PIC16F/18F Mini Development Board

An alternative to Arduino Uno, for Studying Embedded Systems hands-on

Reference Manual

AMOTECH LABS

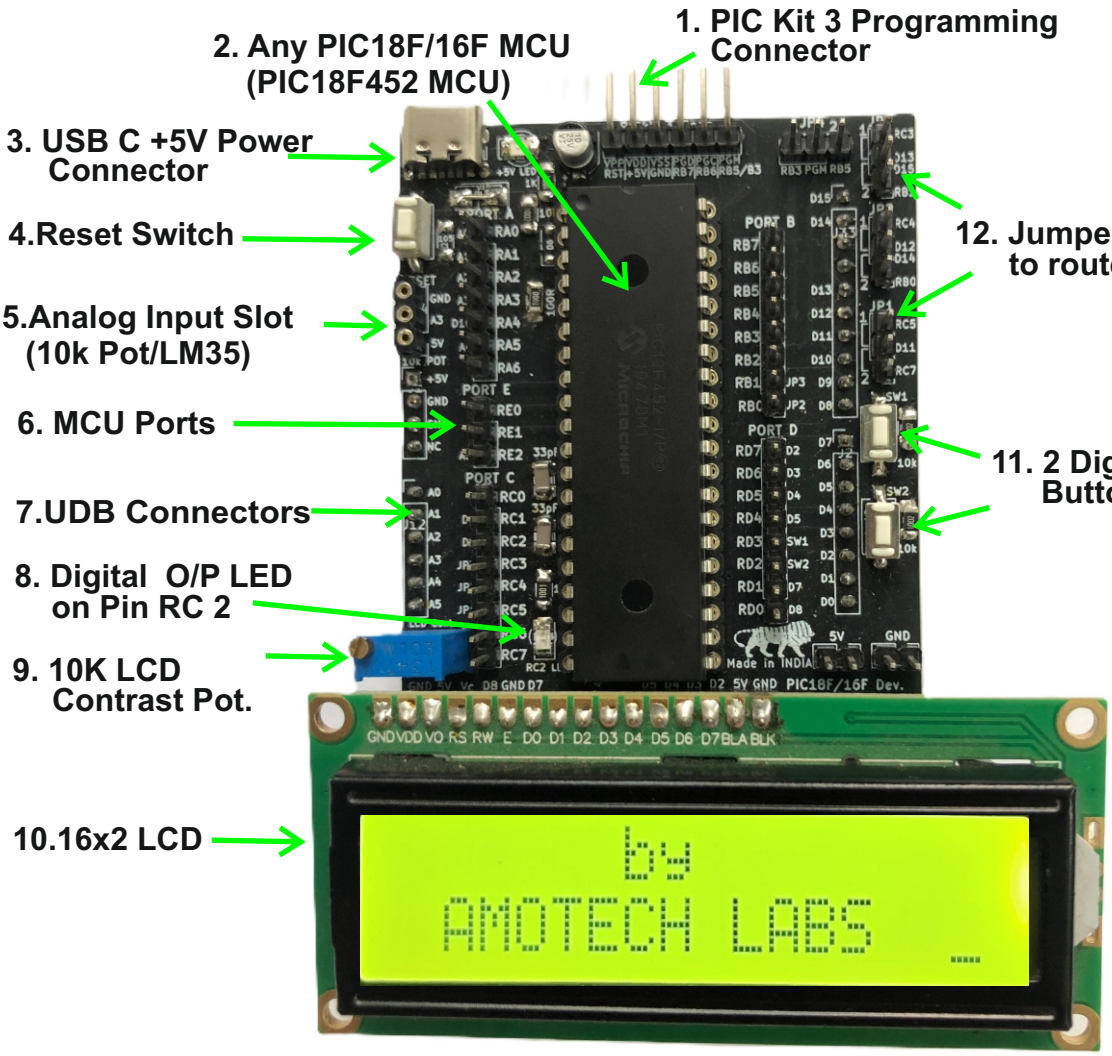
Embedded systems | Industrial Automation | Robotics

Scan to download Soft Copy of Manual & Online Purchase

Scan for Video Tutorials

Contents

Overview of PIC18F/16F Mini Development Board



Most of above labels are self explanatory. Below is the description needed for some of the labels. Also check brief Introduction to PIC18F452 & Schematic of the Kit in following pages for detail understanding..

1. PIC Kit 3 Prog. Conn.: It is connected to PIC Kit 3/3.5 programmer to program the MCU. RST pin of the kit is connected to a Programmers RST pin, indicated by a notch symbol.

2. PIC16F/18F MCU: This kit can be used with multiple 40Pin DIP MCU's from PIC like PIC16F887A/887, PIC18F4550/4580/4520 etc and all those which are pin compatible with them.

3. USB C +5V Power Conn.: This Kit can be powered by USB C cable using any +5V power Adapter/USB.

4. Reset Switch: Resets the MCU to restart the program execution. Press Reset after program download.

5. Analog Input Slot: This slot can be used to insert Lm35 or 10K Pot to feed variable Analog voltage to A3.

6. MCU Ports.: Port pins of Port A,B,C,D & E of PIC MCU can accessed with these connectors. Label of each pin is given on the right side of each pin and left side labels are of UDB connectors.

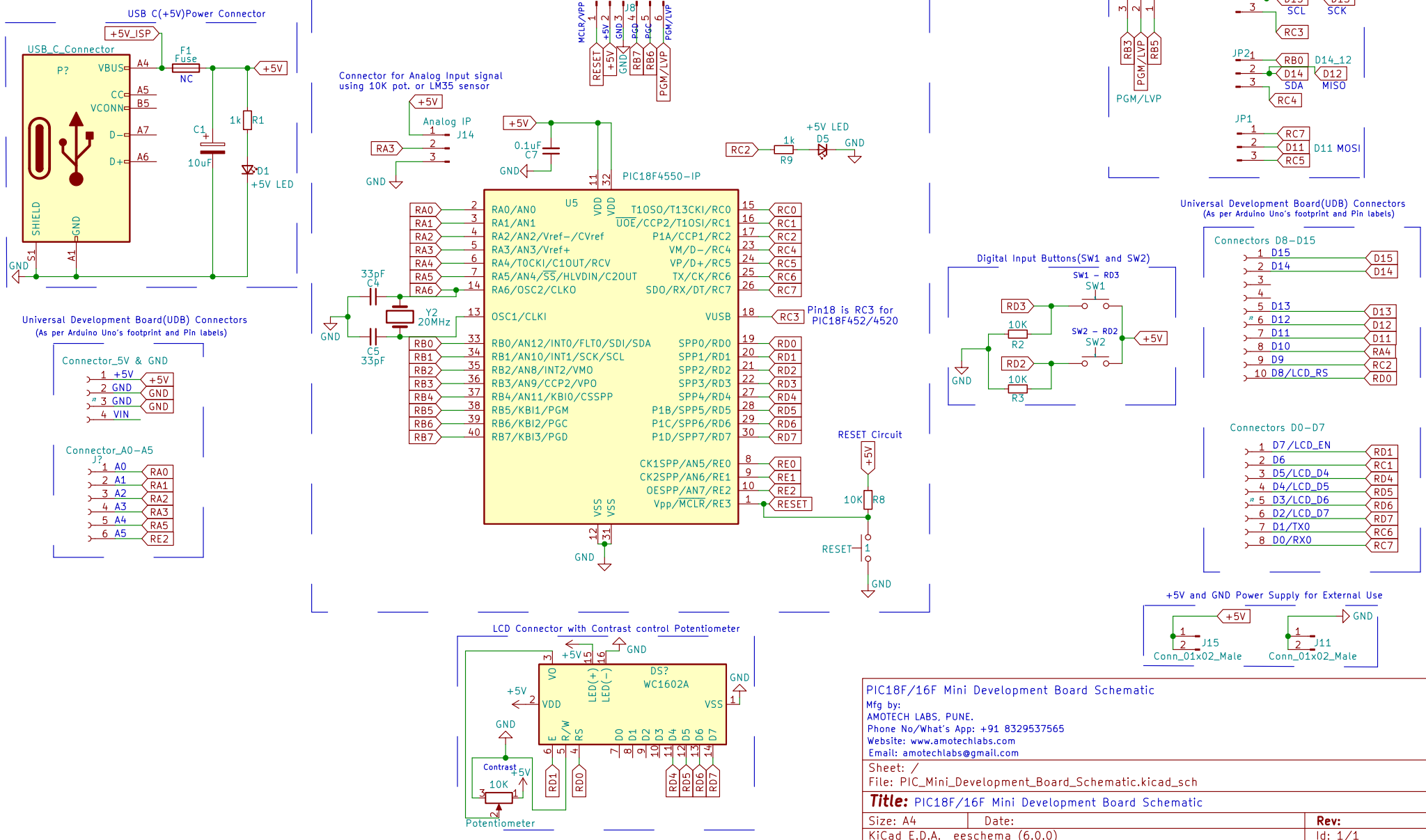
7. UDB Conn.: These connectors are used to attach this kit to **'Universal Development Board(UDB)'** of Amotech Labs. Check UDB details on our website.

8. Digital O/P LED on Pin RC2: The pin RC2 is an PWM enable pin on almost all PIC16F/18F MCUs. So, an LED is connected on RC 2 pin to test Output on it.

11. 2 Digital Input Buttons: These 2 Input Buttons are connected to RD 3 and RD 2 pins of MCU.

12. Jumper Sets JP 1 to JP 4 : These jumpers are used to route SPI and I2C pins of different PIC18F/16F MCU's to UDB's SPI & I2C pins alloted from D10 to D15.

PIC18F/16F Mini Development Board Schematic



PIC18F/16F Mini Development Board Schematic

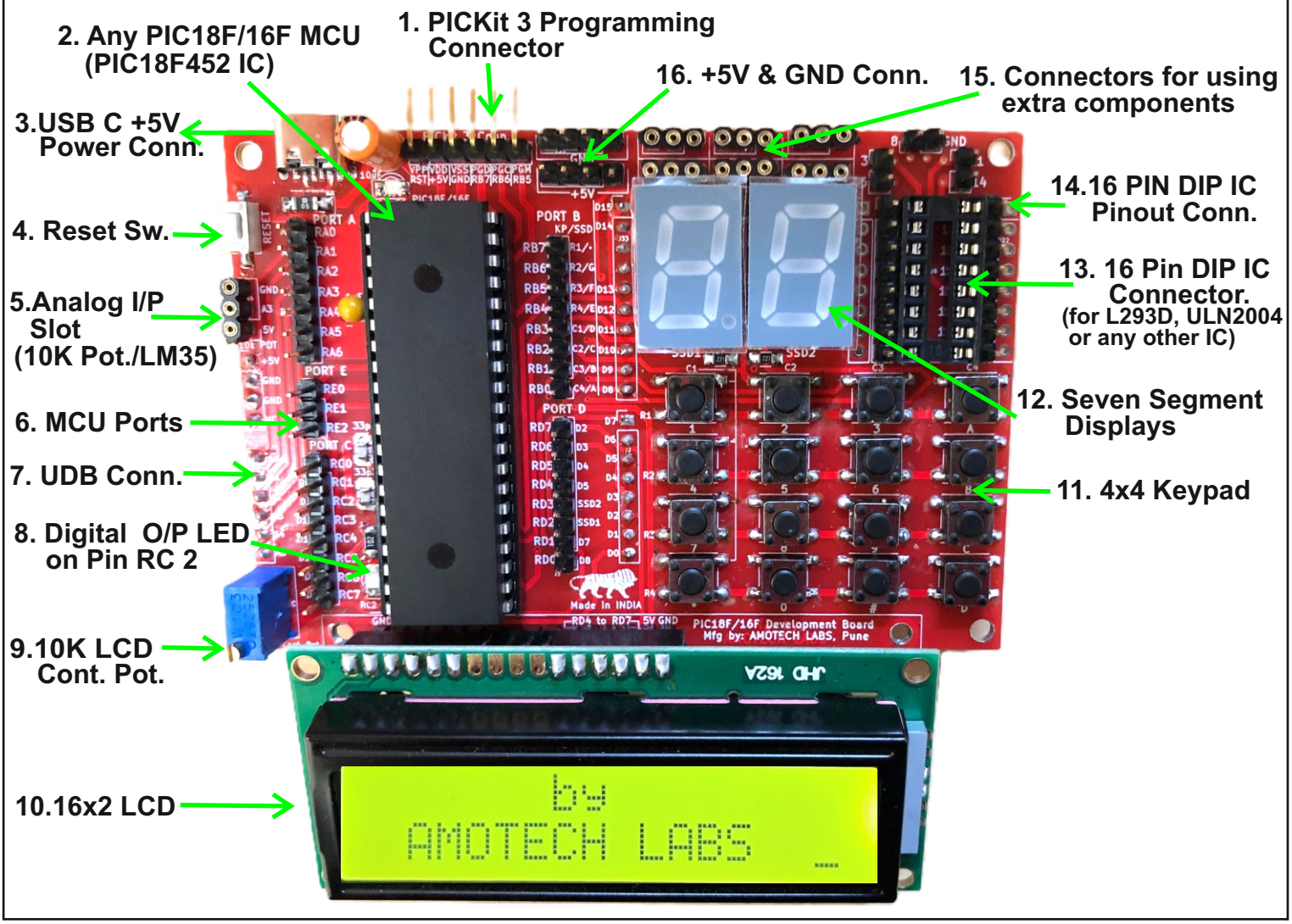
Mfg by:
AMOTECH LABS, PUNE.
Phone No/What's App: +91 8329537565
Website: www.amotechlabs.com
Email: amotechlabs@gmail.com

Sheet: /
File: PIC_Mini_Development_Board_Schematic.kicad_sch

Title: PIC18F/16F Mini Development Board Schematic

Size: A4	Date:	**Rev:**
KiCad E.D.A. eeschema (6.0.0)		Id: 1/1

Overview of PIC18F/16F Development Board Kit:



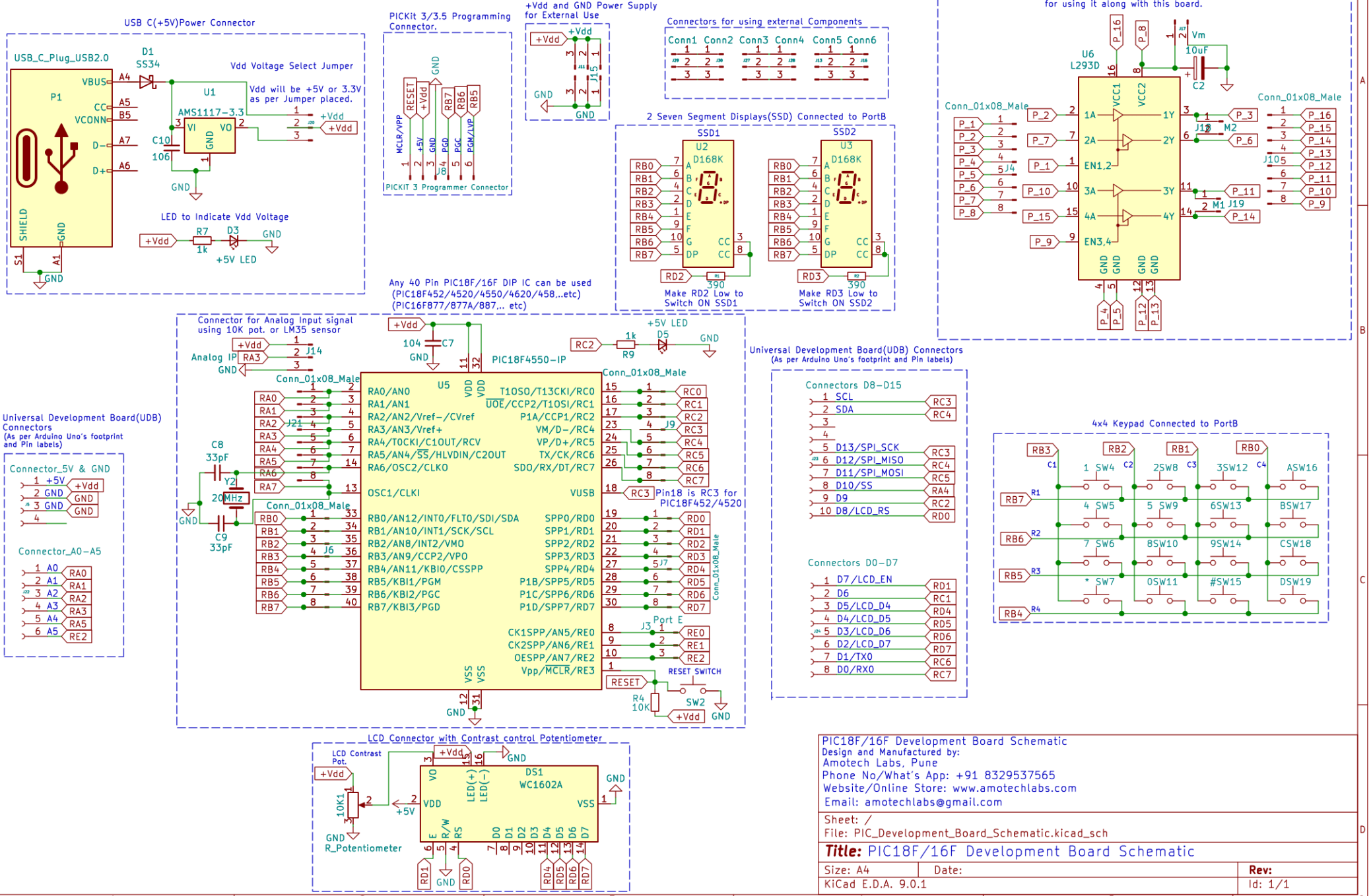
Above Kit is similar to the 'PIC18F/16F Mini Development Board', it just has additional features as seen, a on-board 4x4 Keypad,2 Seven Segment Displays, a 16 Pin DIP IC slot for using L293D, ULN2004 and any other IC within a package 8 and 16 pin DIP IC's. Additionally, it also has some empty bread board like connectors which can be used to form a circuit using external components.

11. 4x4 Keypad: The on-board 4x4 Keypad is connected to Port B of the PIC MCU as seen in the Schematic.

12. 2 Seven Segment Displays: There are 2 Seven Segment Displays on-board, **SSD1** and **SSD2** which are connected to the **Port B** of the PIC MCU. The common Cathode's of **SSD1** & **SSD2** are connected to the pins **RD2** and **RD3**. When **RD2** is low then SSD1 will display number as per Port B pin states, and same implies with **SSD2** & **RD3.** By using **PortB** and switching **RD2** & **RD3,** we can display 2 digits numbers on **SSD1** & **SSD2**.

13. 16 Pin DIP IC Connector: The 16-Pin DIP connector on board, can be used to insert and use any 8 or 16 Pin DIP IC's like **L293D** Motor Driver, **ULN2003** or any **8 pin Op-amp**. All pinouts are accessible and can be used to connect with PIC MCU on the board. If used for L293D Motor Driver, there are are also separate, 2 pin connectors for connecting 2 DC Motors and giving external DC voltage for DC Motors.

PIC18F/16F Development Board Schematic



PIC18F/16F Development Board Schematic
Design and Manufactured by:
Amotech Labs, Pune
Phone No/What's App: +91 8329537565
Website/Online Store: www.amotechlabs.com
Email: amotechlabs@gmail.com

Sheet: /
File: PIC_Development_Board_Schematic.kicad_sch

Title: PIC18F/16F Development Board Schematic

Size: A4	Date:	**Rev:**
KiCad E.D.A. 9.0.1		Id: 1/1

AMOTECH LABS

'PIC18F/16F Mini Development Board' can be mounted on below UDB Kit:

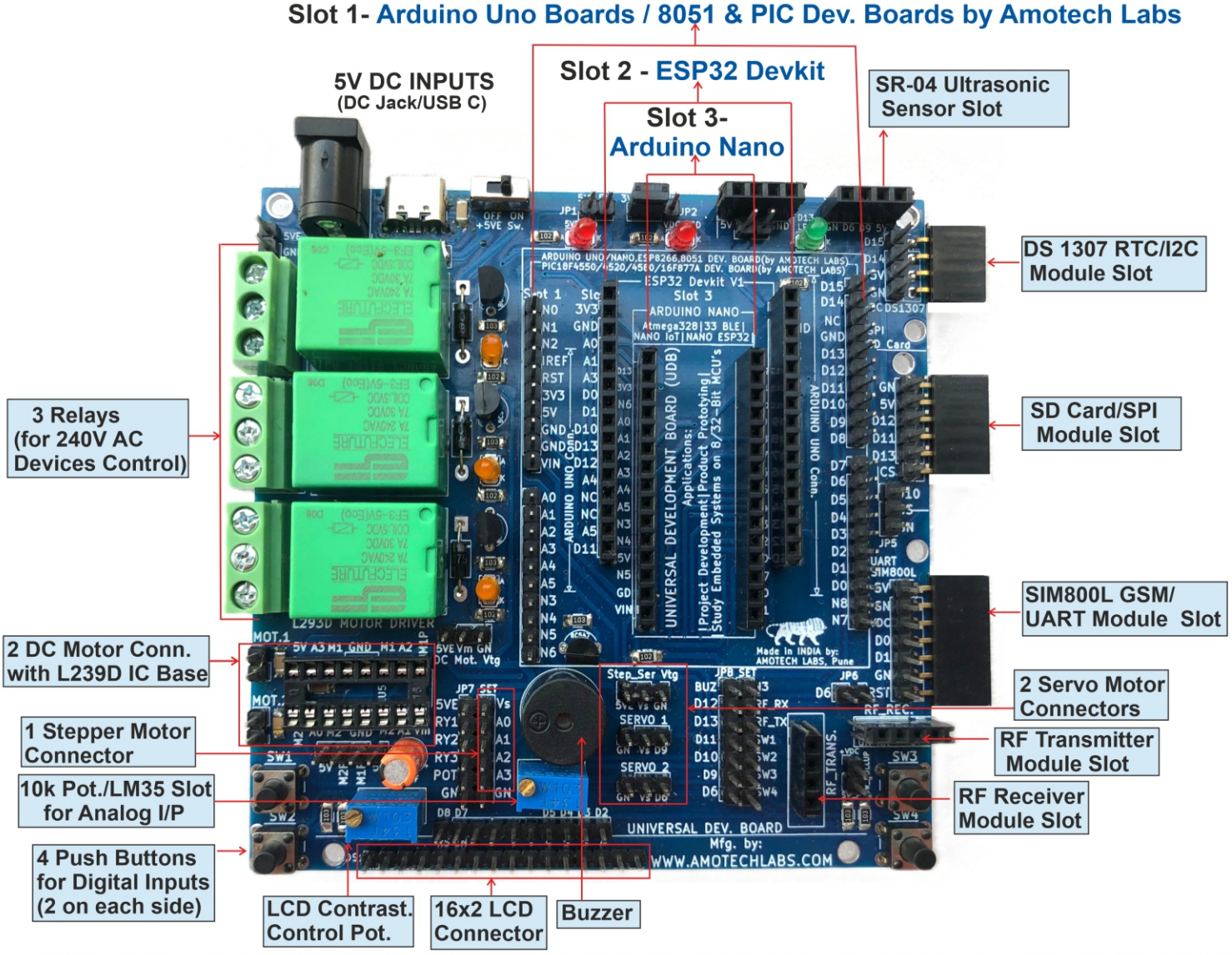
Universal Development Board(UDB) kit

One Development for

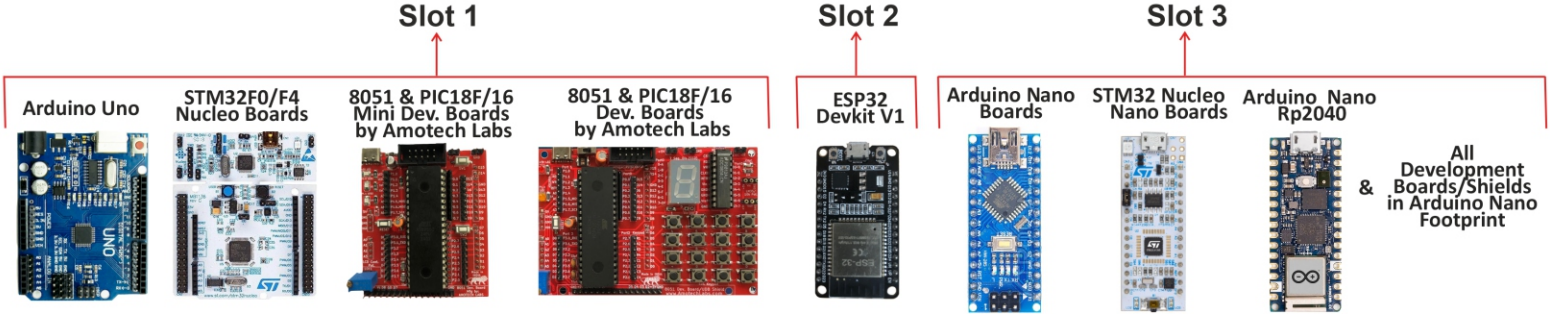
Studying Embedded Systems | Project Development | Product Prototyping

on below multiple 8 & 32-Bit Microcontrollers

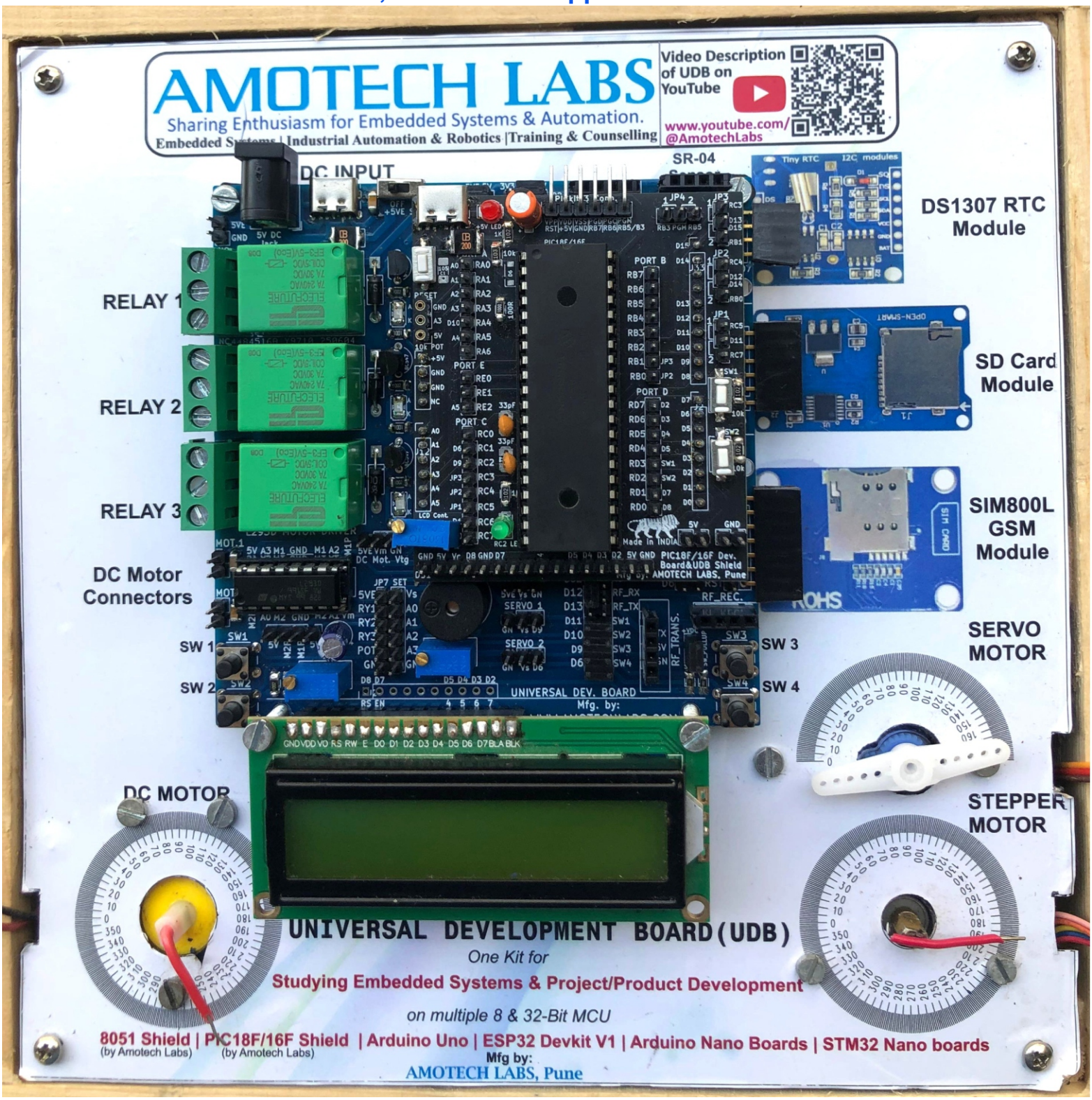
8051 Shield & PIC18F/16F Dev. Boards | Arduino Uno | ESP32 Devkit V1 | Arduino/STM32 Nano Boards
(Made by Amotech Labs)



Below Micro-controller Shields/Boards can be inserted into UDB and interfaced with all above Peripherals on it

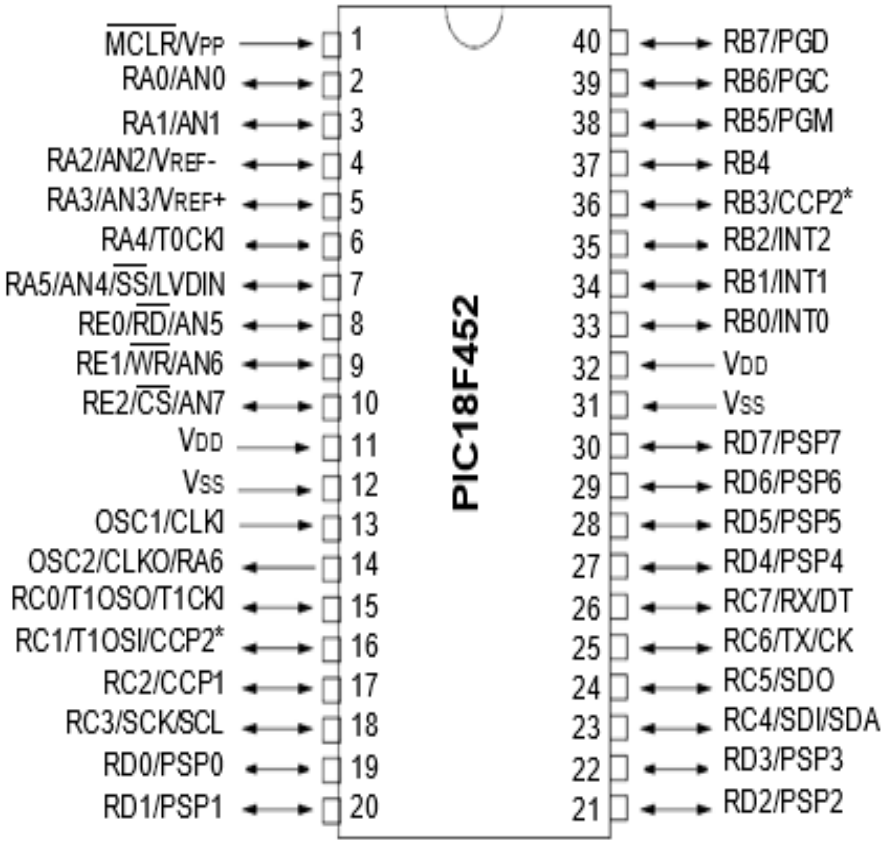


'PIC18F/16F Mini Development Board' mounted on UDB Kit's Slot 1 can be seen in the below UDB Kit which has DC, Servo and Stepper Motors mounted on-board.



Note:This Kit was assembled for College Lab use in Wooden Box Enclosure. That's why it is fitted on a Acrylic Sheet with Motors mounted on it for studying their interfacing using PIC18F MCU & UDB. For Students, only UDB is also available to buy with 'UDB 2 Wheel Robot Acrylic Sheet'. Students can configure the UDB by adding the peripherals as per their application needs.

Introduction to PIC18F452 Microcontroller



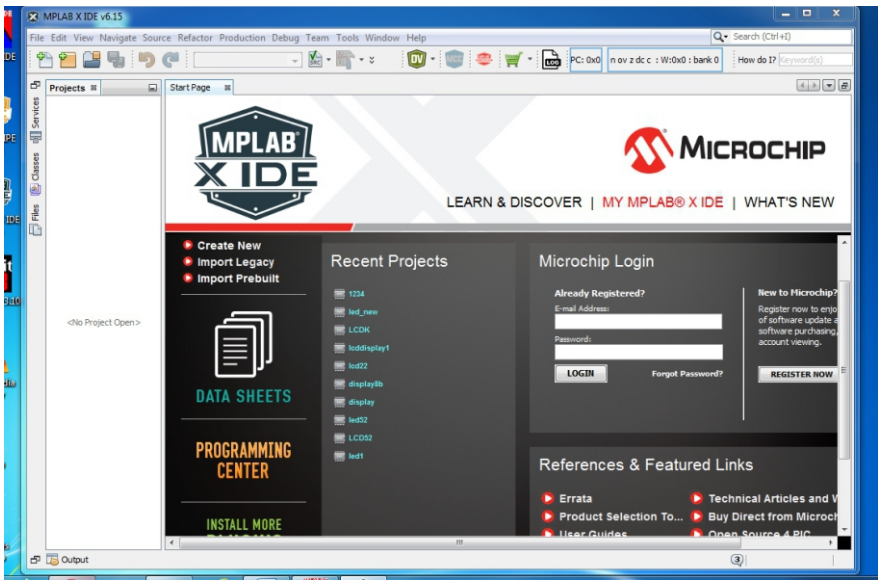
Features of PIC18f452:

- ✓ Operating Frequency : DC - 40 Mhz
- ✓ Program Memory (Bytes): 32K
- ✓ Program Memory (Instructions) : 16384
- ✓ Data Memory (Bytes): 1536
- ✓ Data EEPROM Memory (Bytes) : 256
- ✓ Interrupt Sources 18
- ✓ I/O Ports : Ports A, B, C, D, E
- ✓ Timers : 4
- ✓ Capture/Compare/PWM Modules : 2
- ✓ Enhanced Capture/ Compare/PWM Modules : 2
- ✓ Serial Communications : MSSP, Addressable USART
- ✓ Universal Serial Bus (USB) Module : No
- ✓ Streaming Parallel Port (SPP) : Yes
- ✓ 10-Bit Analog-to-Digital Module : 8 Input Channels
- ✓ Comparators : 2
- ✓ Programmable Low-Voltage Detect : Yes
- ✓ Instruction Set : 75 Instructions; 75 Instructions
- ✓ Packages : 40-pin DIP, 44-pin PLCC , 44-pin TQFP

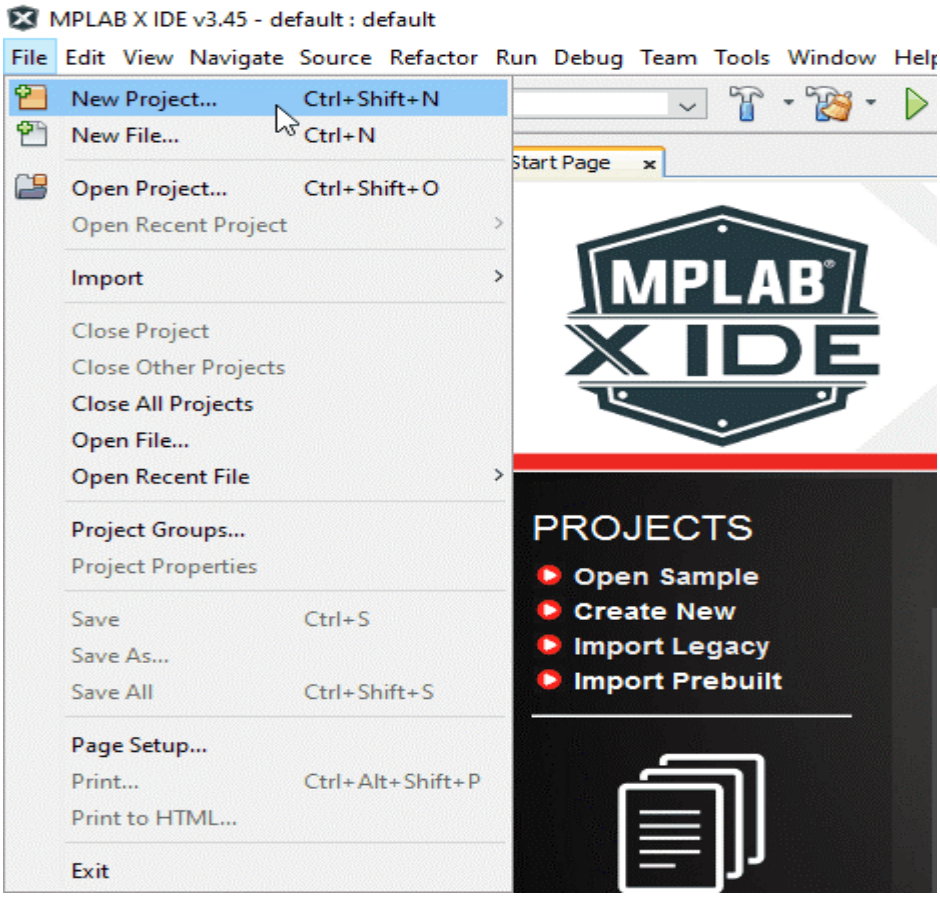
MPLAB X IDE for Programming PIC18F/16F MCU's

Below are the steps to program any PIC18F/16F MCU in MPLAB X IDE.

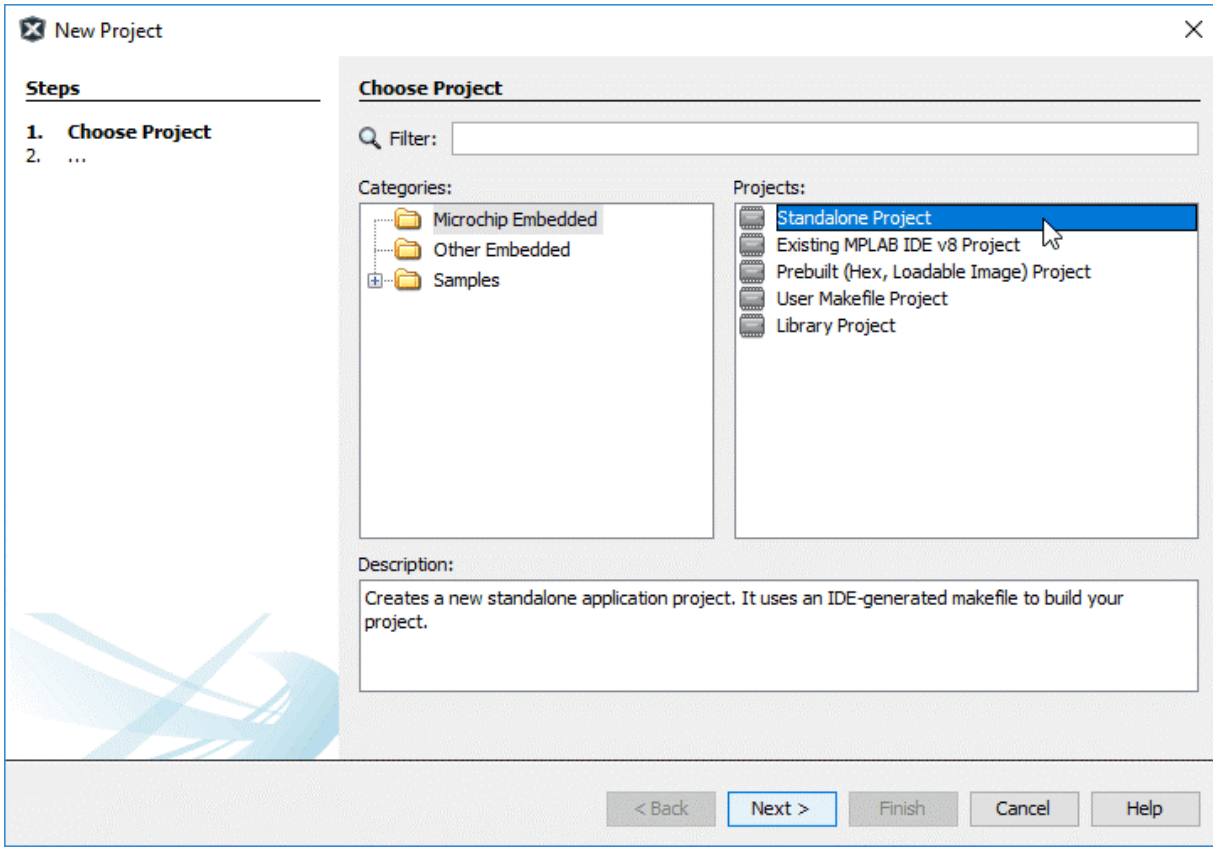
1. Double click on MPLAB X IDE icon on the desktop & Open MPLAB X IDE.



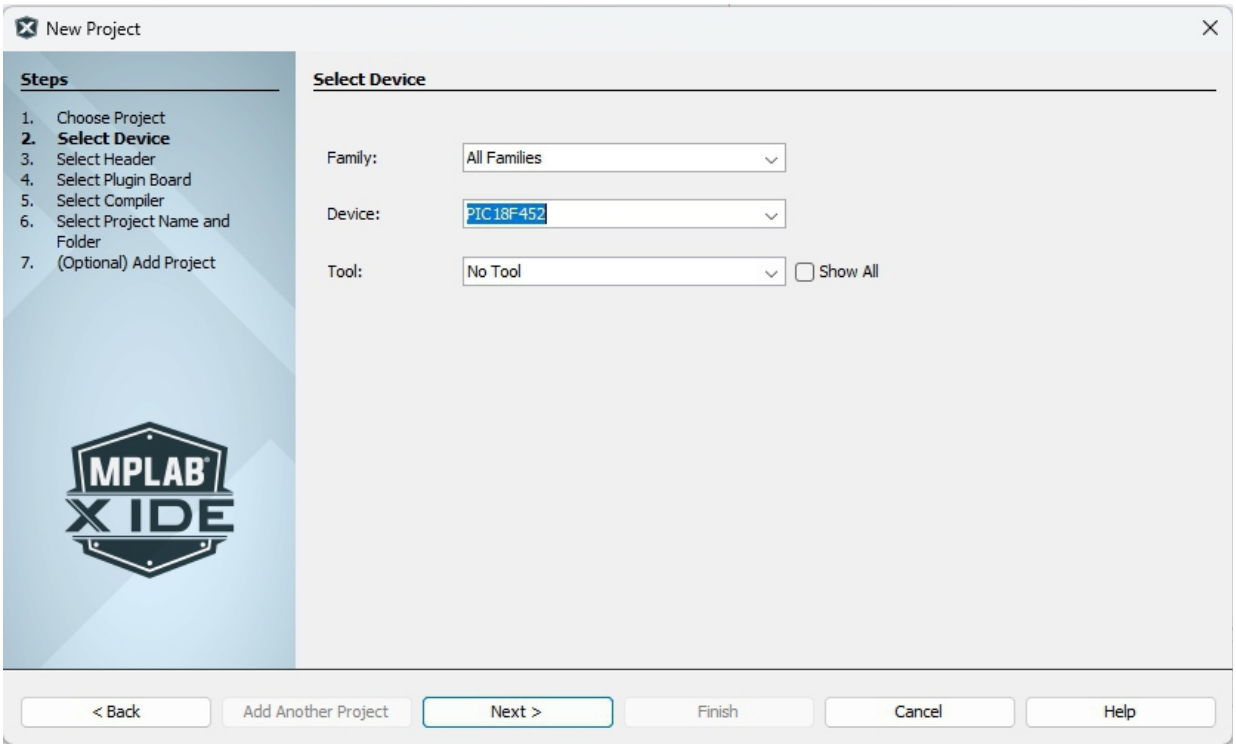
2. To create new project. Select File -> New Project.



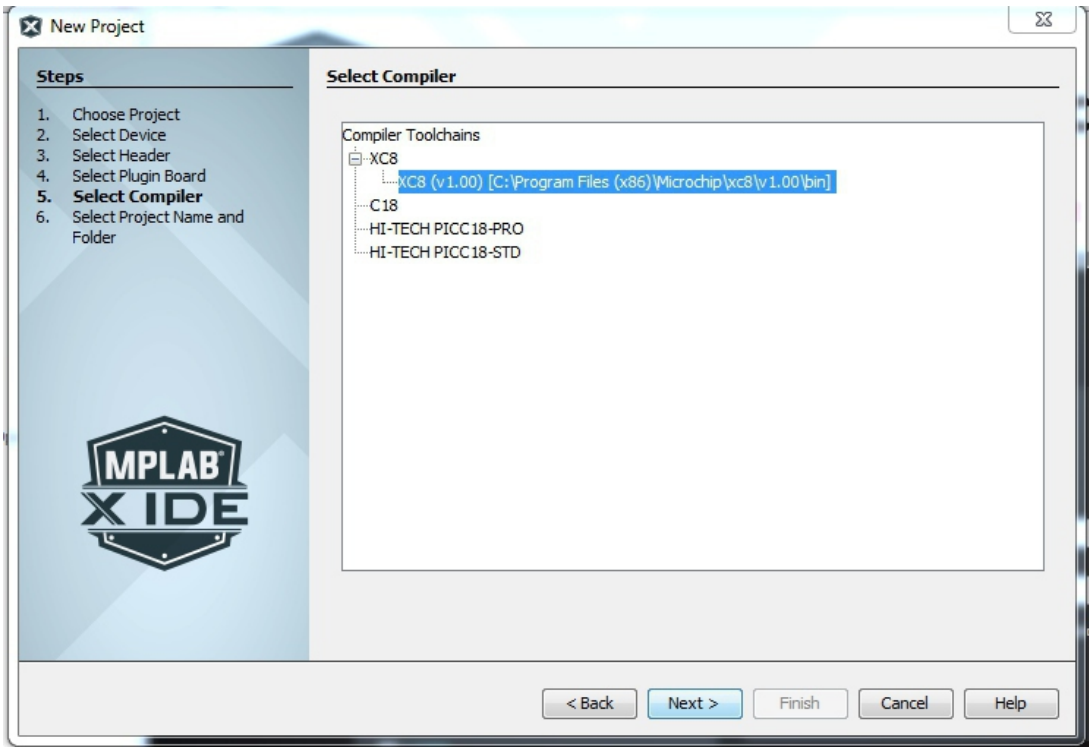
3. After that below window will appear. Select Microchip Embedded ->Standalone Project & Click next.



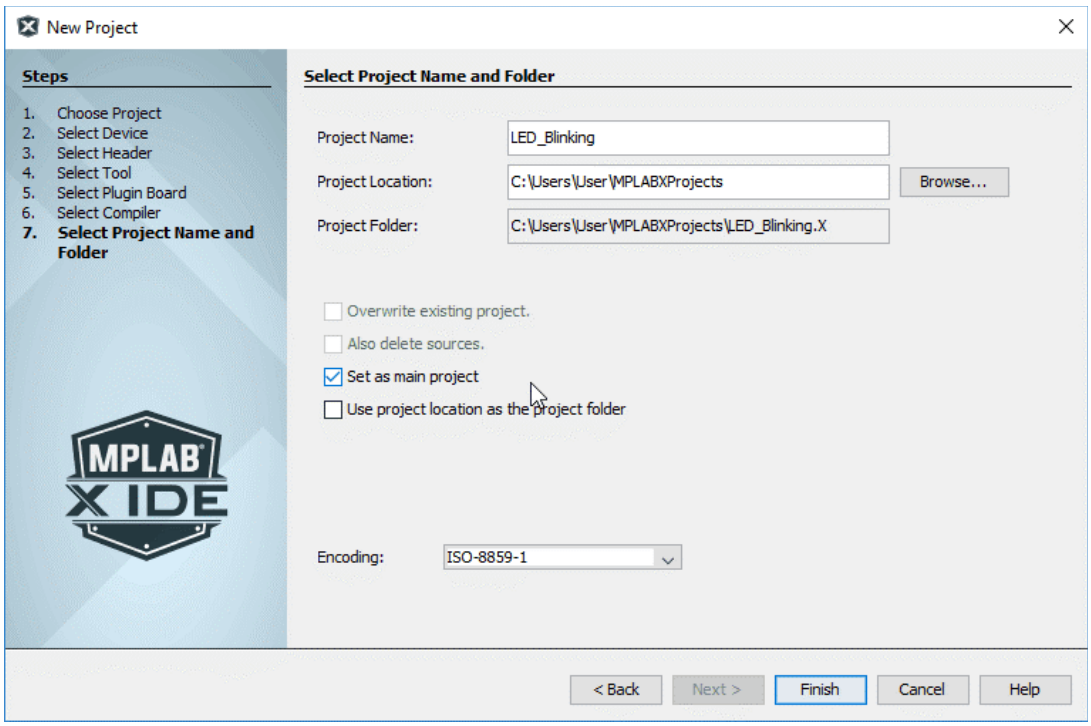
4. Select Family >Advanced 8-bit MCUs(PIC18) then select Device>PIC18F452 or any other MCU. In Tool you can select No Tool.



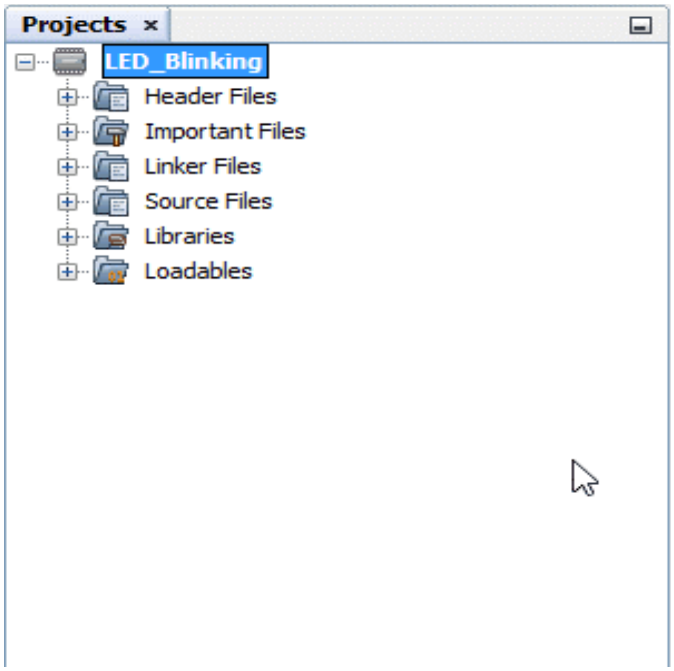
5. Now select the previously installed latest version of XC8 compiler and then click Next.



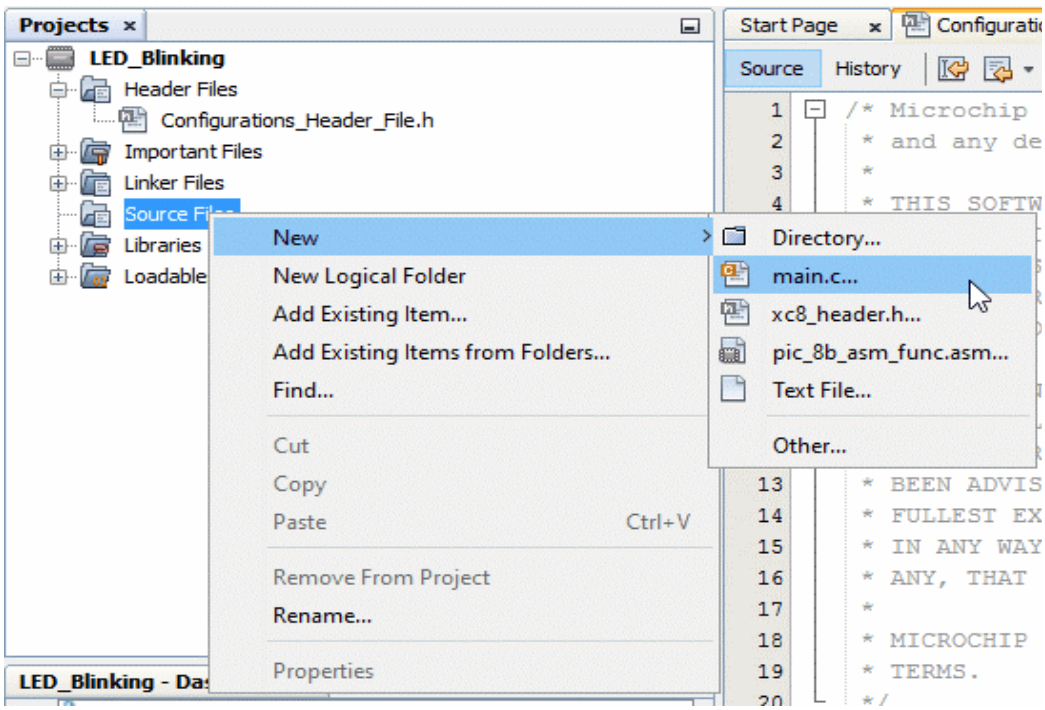
6. Enter your Project name > Click on Browse and Choose Project Folder location > Click on Finish.



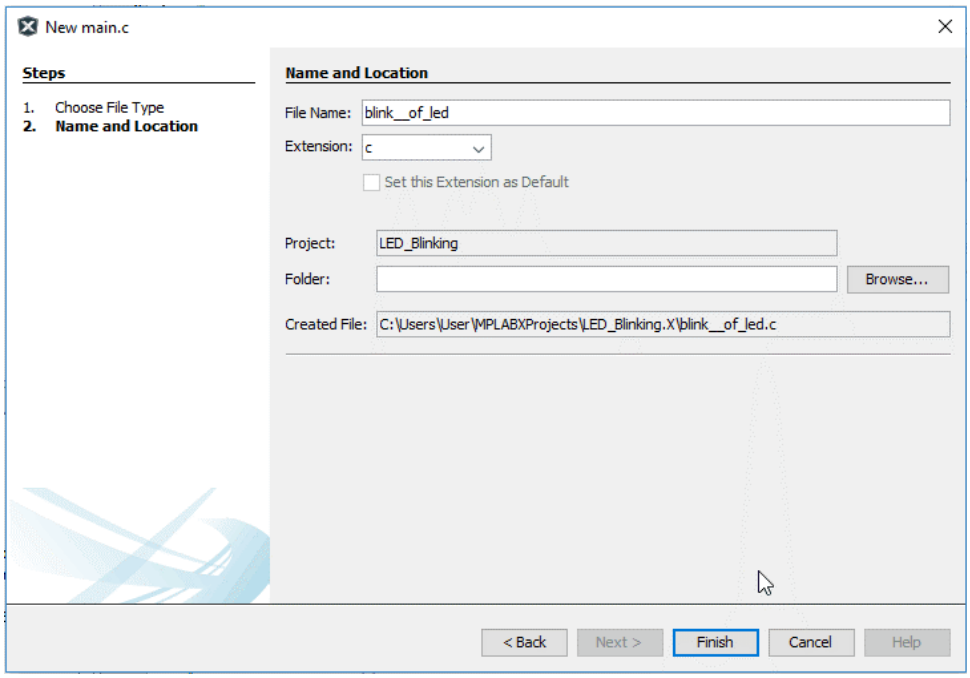
7. Now our project is displayed in Project window.



8. Create the Main file for developing code. Right Click on Source file in the project -> Select new -> Select main.c File.

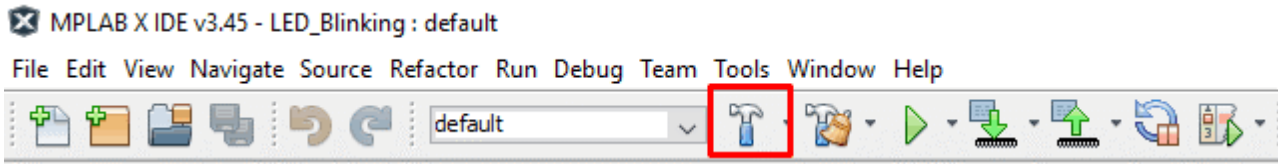


9. Give name to C file and click on finish.

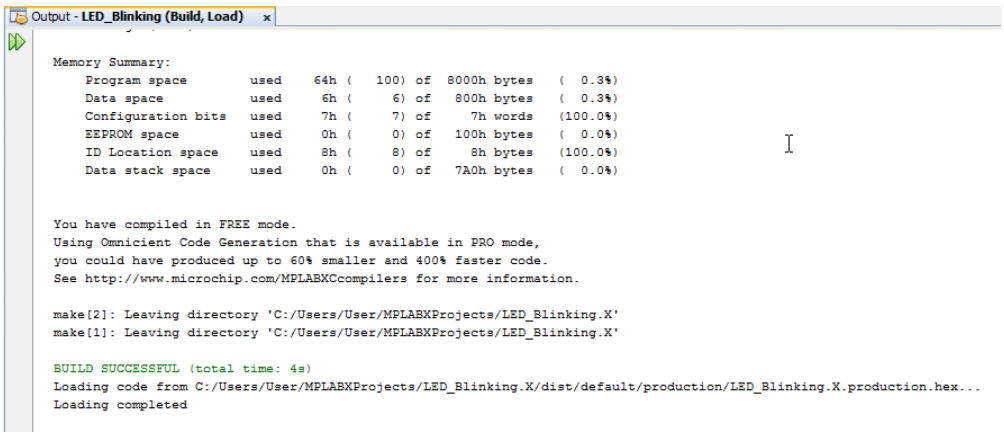


10. Write your program.

11. Afer you complete your program then Build it by clicking on a Hammer as seen below.



12. Afer building a Project, output Window appears which gives errors and warnings if any. otherwise Build Successful message appears as seen below.



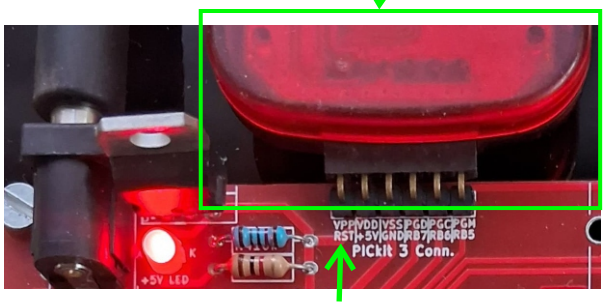
12. After BUILD SUCCESSFUL, Hex file is created in the project folder. In project folder Hex file is created in dist/default/production. You can also see a path to the folder under BUILD SUCCESSFUL.

MPLAB X IPE for Programming PIC18F/16F MCU's

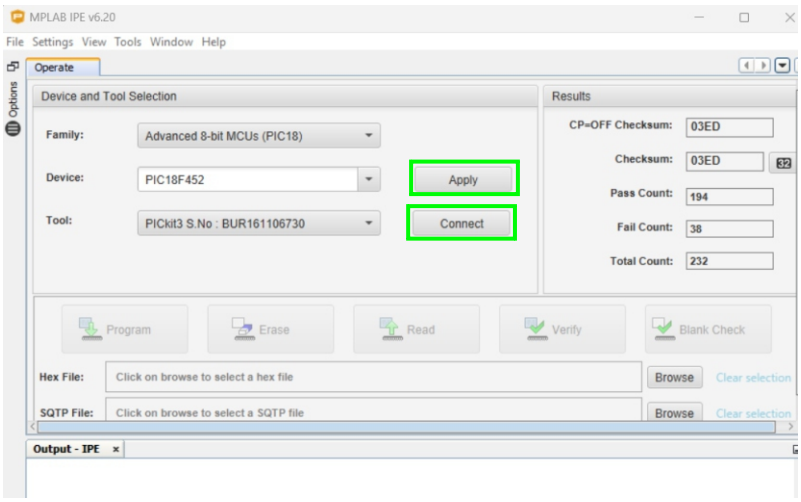
Below are the steps to program any PIC18F/16F MCU in MPLAB X IDE.

1.Open MPLAB IPE software. Select the Family and Device name of the MCU you want to program. Connect the USB of PICKit 3/3.5 Programmer to your PC, after connecting it the programmer name will appear in the tool. Now connect 12V power supply adapter to PIC18F/16F Kit and PICKit 3's 6 Pin connector to your PIC18F/16 kit directly as seen in the Kit Overview Pic on Page 3. Also, you can connect it using 6-pin F-F connector wires. PICKit 3's first pin indicated with arrow should be connected to RST pin of your PIC18F/16F kits 6-pin connector.
Now, click on **'Apply'** and then on **'Connect'**.

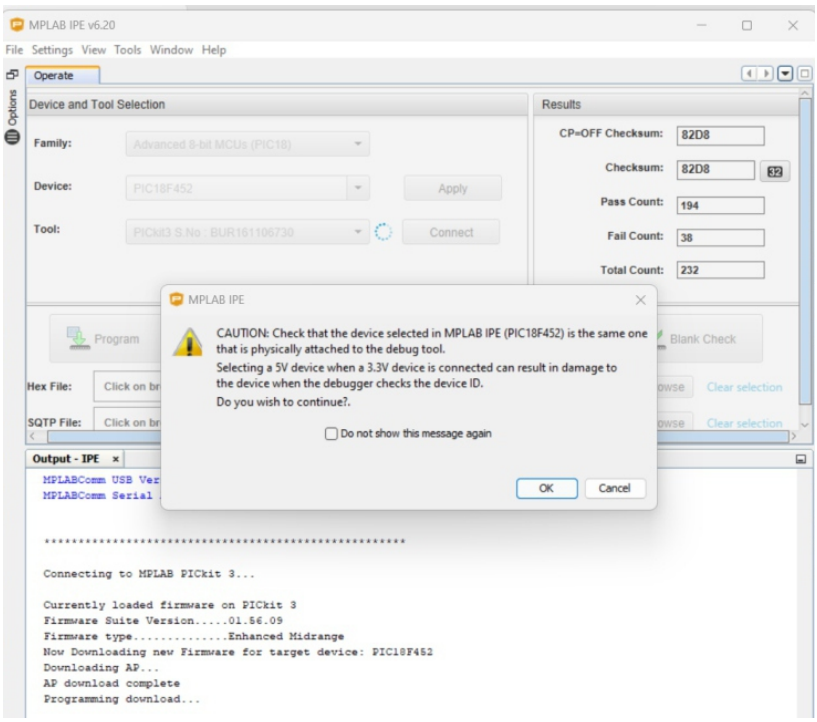
ICSP Conn & PICkit 3 Programmer.



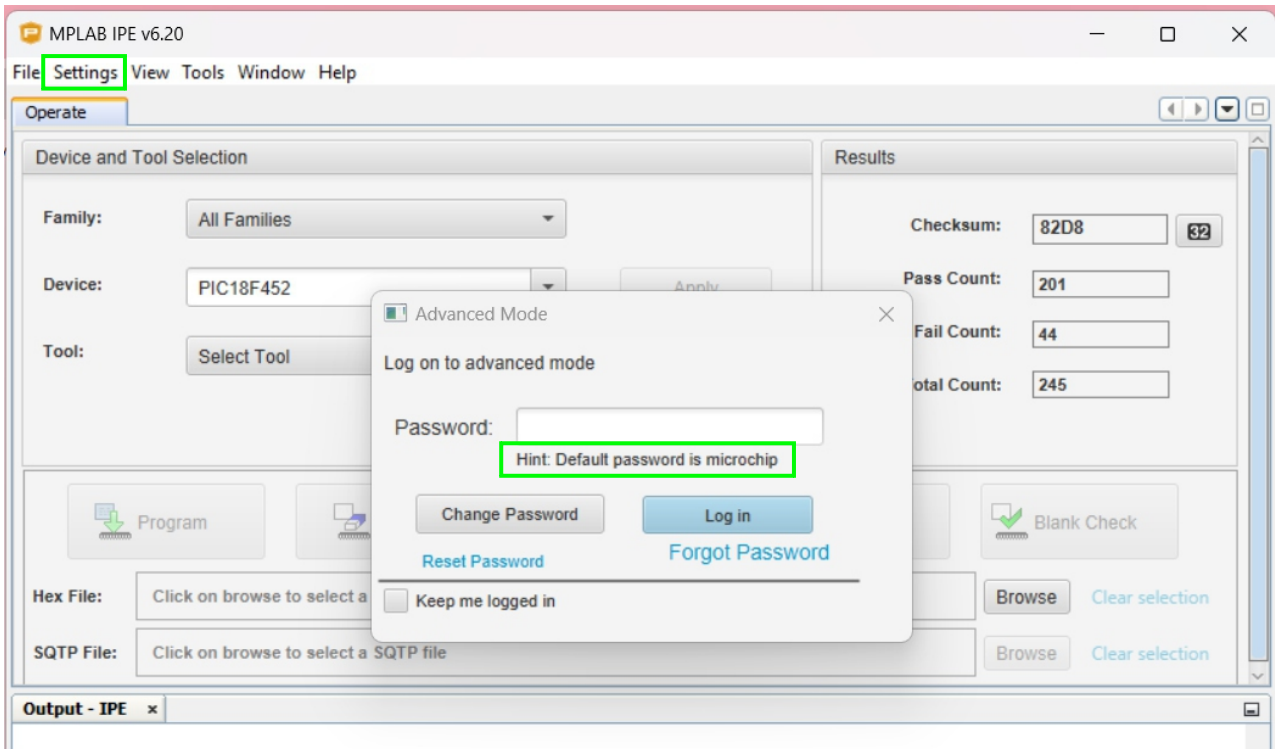
Reset Pin of ICSP Conn. on PIC18F/16F Kit



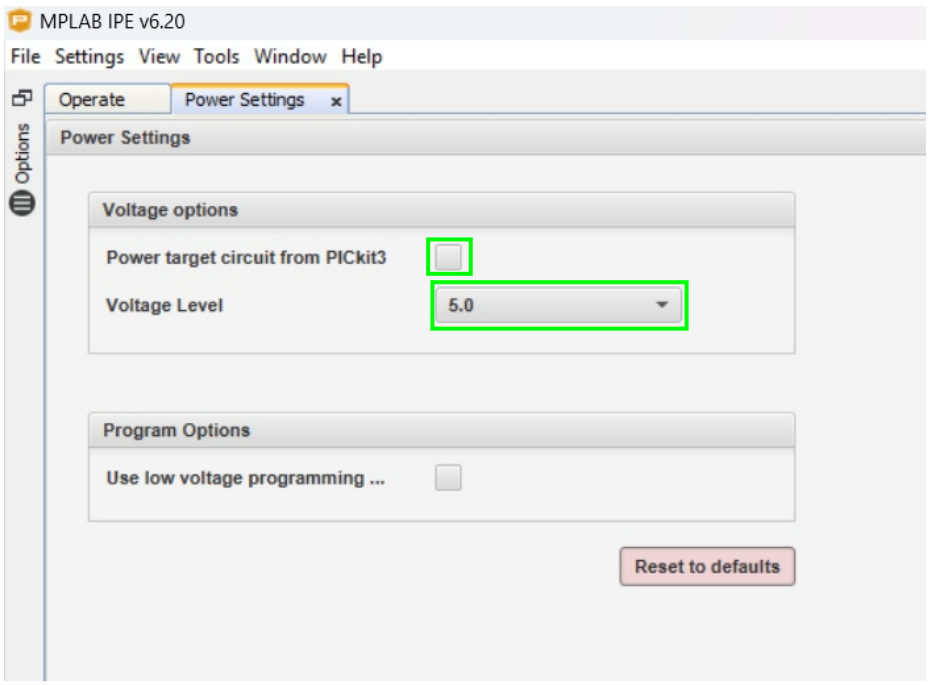
2. Once you click on Connect, below pop-up window may appear if you have not done Programming Power Source and Voltage level settings. Programming voltage and Power settings should be done before downloading the Program.



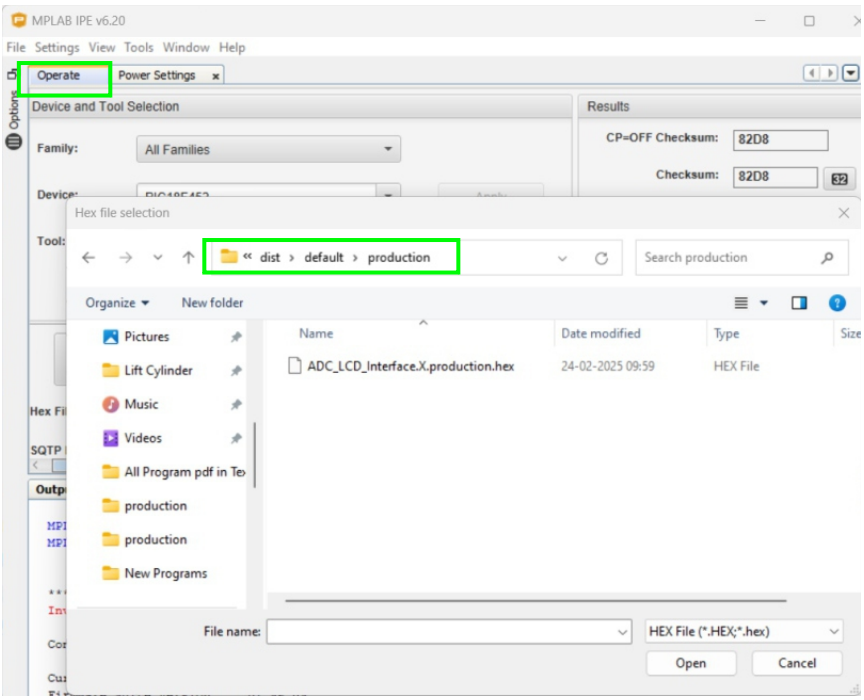
3. To confirm if correct Programming Voltage and Power Source is selected, we need to click on Setting and select Advance Mode. Then below Pop-Up window will appear. Enter the default password **'microchip'** as given in the Hint and press Log in. Now you go in Advanced mode.



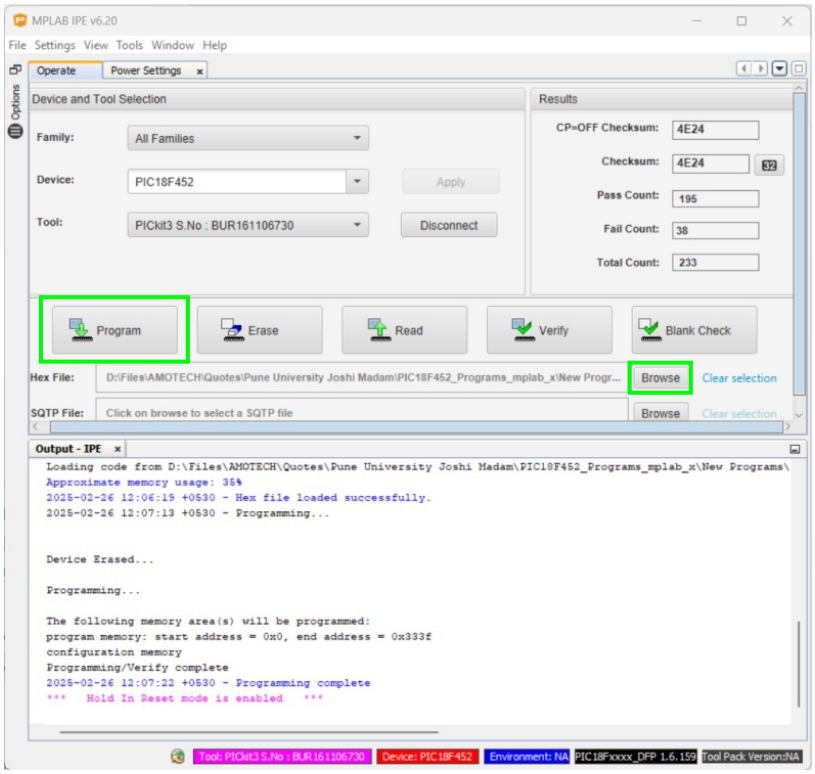
4. Once you are in Advance Mode, you will see the **'Options'** tab on the left. Click on it and select **'Power'** tab, then below window will appear. Un tick the below check box and select the Voltage level as **5.0** Volt. Now, must be powered by external Power Supply before downloading the program.



5. Once Voltage and Power Source is selected, click on **'Operate'** tab. Now, select you Hex file by by clicking on 'Browse' and navigate to your Hex file location. In your Projet Folder you need to navigate to dist>default>production and select your Hex File.

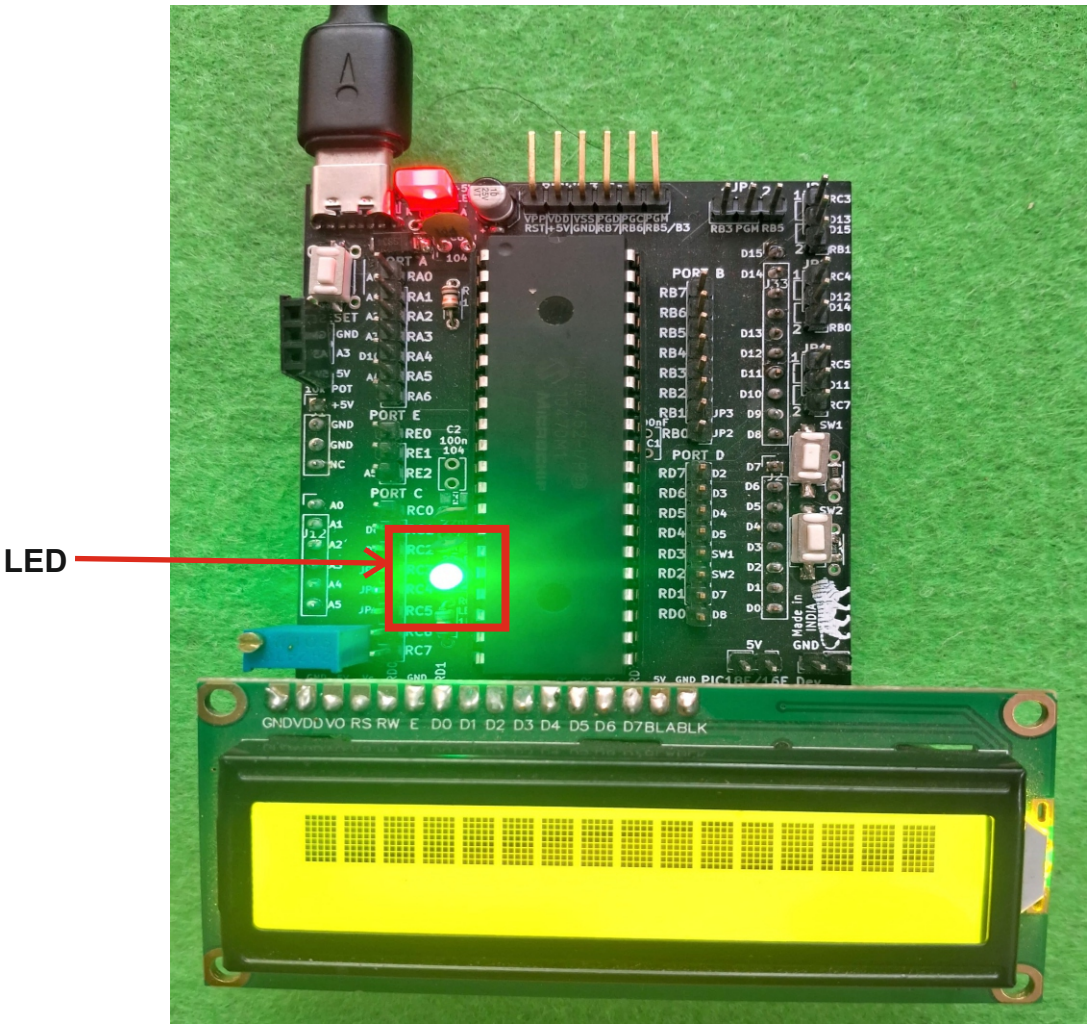


6. Once you select the Hex file you will see the notification Hex File Loaded Successfully in 'Output IPE'. Click on 'Program' button.You will see PICKit 3 LED's flashing and **'Programming...'** in Output-IPE. And, finally **'Programming complete'** once downloaded. Now remove PICKit 3 and see your program running on the Kit.



Lab 1 - LED Blinking

Aim: To study the LED's Interfacing with PIC18F452 mini development.



Procedure:

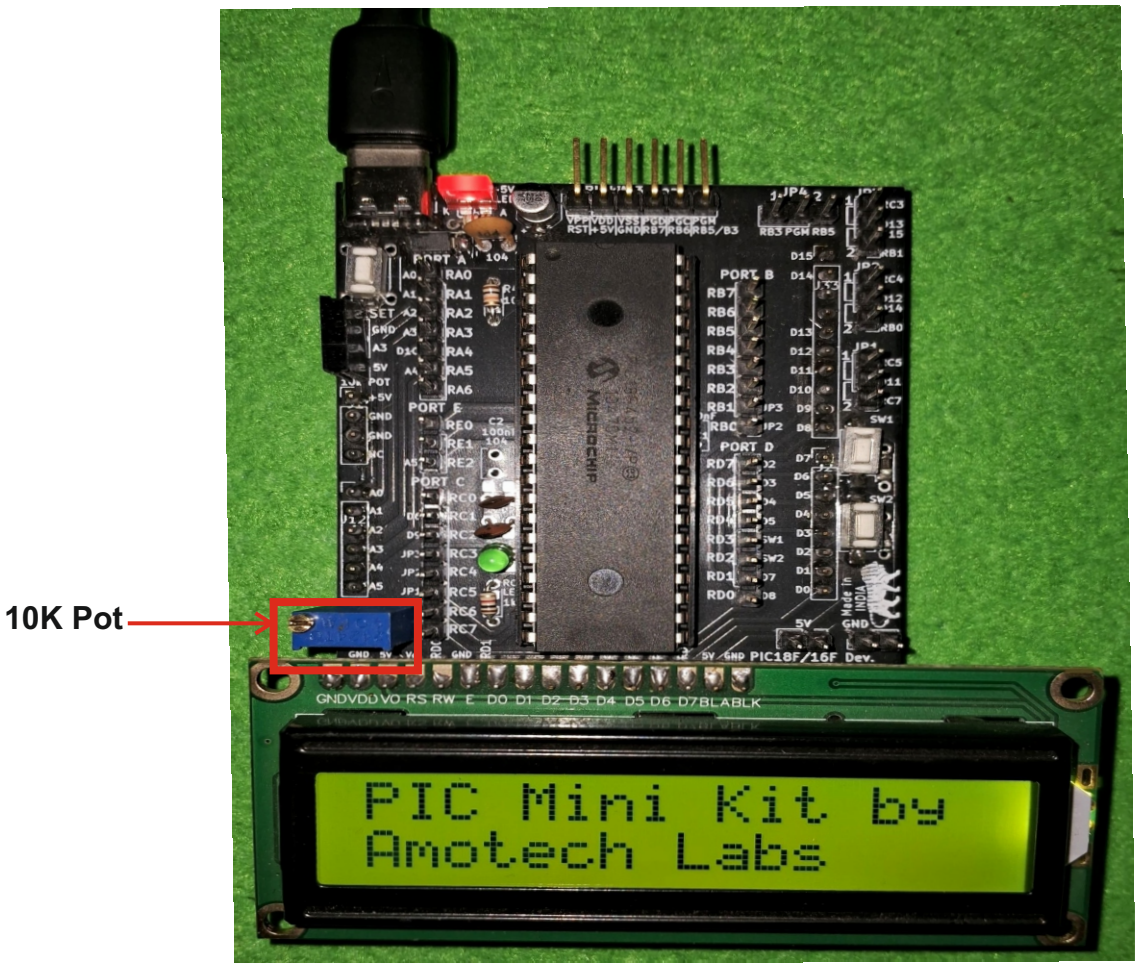
1. Read the LED Blink Program on next page with all the comments which explains the program.
2. Open the Project folder for the program in MPLAB X IDE or write it by making new project as per MPLAB X IDE procedure explained in the manual. Build and generate the Hex file for the same.
3. Open MPLAB IPE and follow the procedure explained to download the Hex file for the above program.
4. Use PICkit3 programmmer to program the PIC microcontroller with the HEX file.
5. Make sure that the arrow side of the programmer must be connected to the RST pin of the ICSP connector of the board.

Program:

```
1 #include<xc.h>
2 #define_XTAL_FREQ  20000000    // Define the crystal frequency (8 MHz)
3 #include<pic18f452.h>         // Header file to include PIC18F452 MCU configuration
4 #pragma config OSC = HS    // High Frequency External Crystal selected
5 #pragma config WDT = OFF   //Watch Dog Timer OFF
6 #pragma config LVP = OFF   //Low Voltage Programming OFF
7 // LED Pin
8 #define LED LATCbits.LATC2    // RC.2 connected to LED
9 voidmain ()
10 {
11    TRISCbits.TRISC2 = 0;   // Configure LED Pin as Output Pin
12    while(1)
13    {
14      LED = 0;
15      __delay_ms(1000);   // 1000 mSec delay
16      LED = 1;
17      __delay_ms(1000);  // 1000 mSec delay
18    }
19 }
20
```

Lab 2 - LCD Interfacing

Aim: To study the 16x2 LCD Interfacing with PIC18F452 mini development kit.



Procedure:

1. Read the LCD Interfacing Program on next page with all the comments which explains the program.
2. Open the Project folder for the program in MPLAB X IDE or write it by making new project as per MPLAB X IDE procedure explained in the manual. Build and generate the Hex file for the same.
3. Open MPLAB IPE and follow the procedure explained to download the Hex file for the above program.
4. LCD contrast can be adjusted using 10k pot beside LCD. Once you download the program and remove the programmer, you will see above contents on the LCD.
5. Make sure that the arrow side of the programmer must be connected to the RST pin of the ICSP connector of the board.

Program:

```
#include<xc.h>
#define_XTAL_FREQ  20000000 // Define the crystal frequency (8 MHz)
#include<pic18F452.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#pragma config OSC = HS
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOR = OFF   // Brown-out Reset disabled
// LCD Pins
#define RS LATDbits.LATD0            // RS connected to RE0
#define EN LATDbits.LATD1            // EN connected to RE1
#define D4 LATDbits.LATD4            // Data line 4
#define D5 LATDbits.LATD5            // Data line 5
#define D6 LATDbits.LATD6            // Data line 6
#define D7 LATDbits.LATD7            // Data line 7
void LCD_Init();                 /*Initialize LCD*/
void LCD_Command(unsigned char );  /*Send command to LCD*/
void LCD_Char(unsigned char x);    /*Send data to LCD*/
void LCD_String(const char *);     /*Display data string on LCD*/
void LCD_String_xy(char, char , const char *);
void LCD_Clear();                /*Clear LCD Screen*/
int main(void)
{
   /* OSCCON = 0x72;  //Use internal oscillator and
                  // set frequency to 8 MHZ*/
   TRISD = 0x00; // Set PORTD as output for LCD data
      LCD_Init();  /*Initialize LCD to 5*8 matrix in 4-bit mode*/
      LCD_String_xy(1,0,"PIC Mini Kit by");  /*Display string on 1st row*/
   __delay_ms(1000);
   LCD_String_xy(2,0,"Amotech Labs");  /*Display string on 2nd row,*/
   __delay_ms(1000);
      while(1);
}
/***************************Functions*******************************/
void LCD_Init()
{
   LCD_Command(0x02); // Initialize LCD in 4-bit mode
   LCD_Command(0x28); // 2 lines, 5x7 matrix
   LCD_Command(0x0C); // Display ON, Cursor OFF
   LCD_Command(0x06); // Increment cursor
   LCD_Command(0x01); // Clear display
   __delay_ms(2);
}
void LCD_Command(unsigned char cmd )
{
       RS = 0; // Command mode
```

```
49     D4 = (cmd >> 4) & 0x01;
50     D5 = (cmd >> 5) & 0x01;
51     D6 = (cmd >> 6) & 0x01;
52     D7 = (cmd >> 7) & 0x01;
53     EN = 1; __delay_ms(1); EN = 0; // Latch the command
54
55     D4 = cmd & 0x01;
56     D5 = (cmd >> 1) & 0x01;
57     D6 = (cmd >> 2) & 0x01;
58     D7 = (cmd >> 3) & 0x01;
59     EN = 1; __delay_ms(1); EN = 0; // Latch the command
60 }
61 void LCD_Char(unsigned char data)
62 {
63         RS = 1; // Data mode
64     D4 = (data >> 4) & 0x01;
65     D5 = (data >> 5) & 0x01;
66     D6 = (data >> 6) & 0x01;
67     D7 = (data >> 7) & 0x01;
68     EN = 1; __delay_ms(1); EN = 0; // Latch the data
69
70     D4 = data & 0x01;
71     D5 = (data >> 1) & 0x01;
72     D6 = (data >> 2) & 0x01;
73     D7 = (data >> 3) & 0x01;
74     EN = 1; __delay_ms(1); EN = 0; // Latch the data
75 }
76 void LCD_String(const char *msg)
77 {    while((*msg)!=0) {
78         LCD_Char(*msg);
79           msg++;       }
80  }
81 void LCD_String_xy(char row,char pos,const char *msg)  {
82  char location=0;
83  if(row<=1) {
84  location=(0x80)|((pos)&0x0f);         /*Print message on 1st row and given column*/
85  LCD_Command(location); }
86  else {
87   location=(0xC0)|((pos)&0x0f);         /*Print message on 2nd row and given column*/
88   LCD_Command(location);
89    }
90   LCD_String(msg);
91  }
92  void LCD_Clear()
93  {
94      LCD_Command(0x01);  /*clear display screen*/
95        __delay_ms(3);
96   }
```

Lab 3 - ADC Interfacing

Aim: To study the ADC interfacing with PIC18F452 mini development kit.

10k Pot. inserted in Analog Input Slot

Procedure:

1. Read the ADC Interfacing Program on next page with all the comments which explains the program. Open the Project folder for the program in MPLAB X IDE or write it by making new project as per the MPLAB X IDE procedure explained in the manual. Build and generate the Hex File for the same.
2. Open MPLAB IPE and follow the procedure explained to download the Hex file for the above Program.
3. Use PICkit3 programmmer to program the PIC microcontroller with the HEX file.
4. Make sure that the arrow side of the programmer must be connected to the RST pin of the ICSP connector of the board.

Program:

```
#include<xc.h>
#include<pic18F452.h>
#include<stdio.h>
#include<string.h>
#include <stdlib.h>
#define _XTAL_FREQ  20000000 // Define the crystal frequency (20 MHz)
#pragma config OSC = HS
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOR = OFF   // Brown-out Reset disabled
// LCD Pins
#define RS LATDbits.LATD0// RS connected to RE0
#define EN LATDbits.LATD1// EN connected to RE1
#define D4 LATDbits.LATD4// Data line 4
#define D5 LATDbits.LATD5// Data line 5
#define D6 LATDbits.LATD6// Data line 6
#define D7 LATDbits.LATD7// Data line 7
/*********************Proto-Type Declaration*****************************/
void LCD_Init();                 /*Initialize LCD*/
void LCD_Command(unsigned char );  /*Send command to LCD*/
void LCD_Char(unsigned char x);    /*Send data to LCD*/
void LCD_String(const char *);     /*Display data string on LCD*/
void LCD_String_xy(char,char  , const char *);
void LCD_Clear();                /*Clear LCD Screen*/
void ADC_Init();
int ADC_Read(int);
#define vref 5.00
int main(void)
{
    char data[10];
    TRISD = 0x00; // Set PORTD as output for LCD data
    unsigned int digital;
    float voltage;
    int k;
    char ADC_Array[5];
   // OSCCON = 0x72; // Use internal oscillator and
                   //   set frequency to 8 MHZ
  LCD_Init();  /*Initialize LCD to 5*8 matrix in 4-bit mode*/
  ADC_Init();
  LCD_String_xy(1,0,"voltage is..");/*Display string on 1st row,5th location*/
    LCD_Command(0xC0);

        while(1)
   {
        digital=ADC_Read(3);
        /*Convert digital value into analog voltage*/
        voltage= digital*((float)vref/(float)1023);
        for(k=0; k<=3; k++)          /* Convert the result into ASCII*/
```

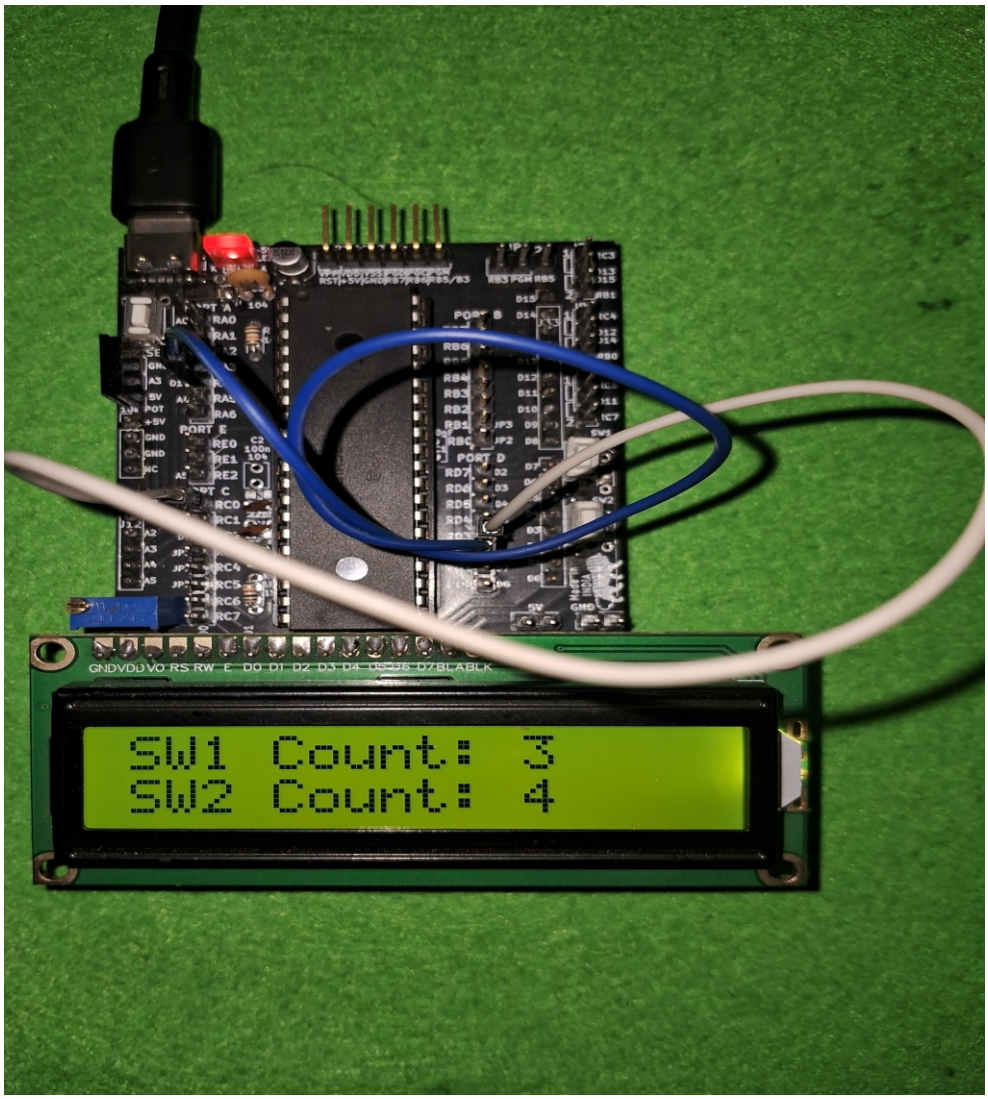
```
49  {                       /* Separate each digit of the integer */
50   ADC_Array[k] = digital%10+'0';
51   digital    = digital/10;
52  }
53  LCD_Command(0XC0);
54  for(k=3; k>=0; k--)
55  {
56   LCD_Char(ADC_Array[k]);         /* Display the result on LCD */
57  }
58       /*It is used to convert integer value to ASCII string*/
59       sprintf(data,"%.2f",voltage);
60
61       strcat(data," V");      /*Concatenate result and unit to print*/
62      LCD_String_xy(2,6,data);          /*Send string data for printing*/
63       __delay_ms(250);
64    }
65 }
66 void ADC_Init()
67 {
68    TRISA = 0xff;              /*Set as input port*/
69    ADRESH=0;                  /*Flush ADC output Register*/
70    ADRESL=0;
71 }
72 int ADC_Read(int channel)
73 {
74    unsigned int digital;
75    ADCON0 =(ADCON0 & 0b11000111)|((channel<<3) & 0b00111000);
76    /*channel 0 is selected i.e.(CHS3CHS2CHS1CHS0=0000)& ADC is disabled*/
77   // ADCON0 |= ((1<<ADON)|(1<<GO));/*Enable ADC and start conversion*/
78   ADCON0bits.ADON=1;
79   ADCON0bits.GO_nDONE=1;
80    /*wait for End of conversion i.e. Go/done'=0 conversion completed*/
81    while(ADCON0bits.GO_nDONE==1);
82    digital = ADRESH;
83    digital <<= 8;  //Left shift the lower byte into the higher byte of tvalue
84    digital += ADRESL; // Insert the TL0 byte into the lower byte of tvalue
85    digital >>= 6;
86    return(digital);
87 }
88 /*****************************Functions*********************************/
89 void LCD_Init()
90 {
91    LCD_Command(0x02); // Initialize LCD in 4-bit mode
92    LCD_Command(0x28); // 2 lines, 5x7 matrix
93    LCD_Command(0x0C); // Display ON, Cursor OFF
94    LCD_Command(0x06); // Increment cursor
95    LCD_Command(0x01); // Clear display
96    __delay_ms(2);
97 }
```

```
 98 void LCD_Command(unsigned char cmd )
 99 {
100         RS = 0; // Command mode
101     D4 = (cmd >> 4) & 0x01;
102     D5 = (cmd >> 5) & 0x01;
103     D6 = (cmd >> 6) & 0x01;
104     D7 = (cmd >> 7) & 0x01;
105     EN = 1; __delay_ms(1); EN = 0; // Latch the command
106
107     D4 = cmd & 0x01;
108     D5 = (cmd >> 1) & 0x01;
109     D6 = (cmd >> 2) & 0x01;
110     D7 = (cmd >> 3) & 0x01;
111    EN = 1; __delay_ms(1); EN = 0; // Latch the command
112 }
113 void LCD_Char(unsigned char data)
114 {
115         RS = 1; // Data mode
116     D4 = (data >> 4) & 0x01;
117     D5 = (data >> 5) & 0x01;
118     D6 = (data >> 6) & 0x01;
119     D7 = (data >> 7) & 0x01;
120     EN = 1; __delay_ms(1); EN = 0; // Latch the data
121
122     D4 = data & 0x01;
123     D5 = (data >> 1) & 0x01;
124     D6 = (data >> 2) & 0x01;
125     D7 = (data >> 3) & 0x01;
126     EN = 1; __delay_ms(1); EN = 0; // Latch the data
127 }
128 void LCD_String(const char *msg)
129 {
130         while((*msg)!=0)
131         {
132           LCD_Char(*msg);
133          msg++;
134     }
135 }
136 void LCD_String_xy(char row,char pos,const char *msg)
137 {
138     char location=0;
139     if(row<=1)
140     {
141         location=(0x80) | ((pos) & 0x0f);  /*Print message on 1st row
142                                   * and desired location*/
143         LCD_Command(location);
144     }
```

```
145     else
146     {
147         location=(0xC0) | ((pos) & 0x0f);  /*Print message on 2nd row
148                                             *  and desired location*/
149         LCD_Command(location);
150     }
151
152     LCD_String(msg);
153 }
154 void LCD_Clear()
155 {
156         LCD_Command(0x01);  /*clear display screen*/
157     __delay_ms(3);
158 }
159
160
```

Lab 4 - Pulse Counter

Aim: To study the Pulse counter program with PIC18F451 mini board.



.**Procedure:**

1. Read the pulse counter Program on next page with all the comments which explains the program.
2. Open the Project folder for the program in MPLAB X IDE or write it by making new project as per MPLAB X IDE procedure explained in the manual. Build and generate the Hex file for the same.
3. Open MPLAB IPE and follow the procedure explained to download the Hex file for the above program.
4. Make the connections as above shown in figure.

Program:

```
#include<xc.h>                          /*The #include <xc.h> directive is used in PIC microcontroller
                                        programming with the MPLAB X IDE and XC8 compiler */
#define _XTAL_FREQ  20000000            // Define the crystal frequency (20 MHz)
#include<pic18F452.h>
#include<string.h>
#include<stdio.h>                       // For sprintf
#pragma config OSC = HS
#pragma config WDT = OFF                //Watchdog Timer is disabled.
#pragma config LVP = OFF                /* LVP is disabled*/
#pragma config BOR = OFF                 // Brown-out Reset disabled
// LCD Pins
#define RS LATDbits.LATD0               // RS connected to RE0
#define EN LATDbits.LATD1               // EN connected to RE1
#define D4 LATDbits.LATD4               // Data line 4
#define D5 LATDbits.LATD5               // Data line 5
#define D6 LATDbits.LATD6               // Data line 6
#define D7 LATDbits.LATD7               // Data line 7
// Function prototypes
void LCD_Init(void);
void LCD_Command(unsigned char cmd);
void LCD_Char(unsigned char data);
void LCD_String(const char *str);
void LCD_Clear(void);
void LCD_SetCursor(unsigned char row, unsigned char column);
void setup();
void displayCount(unsigned int count);
// Global variables
volatile unsigned int pulseCount = 0;

void setup() {
    // Configure Timer1 in Counter Mode
    T1CON = 0x87;                       //Timer1 ON, External clock from RC2 (T1CKI),Prescaler 1:1
    TRISCbits.TRISC0 = 0;               // Set RC2/T1CKI as input
     // Configure Timer0 in Counter Mode for Ra4
    T0CON = 0xA8;                       // Timer0 ON, 8-bit counter mode,
                                         //External clock source on RA4/T0CKI, no prescaler

    TRISAbits.TRISA4 = 1;               // Set RA4 as input
    TRISD = 0x00;                       // Set PORTD as output for LCD data
    LCD_Init();                         // Initialize the LCD
    TMR0L = 0;                          // Clear Timer0 and Timer1 counts
    TMR0H = 0;
    TMR1H = 0;                          // Clear Timer1 High byte
    TMR1L = 0;                          // Clear Timer1 Low byte
}
```

```
45 void main() {
46     setup();
47     LCD_Clear();
48     while (1)
49     {
50      // Read the Timer1 counter value (pulse count)
51      unsigned int t1PulseCount = (TMR1H << 8) | TMR1L;
52      unsigned int t0PulseCount = (TMR0H << 8) | TMR0L;
53      char t0buffer[16],t1buffer[16];
54      sprintf(t1buffer, "SW1 Count: %u", t1PulseCount);          //Format the count value
55      sprintf(t0buffer, "SW2 Count: %u", t0PulseCount);          //Format the count value
56      // LCD_Clear();
57      LCD_SetCursor(1,1);
58      LCD_String(t1buffer);                                       // Display on LCD
59      LCD_SetCursor(2,1);
60      LCD_String(t0buffer);                                      // Display on LCD
61       __delay_ms(5);                                            // Update the display every 500 ms
62     }
63 }
64 void displayCount(unsigned int count) {
65     char buffer[16];
66     LCD_Clear();
67     sprintf(buffer, "Count: %u", count); // Format the count value
68     LCD_String(buffer); // Display on LCD
69 }
70 void LCD_Init(void) {
71     LCD_Command(0x02);                                  // Initialize LCD in 4-bit mode
72     LCD_Command(0x28);                                   // 2 lines, 5x7 matrix
73     LCD_Command(0x0C);                                   // Display ON, Cursor OFF
74     LCD_Command(0x06);                                  // Increment cursor
75     LCD_Command(0x01);                                  // Clear display
76     __delay_ms(2);
77 }
78 void LCD_Command(unsigned char cmd) {
79     RS = 0; // Command mode
80     D4 = (cmd >> 4) & 0x01;
81     D5 = (cmd >> 5) & 0x01;
82     D6 = (cmd >> 6) & 0x01;
83     D7 = (cmd >> 7) & 0x01;
84     EN = 1;
85      __delay_ms(1); EN = 0;                             // Latch the command
86     D4 = cmd & 0x01;
87     D5 = (cmd >> 1) & 0x01;
88     D6 = (cmd >> 2) & 0x01;
89     D7 = (cmd >> 3) & 0x01;
90     EN = 1; __delay_ms(1); EN = 0;                      // Latch the command
91 }
92 void LCD_Char(unsigned char data) {
93     RS = 1; // Data mode
94     D4 = (data >> 4) & 0x01;
95     D5 = (data >> 5) & 0x01;
```

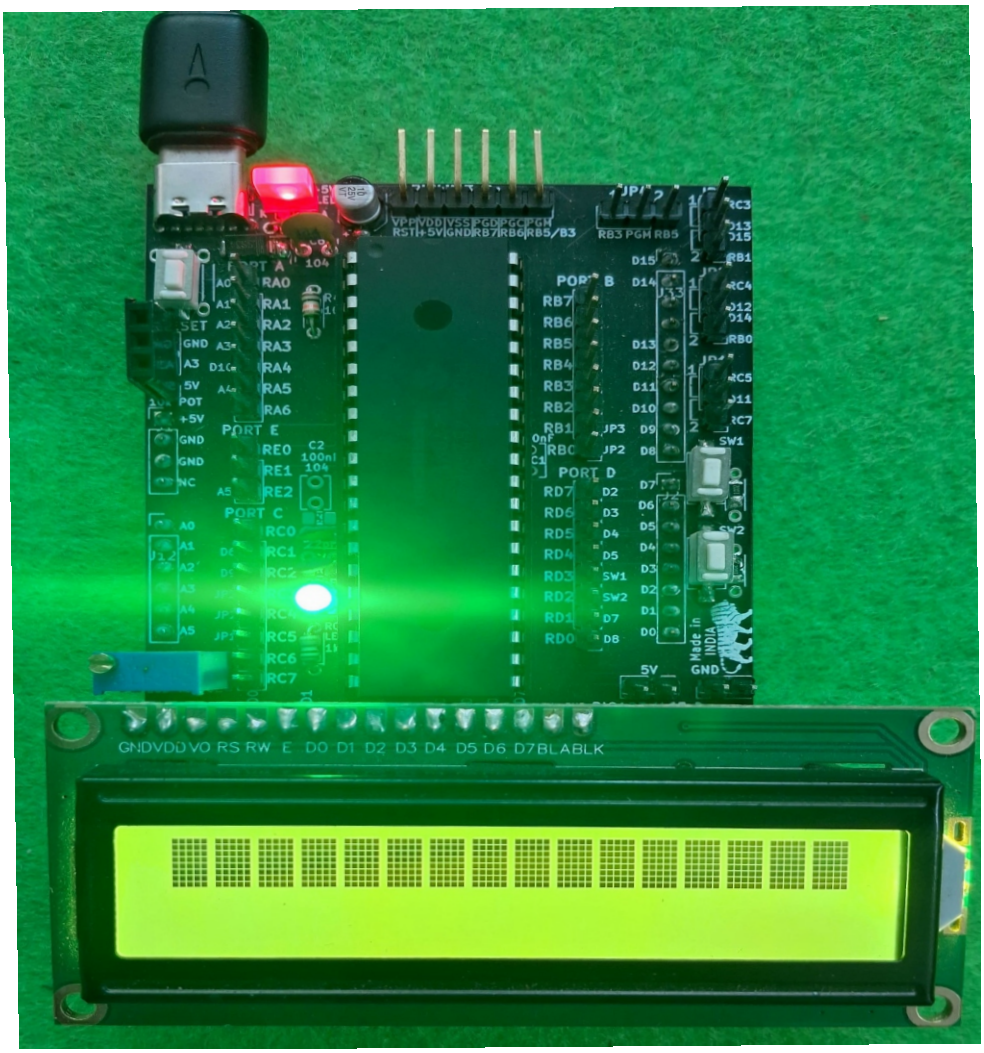
```
    D6 = (data >> 6) & 0x01;
    D7 = (data >> 7) & 0x01;
    EN = 1; __delay_ms(1); EN = 0;                  // Latch the data

    D4 = data & 0x01;
    D5 = (data >> 1) & 0x01;
    D6 = (data >> 2) & 0x01;
    D7 = (data >> 3) & 0x01;
    EN = 1; __delay_ms(1); EN = 0;                  // Latch the data
}
void LCD_String(const char *str) {
    while (*str) {
        LCD_Char(*str++);
    }
}
void LCD_Clear(void) {
    LCD_Command(0x01);                              // Clear display
    __delay_ms(2);
}
void LCD_SetCursor(unsigned char row, unsigned char column) {
    unsigned char position;
    if (row == 1) position = 0x80 + column - 1;
    else if (row == 2) position = 0xC0 + column - 1;
    LCD_Command(position);
}

```

Lab 5 - PWM

Aim: To study the PWM with PIC18F452 mini development kit



Procedure:

1. Read the PWM Program on next page with all the comments which explains the program.
2. Open the Project folder for the program in MPLAB X IDE or write it by making new project as per MPLAB X IDE procedure explained in the manual. Build and generate the Hex file for the same.
3. Open MPLAB IPE and follow the procedure explained to download the Hex file for the above program.
4. As you vary the PWM signals fed to pin **RC2**, you will see brightness of LED connected to **RC2** changes.
5. Make sure that the arrow side of the programmer must be connected to the RST pin of the ICSP connector of the board.

Program:

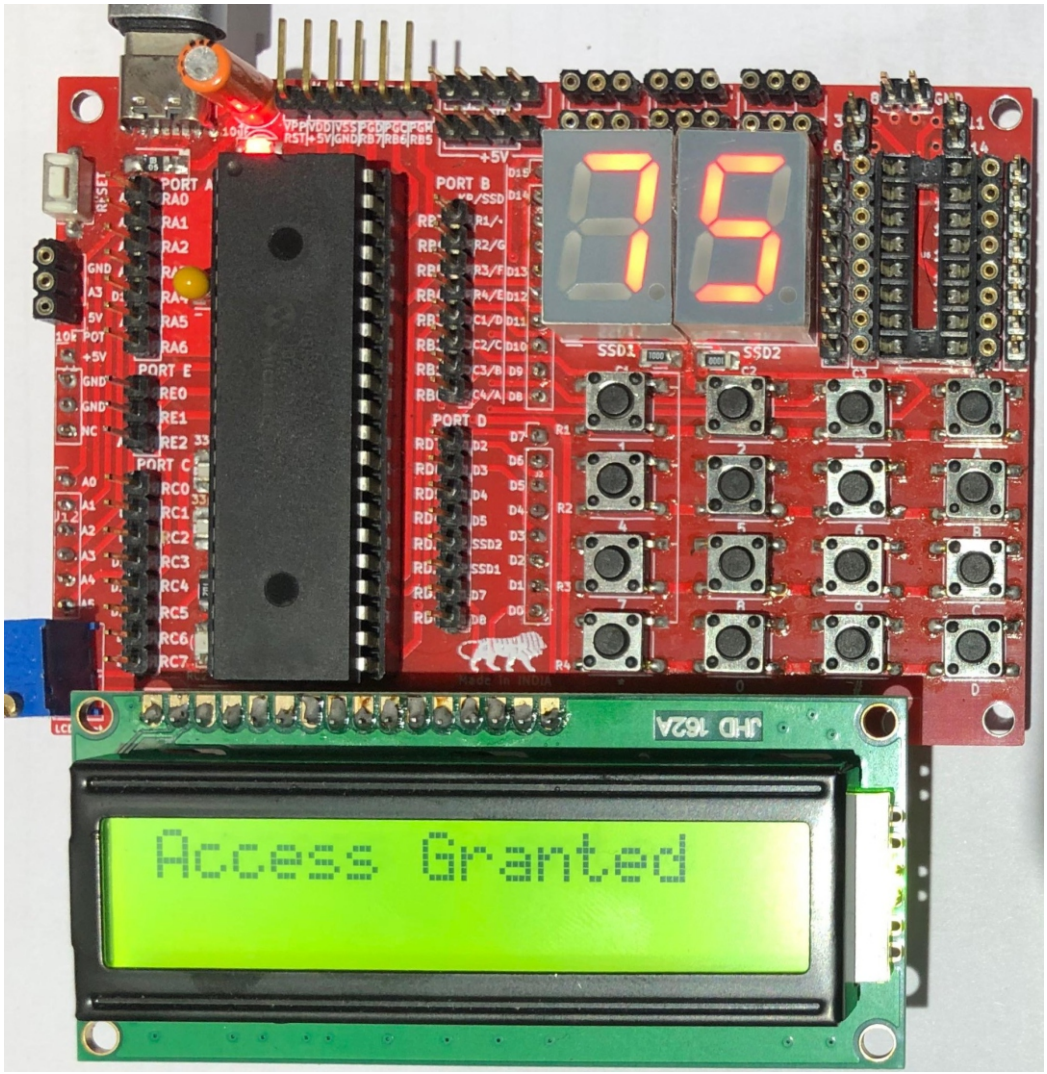
```
#include<xc.h>
#include<pic18F452.h>
#include<string.h>
// Configuration Bits
#pragma config OSC = HS       // High-speed oscillator
#pragma config WDT = OFF      // Watchdog Timer Disable
#pragma config LVP = OFF      // Low Voltage ICSP Disable
#pragma config BOR = OFF      // Brown-out Reset Disable
#define _XTAL_FREQ  20000000   // Define system clock frequency (20 MHz)
void main()
{
    unsigned int duty_cycle;   /* Set CCP1 pin (RC2) as output for PWM */
    TRISCbits.TRISC2 = 0;
    PR2 = 124;    /* Load period value in PR2 register for 100us period*/
    CCPR1L = 1;   /* Load initial duty cycle */
    T2CON = 0x04;     /* Timer2 ON, Prescaler = 4 */
    CCP1CON = 0x0C;   /* Set PWM mode */
    TMR2 = 0;        /* Clear Timer2 */
    T2CONbits.TMR2ON = 1;  /* Turn ON Timer2 */
    while(1)
    {
        for(duty_cycle = 1; duty_cycle < 124; duty_cycle++)
        {
            CCPR1L = duty_cycle;
             __delay_ms(20);
        }
         __delay_ms(500);

        for(duty_cycle = 124; duty_cycle > 1; duty_cycle--)
        {
            CCPR1L = duty_cycle;
             __delay_ms(20);
        }
        __delay_ms(500);
    }
}

```

Lab 6 - Interfacing of Keypad and Seven Segment Display

Aim: To study the Interfacing of Keypad and 2 Seven Segment Displays on a 'PIC18F/16F Development Board' which has on-board Keypad, 2 Seven Segment Displays and 1 L293D IC/16 pin DIP IC slot.



Procedure:

1. In this program, we enter the 4 digit code '4201' after which 'Access Granted' appears on LCD and a 2 digit number from 0 to 99 are displayed on 2 Seven Segment Displays after 1 Second delay.
2. Read the Program on next page with all the comments which explains the program.
3. Open the Project folder for the program in MPLAB X IDE or write it by making new project as per MPLAB X IDE procedure explained in the manual. Build and generate the Hex file for the same.
4. Open MPLAB IPE and follow the procedure explained to download the Hex file for the above program.
5. As you vary the PWM signals fed to pin **RC2**, you will see brightness of LED connected to **RC2** changes.
6. Make sure that the arrow side of the programmer must be connected to the RST pin of the ICSP connector of the board.

Program:

```
#include <xc.h>/*The #include <xc.h> directive is used in PIC microcontroller
               programming  with the MPLAB X IDE and XC8 compiler to include
               a device-specific header file.*/
#define _XTAL_FREQ 20000000 // Define the crystal frequency (20 MHz)
#include<pic18F452.h>
#include<string.h>
#pragma config OSC = HS
#pragma config WDT = OFF /*Watchdog Timer is disabled. The microcontroller will
                    not reset due to the watchdog timer overflow.*/
#pragma config LVP = OFF /*When LVP is enabled, the PIC18F452 can be programmed
                  in a low-voltage state  (typically around 2V), which
                  is especially useful for in-circuit programming when the
                  voltage supply may not be at the nominal operating voltage*/
#define col1 PORTBbits.RB3
#define col2 PORTBbits.RB2
#define col3 PORTBbits.RB1
#define col4 PORTBbits.RB0
#define row1 PORTBbits.RB7
#define row2 PORTBbits.RB6
#define row3 PORTBbits.RB5
#define row4 PORTBbits.RB4
//Seven Segment Display Pins
#define SSD1 PORTDbits.RD2
#define SSD2 PORTDbits.RD3
unsigned char value[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F},j,k;
// LCD Pins
#define RS LATDbits.LATD0 // RS connected to RE0
#define EN LATDbits.LATD1 // EN connected to RE1
#define D4 LATDbits.LATD4 // Data line 4
#define D5 LATDbits.LATD5 // Data line 5
#define D6 LATDbits.LATD6 // Data line 6
#define D7 LATDbits.LATD7 // Data line 7
void LCD_Init();                /*Initialize LCD*/
void LCD_Command(unsigned char );  /*Send command to LCD*/
void LCD_Char(unsigned char x);    /*Send data to LCD*/
void LCD_String(const char *);     /*Display data string on LCD*/
void LCD_Clear();
char pass[4],i=0;
void keypad_init()
{
    TRISB = 0xF0; // Set rows as output and columns as input
    PORTB = 0x00; // Clear rows
    INTCON2bits.RBPU = 0;    // Enable internal pull-ups
}
```

```
45 /*****************************Functions*********************************/
46 void LCD_Init()
47 {
48    LCD_Command(0x02); // Initialize LCD in 4-bit mode
49    LCD_Command(0x28); // 2 lines, 5x7 matrix
50    LCD_Command(0x0C); // Display ON, Cursor OFF
51    LCD_Command(0x06); // Increment cursor
52    LCD_Command(0x01); // Clear display
53    __delay_ms(2);
54 }
55 void LCD_Command(unsigned char cmd )
56 {
57    RS = 0; // Command mode
58    D4 = (cmd >> 4) & 0x01;
59    D5 = (cmd >> 5) & 0x01;
60    D6 = (cmd >> 6) & 0x01;
61    D7 = (cmd >> 7) & 0x01;
62    EN = 1; __delay_ms(1); EN = 0; // Latch the command
63
64    D4 = cmd & 0x01;
65    D5 = (cmd >> 1) & 0x01;
66    D6 = (cmd >> 2) & 0x01;
67    D7 = (cmd >> 3) & 0x01;
68    EN = 1; __delay_ms(1); EN = 0; // Latch the command
69 }
70 void LCD_Char(unsigned char data)
71 {
72       RS = 1; // Data mode
73    D4 = (data >> 4) & 0x01;
74    D5 = (data >> 5) & 0x01;
75    D6 = (data >> 6) & 0x01;
76    D7 = (data >> 7) & 0x01;
77    EN = 1; __delay_ms(1); EN = 0; // Latch the data
78
79    D4 = data & 0x01;
80    D5 = (data >> 1) & 0x01;
81    D6 = (data >> 2) & 0x01;
82    D7 = (data >> 3) & 0x01;
83    EN = 1; __delay_ms(1); EN = 0; // Latch the data
84 }
85 void LCD_String(const char *msg)
86 {
87       while((*msg)!=0)
88       {
89        LCD_Char(*msg);
90        msg++;
91    }
92 }
```

```
93 void LCD_Clear()
94 {
95       LCD_Command(0x01);  /*clear display screen*/
96    __delay_ms(3);
97 }
98 void keypad()
99 {
100    int cursor=192,flag=0;
101    LCD_Clear();
102    LCD_String("Enter Ur Passkey");
103    LCD_Command(0xc0);
104    i=0;
105    while(i<4)
106    {
107    flag=cursor;
108     col1=0;
109     col2=col3=col4=1;
110     if(!row1)
111      {
112       LCD_Char('1');
113       pass[i++]='1';
114       cursor++;
115       while(!row1);
116      }
117     else if(!row2)
118     {
119       LCD_Char('4');
120       pass[i++]='4';
121       cursor++;
122       while(!row2);
123     }
124     else if(!row3)
125    {
126       LCD_Char('7');
127       pass[i++]='7';
128       cursor++;
129       while(!row3);
130    }
131     else if(!row4)
132    {
133       LCD_Char('*');
134       pass[i++]='*';
135       cursor++;
136       while(!row4);
137    }
138    col2=0;
139    col1=col3=col4=1;
140    if(!row1)
141    {
142      LCD_Char('2');
143      pass[i++]='2';
144      cursor++;
145      while(!row1);
146    }
147     else if(!row2)
148    {
149      LCD_Char('5');
150      pass[i++]='5';
151      cursor++;
152      while(!row2);
153    }
154     else if(!row3)
155    {
156      LCD_Char('8');
157      pass[i++]='8';
158      cursor++;
159      while(!row3);
160    }
161     else if(!row4)
162    {
163      LCD_Char('0');
164      pass[i++]='0';
165      cursor++;
166      while(!row4);
167     }
168    col3=0;
169    col1=col2=col4=1;
170    if(!row1)
171     {
172      LCD_Char('3');
173      pass[i++]='3';
174      cursor++;
175      while(!row1);
176     }
177     else if(!row2)
178    {
179      LCD_Char('6');
180      pass[i++]='6';
181      cursor++;
182      while(!row2);
183     }
```

```
184     else if(!row3)
185     {
186       LCD_Char('9');
187       pass[i++]='9';
188       cursor++;
189       while(!row3);
190     }
191     else if(!row4)
192     {
193       LCD_Char('#');
194       pass[i++]='#';
195       cursor++;
196       while(!row4);
197     }
198     col4=0;
199    col1=col3=col2=1;
200    if(!row1)
201    {
202       LCD_Char('A');
203       pass[i++]='A';
204       cursor++;
205       while(!row1);
206     }
207     else if(!row2)
208    {
209       LCD_Char('B');
210       pass[i++]='B';
211       cursor++;
212       while(!row2);
213     }
214     else if(!row3)
215    {
216       LCD_Char('C');
217       pass[i++]='C';
218       cursor++;
219       while(!row3);
220     }
221     else if(!row4)
222    {
223       LCD_Char('D');
224       pass[i++]='D';
225       cursor++;
226       while(!row4);
227     }
228    if(i>0)
229    {
230        if(flag!=cursor)
231        __delay_ms(100);
232          LCD_Command(cursor-1);
233          LCD_Char('*');
234      }
235   }
236 }
237 void accept()
238 {
239     LCD_Clear();
240     LCD_String("Welcome");
241     LCD_Command(192);
242     LCD_String("Password Accept");
243     __delay_ms(200);
244 }
245 void wrong()
246 {
247     LCD_Clear();
248     LCD_String("Wrong Passkey");
249     LCD_Command(192);
250     LCD_String("PLZ Try Again");
251     __delay_ms(200);   // buzzer=1;
252 }
253 void main()
254 {
255     INTCON = 0x00;  //Disable all Interrupts
256     TRISD=0X00; //make PORTD o/p
257     LCD_Init();
258     LCD_String("Electronic Code");
259     LCD_Command(0xc0);
260     LCD_String("  Lock System  ");
261     __delay_ms(2000);
262     LCD_Clear();
263     LCD_String("by Amotech Labs");
264     __delay_ms(2000);
```

```
265   while(1)
266   {
267         i=0;
268         keypad_init();
269         keypad();
270         if(strncmp(pass,"4201",4)==0)
271         {
272         accept();
273         LCD_Clear();
274          LCD_String("Access Granted ");
275         __delay_ms(300);
276           TRISB = 0x00; //Now configure all PORTB as to outputs for SSD
277           INTCON2bits.RBPU = 1;// Disable pull-ups (optional)
278           while(1)
279           {
280            SSD1 = 1;
281            SSD2 = 1;
282            for (i=0;i<10;i++)
283             {
284               for (j=0;j<10;j++)
285               {
286                for (k=0;k<100;k++)
287                 {
288                  SSD1 = 0;
289                  SSD2 = 1;
290                  PORTB = value[i];
291                  __delay_ms(5);  //add delay of one second
292                  SSD1 = 1;
293                  SSD2 = 0;
294                  PORTB = value[j];
295                  __delay_ms(5);
296                 }
297               }
298             }
299           }
300         }
301      else
302      {
303        LCD_Clear();
304         LCD_String("Access Denied");
305        wrong();
306        __delay_ms(300);
307      }
308   }
309 }
```