Register Width
- Tick:- Up Count or Down Count
- Quantum:-System Clock settings
- Scaling:Pre or Post
- Modes

1. Counter
2. PWM or Pulse Generator
3. PW or PP Measurement etc.,

Examples:

- Requirement – 5 pulses of 8 µsecs
- Resolution – 8 Bit
- Quantum – 1 µsecs
- General



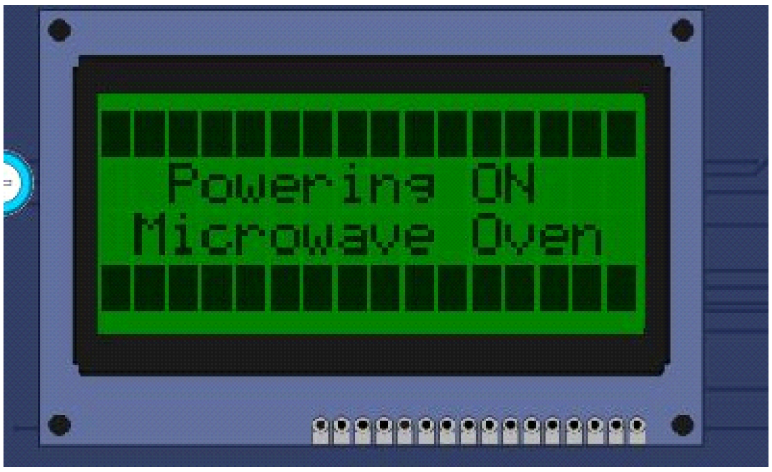
3. Building [Implimentation]

There are four diffrent modules in Microwave_Oven Simulator

- Microwave Mode
- Grill Mode
- Convection Mode
- Start

- **Power On Screen:**

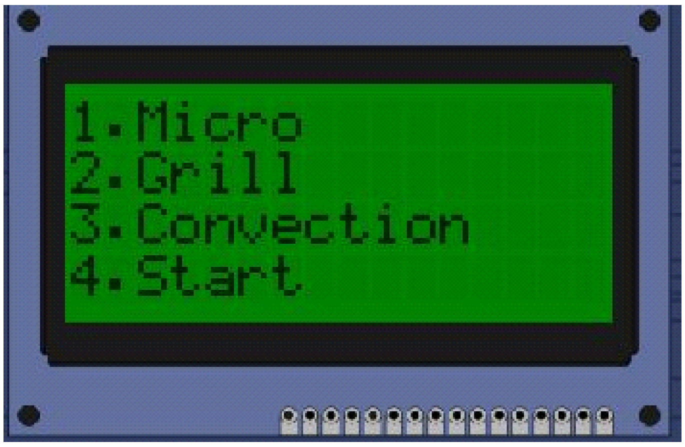
When we turn on the microwave oven it will show as on screen message like "Power ON Microwave Oven"



```
Powering ON
Microwave Oven
```

- **Cooking Mode Display:**

Power on screen will visible for some time after that we will see the cooking moode display screen like bellow:



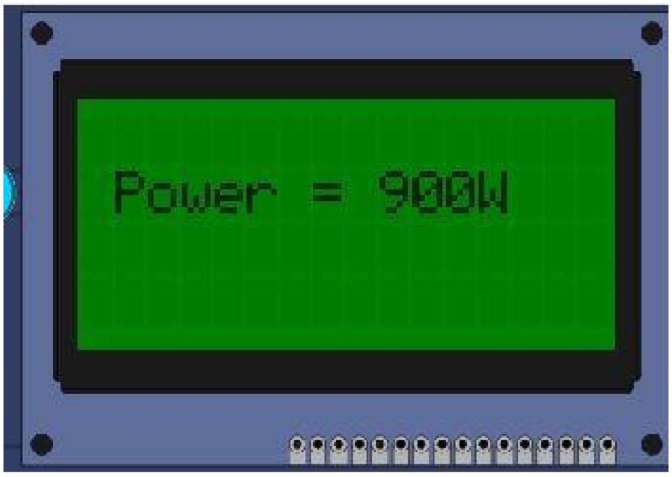
```
1.Micro
2.Grill
3.Convection
4.Start
```

From this we can Select a respective Mode of cooking by pressing key's   we will see each mode one by one:

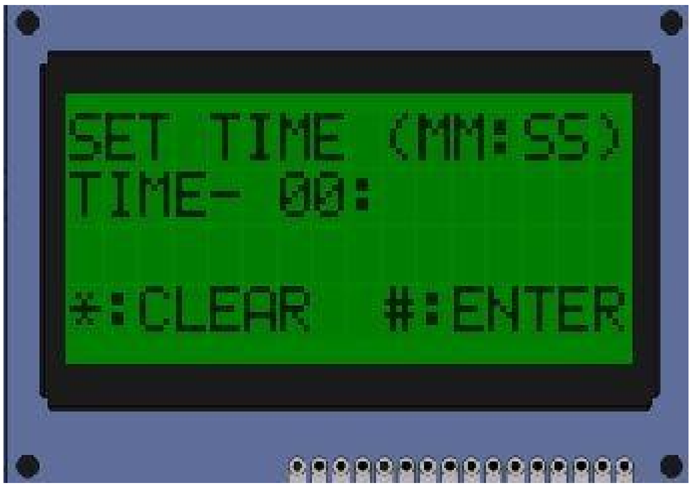
- **Microwave Mode:**

Let's see how we implemented a micro mode:

If we pressed (key ==1) that mence we select micro mode. By entering in micro mode it will 1st display a power level of oven then goes in cooking mode display that there we have to set timmer and start our oven as soon as we start Fan will turn on as well as timer is ON.



```
Power = 900W
```



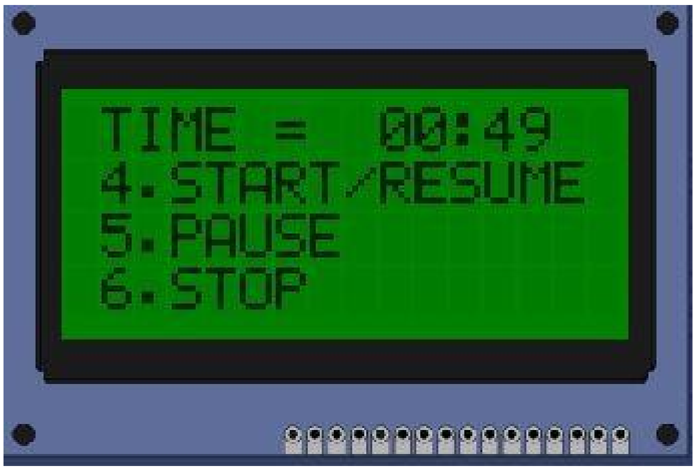
```
SET TIME (MM:SS)
TIME- 00:

*:CLEAR   #:ENTER
```

- **Grill Mode:**

Let's see how we implemented a grill mode:

If we pressed (key ==2) that mence we select Grill mode.as same privious mode we have to set a grilling time at same time it will turn ON the fan and after some time when cooking is deone it will turn OFF the fan and turn ON buzzer and give message on screen i.e. "Cooking time up".

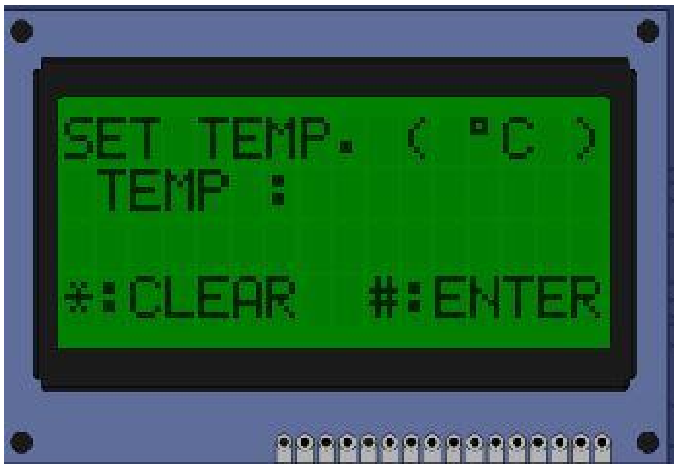


```
TIME =  00:49
4.START/RESUME
5.PAUSE
6.STOP
```

● Convection Mode:

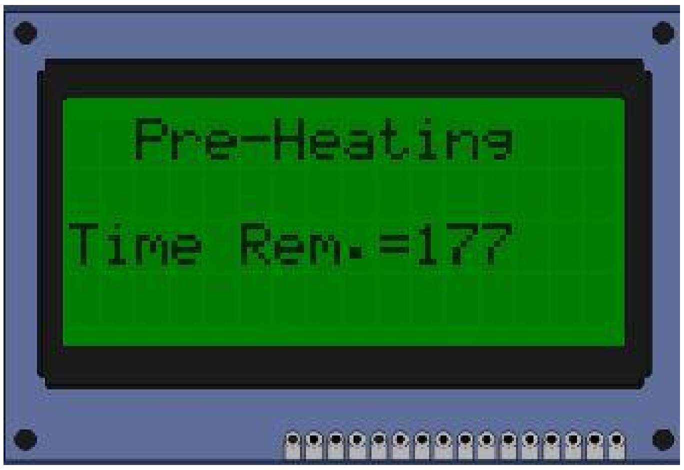
Let’s see how we implemented a Convection mode:

If we pressed (key ==3) that mence we select Convection mode.It will set a Temperature of oven and working is like if the temperature greater than 250 then it will give warning message i.e. “Invalid temp.” and turn ON buzzer for specific time and after some time it will turn it OFF. Else if temp is less than 250 it will start pre-heating of food and basic timer for pre-heating is 180sec.



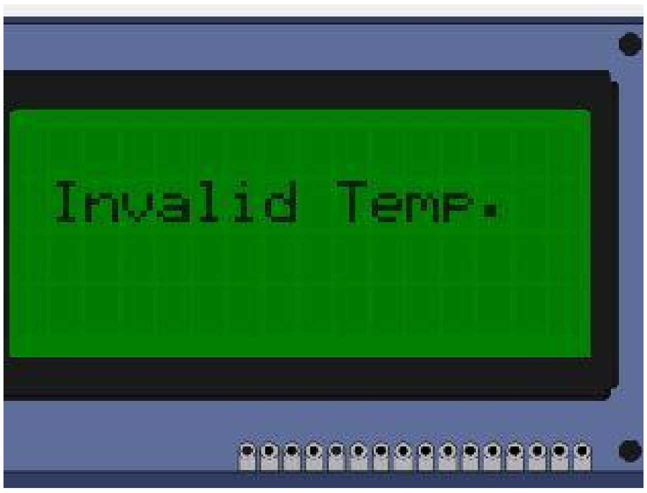
```
SET TEMP. ( °C )
 TEMP :

*:CLEAR   #:ENTER
```



```
  Pre-Heating

Time Rem.=177
```



```
 Invalid Temp.
```

● Start:

Let’s see how we implemented a Star mode:

If we pressed (key == 3) that mence we select Star mode.When the system is in Cooking mode display in start mode time it will automatically start timer from 30sec.

- Bugs Fixing:

When testing and checking all the processes one by one i found out some of the bugs and i have fixed all bugs by adding some extra lines of code.

1] In micro mode blinking bug is there like some unwanted cursor appred

This bug is appred into Set time mode section when excepting the min value from user. So i have used 'else if' insted of 'if'

2]blinking of min:sec after pressing '*'

So the bug was when we clear the min. Then sec should have to blink but it was not happing there that's why i have added that blink_pos = 0; for sec and blink_pos = 1; for min and the problem is solved succefully.

3]Pause mode issue key = = 5

This issue I have seen in micro mode that if I pressed key 5 it will Pause the timer but FAN still ON that's why I have to give command that FAN =OFF; then it work fine

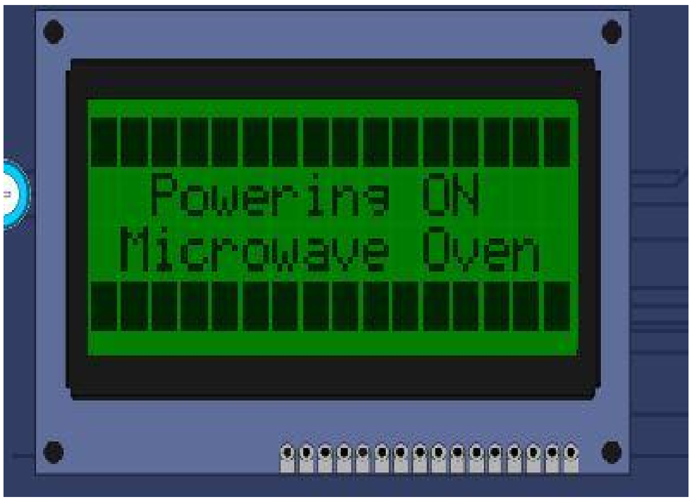
4]Door status

1st it will still doing the cooking processe if door is open so I have give the condition that id Door== Open give alert msg and turn buzzer on.

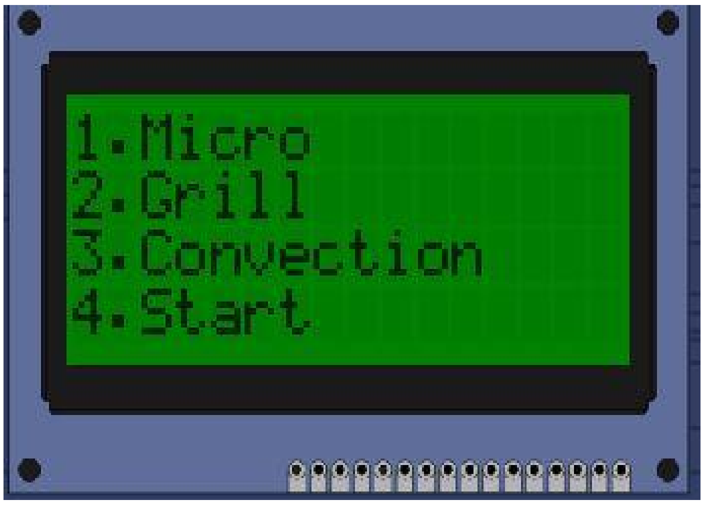
Like this I have solved all bugs I get overthere.

4. Testing

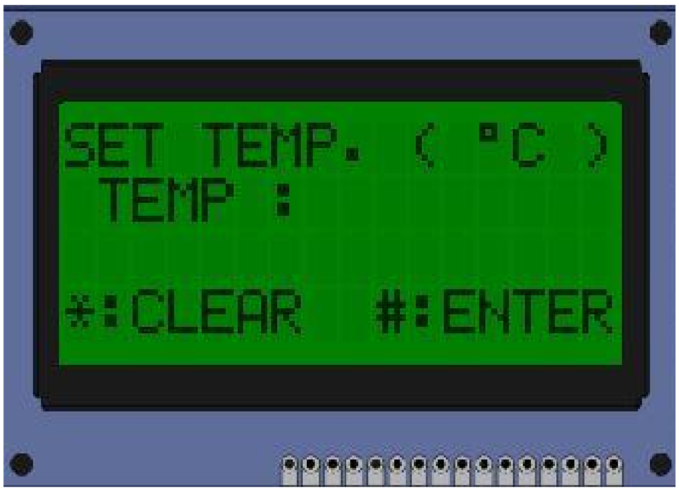
- **Result:**



```
████████████████
  Powering ON
 Microwave Oven
████████████████
```

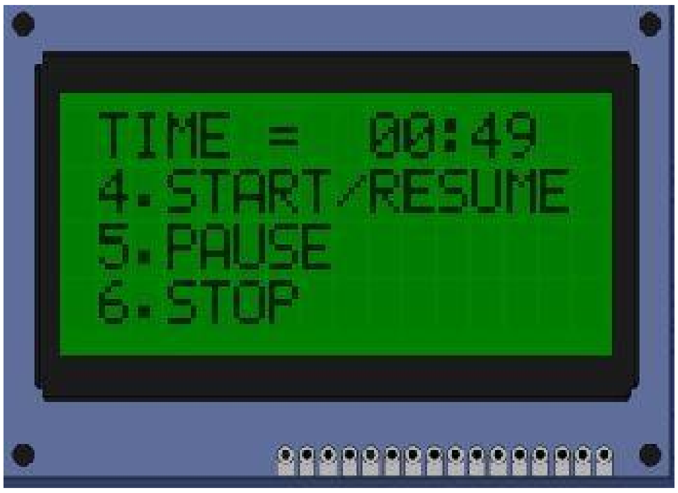


```
1.Micro
2.Grill
3.Convection
4.Start
```

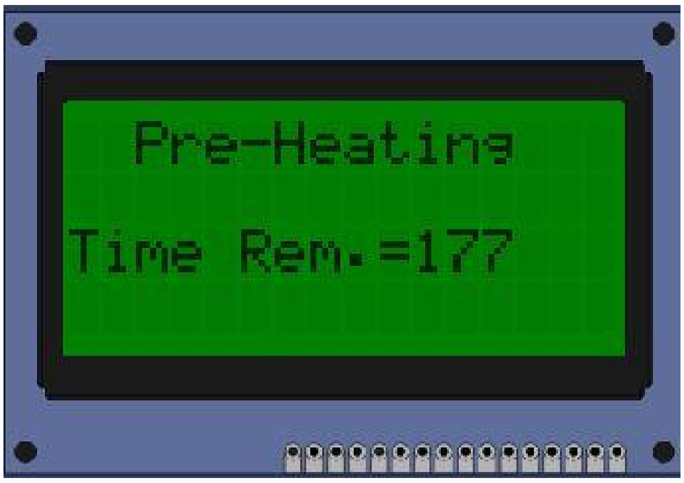


```
SET TEMP. ( °C )
 TEMP :

*:CLEAR   #:ENTER
```

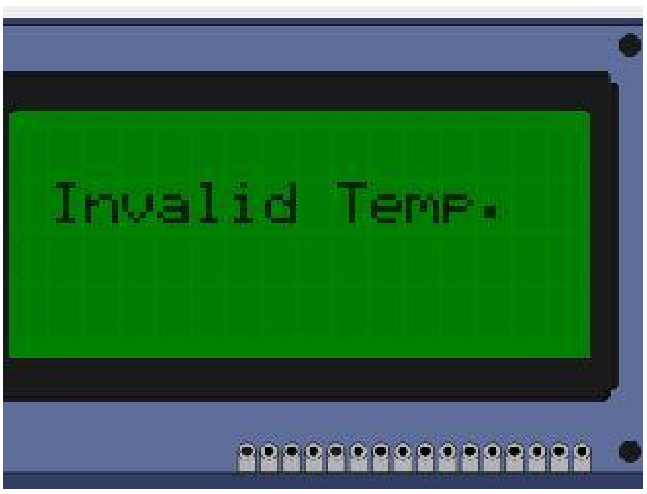


```
 TIME =   00:49
 4.START/RESUME
 5.PAUSE
 6.STOP
```



```
  Pre-Heating

Time Rem.=177
```



```
 Invalid Temp.
```

By observing this snapshots we realise that the idea we want to implement is successfully done