PIC16F/18F MCU Mini Development Board Kit (PIC18F452 MCU)

LAB MANUAL

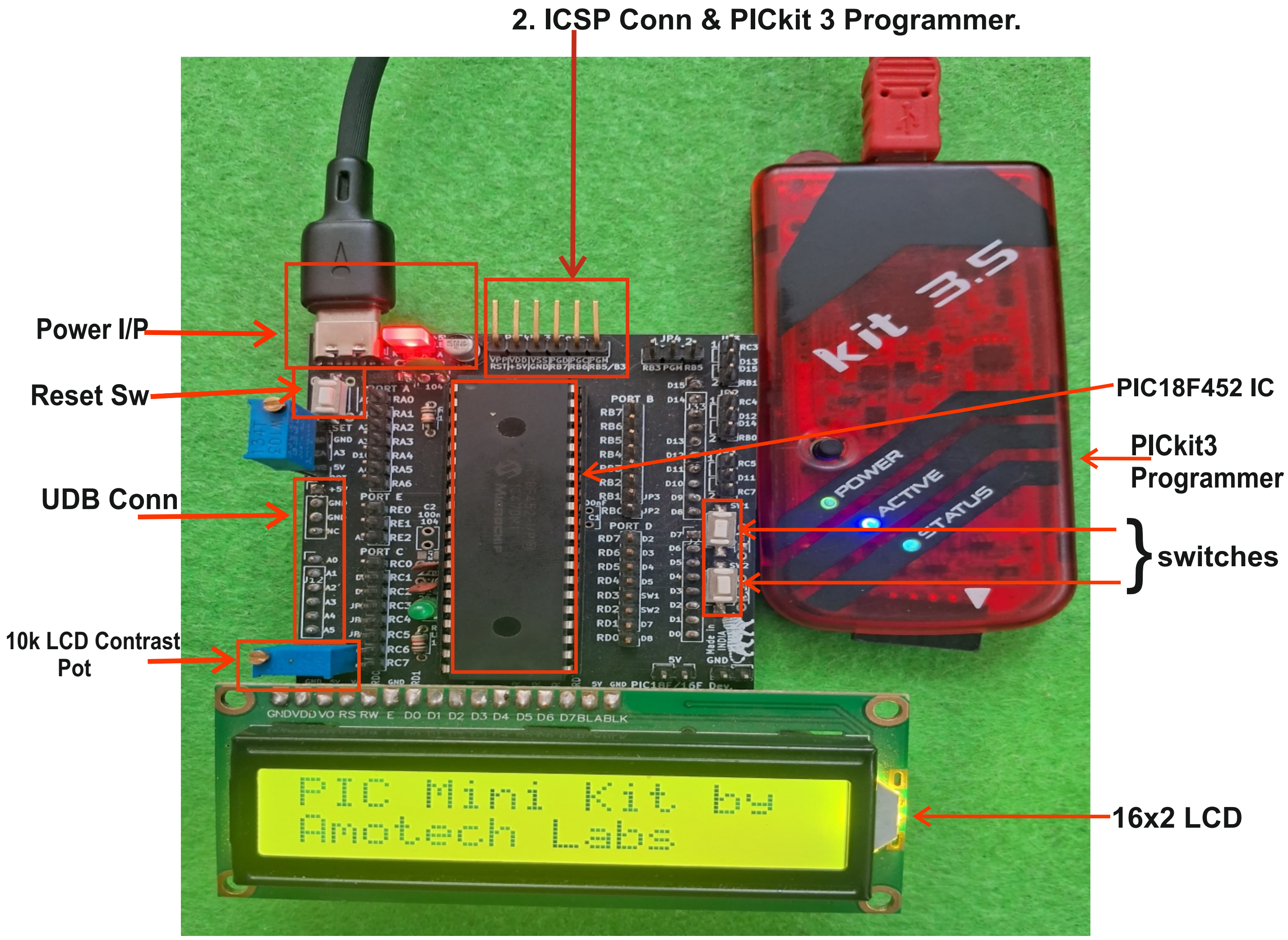
AMOTECH LABS

Embedded systems | Industrial Automation | Robotics

What'sApp /call: 832973756

Contents

Overview of PIC18F452 Development Board Kit:



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. 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.

2. ICSP Conn.: It is connected to PIC Kit 3 to program the MCU. RST pin of the kit is connected on a side where there is a arrow symbol indicating RST pin of the Programmer.

3. 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.

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

5. Keypad DIP.: This DIP switch connects 4x4 Keypad to PortD externally.

6. +5V & GND Conn.: Using this connectors +5V GND can be given to external devices. Don't short them.

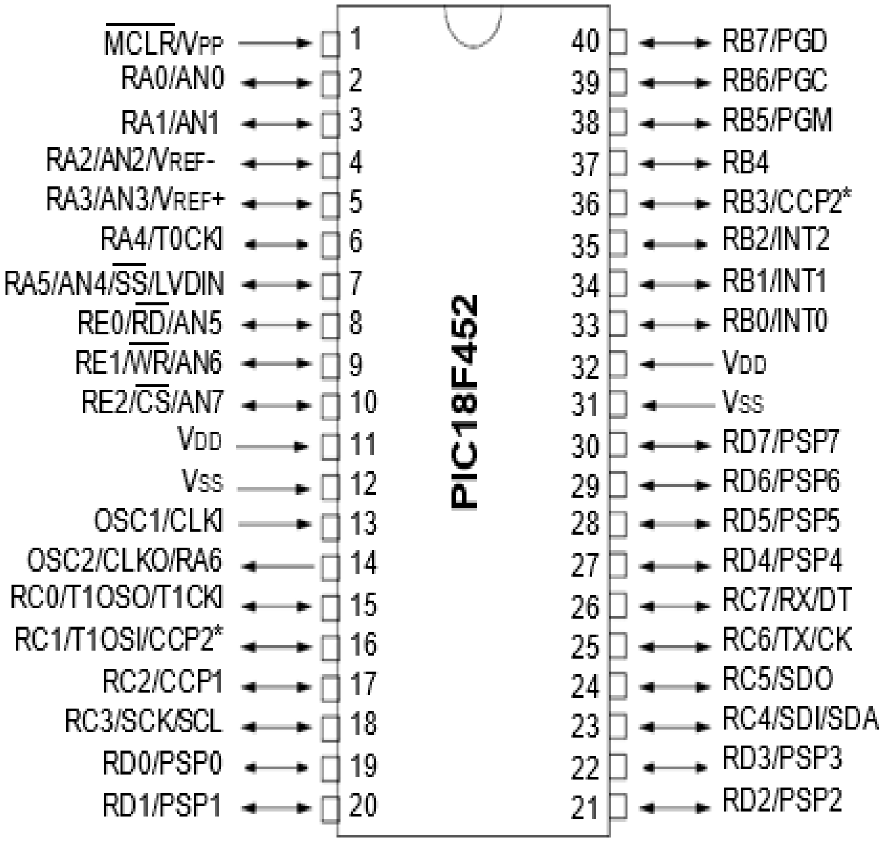
7. Power I/P : Provides power to the board via USB or an adapter.

8. Reset Sw : Resets the microcontroller to restart program execution.

9. LCD Contrast Pot : Adjusts the 16x2 LCD display contrast.

10. Switches: User inputs for controlling various functions.

Introduction to PIC18F452 Microcontroller



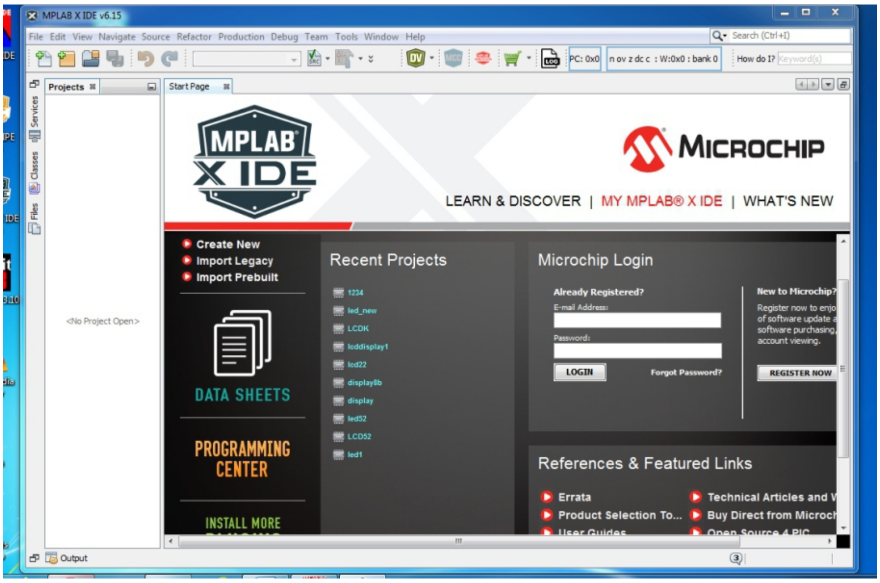
Features of PIC18f452:

✓ 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

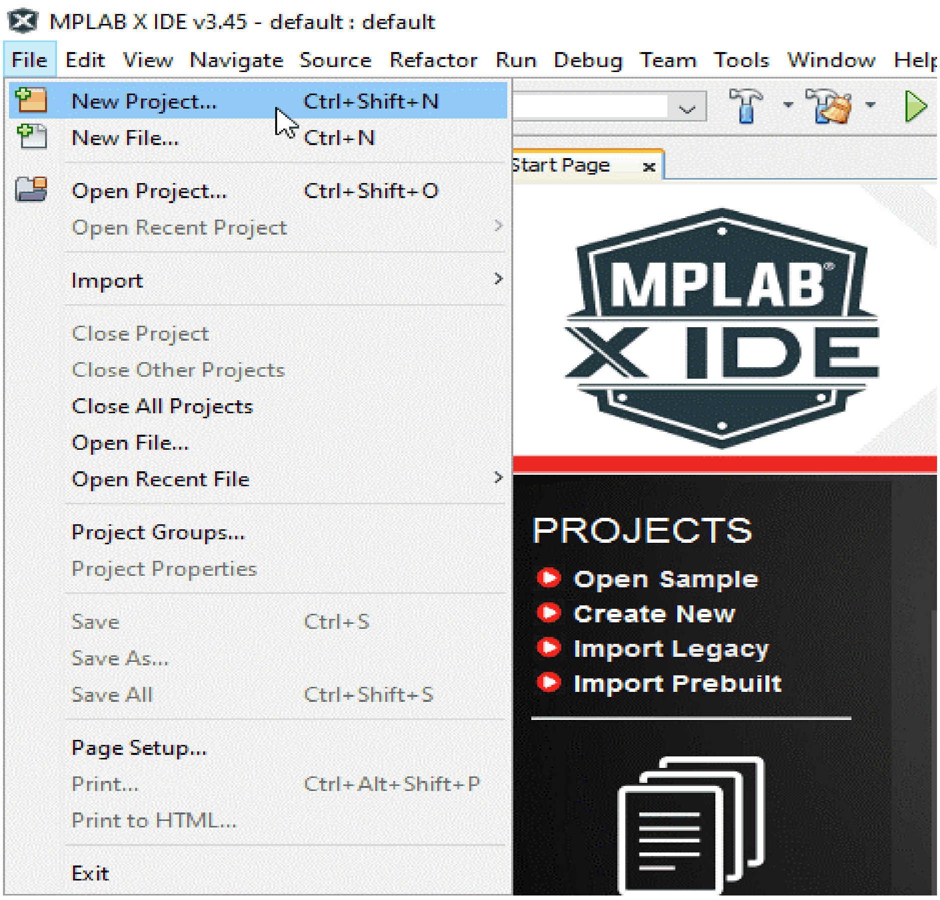
MPLAB X IDE for Programming PIC18F/16F MCU's

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

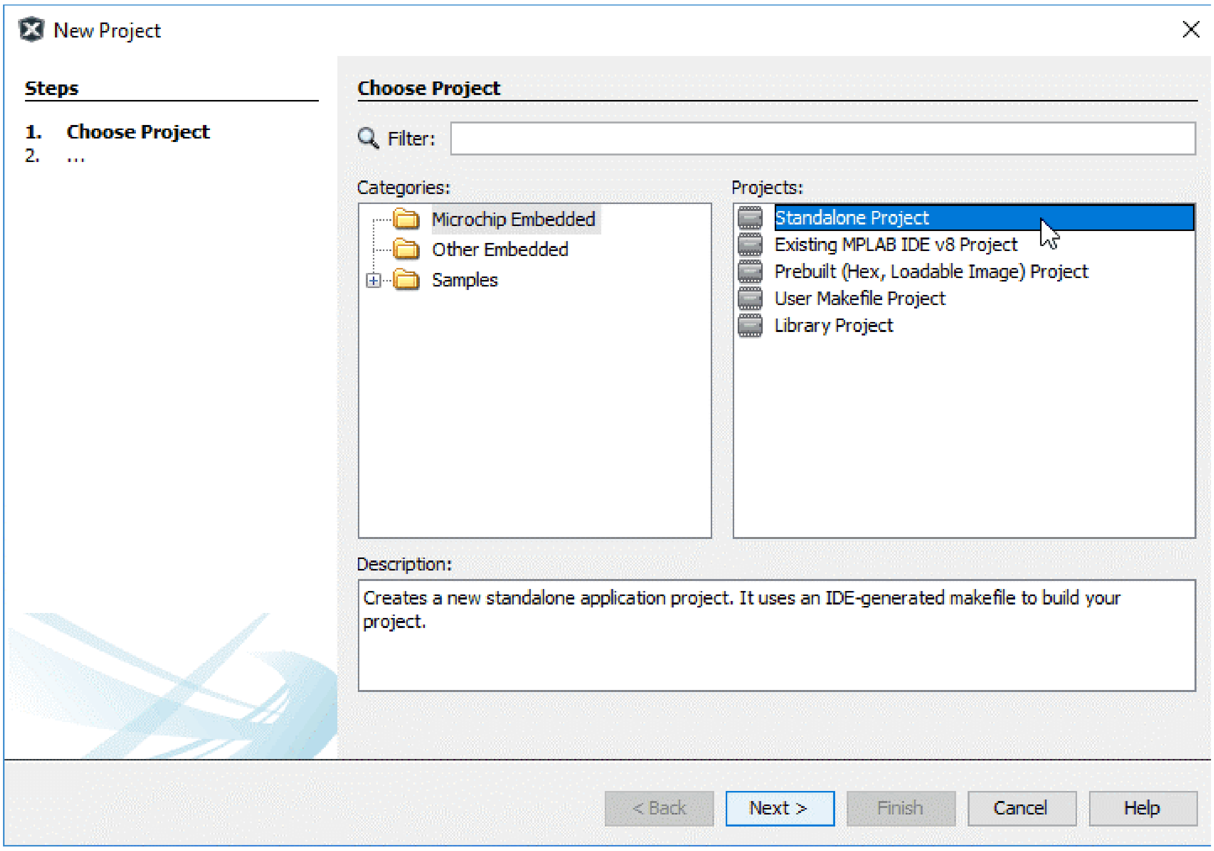
1. Double click on MPLAB X IDE icon on the desktop & Open MPLAB X IDE.



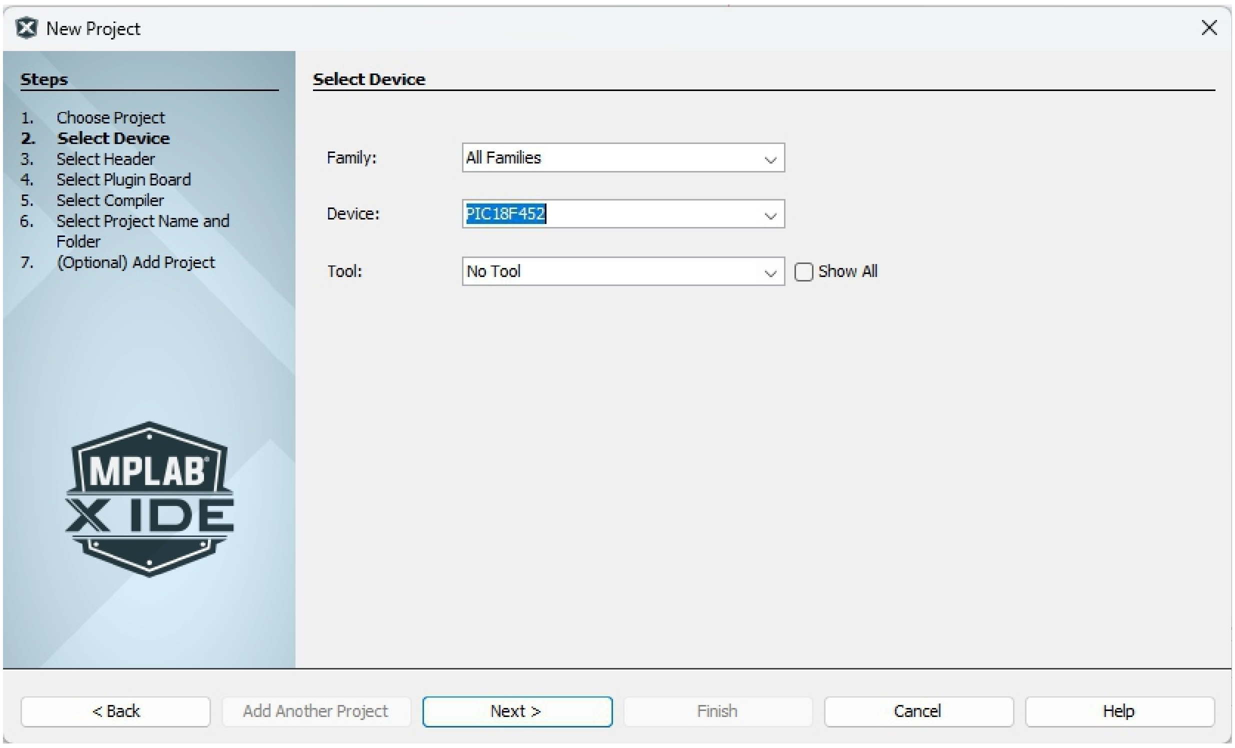
2. To create new project. Select File -> New Project.



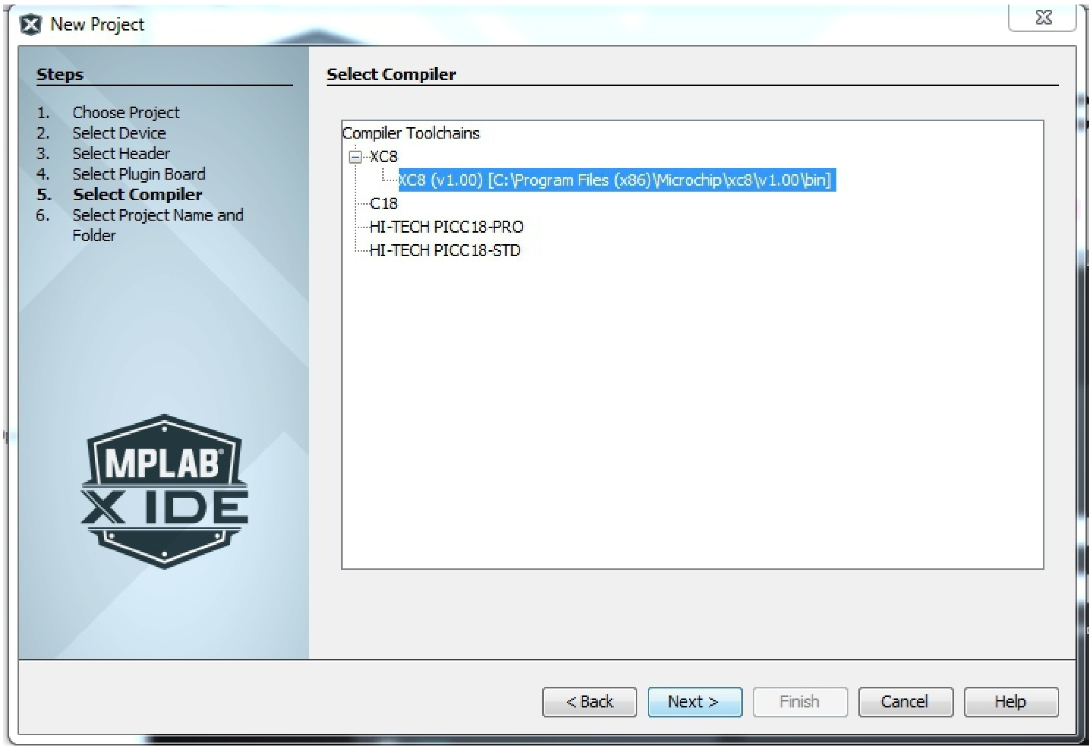
3. After that below window will appear. Select Microchip Embedded ->Standalone Project & Click next.



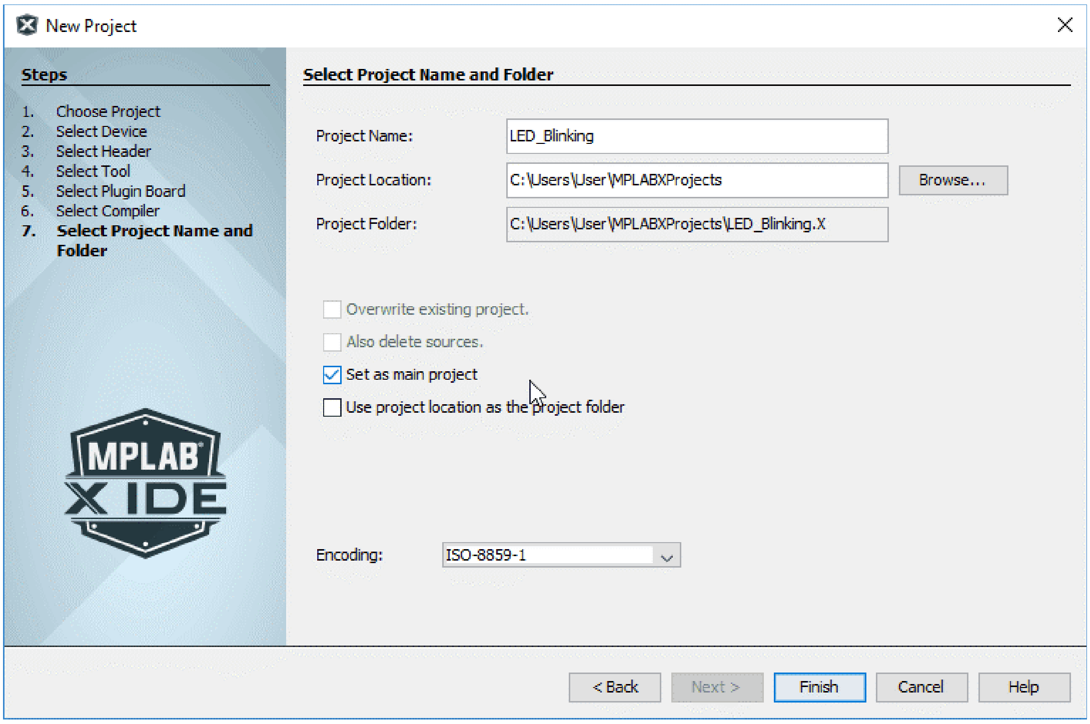
4. Select Family >Advanced 8-bit MCUs(PIC18) then select Device>PIC18F452 or any other MCU. In Tool you can select No Tool.



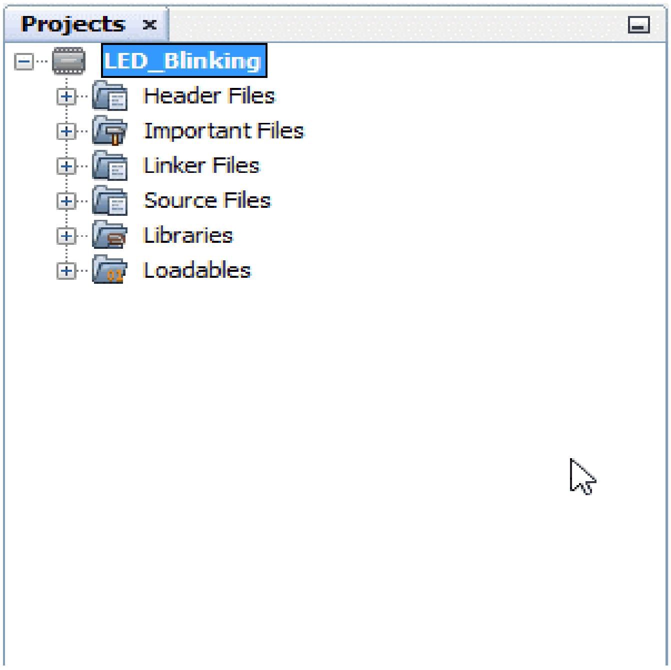
5. Now select the previously installed latest version of XC8 compiler and then click Next.



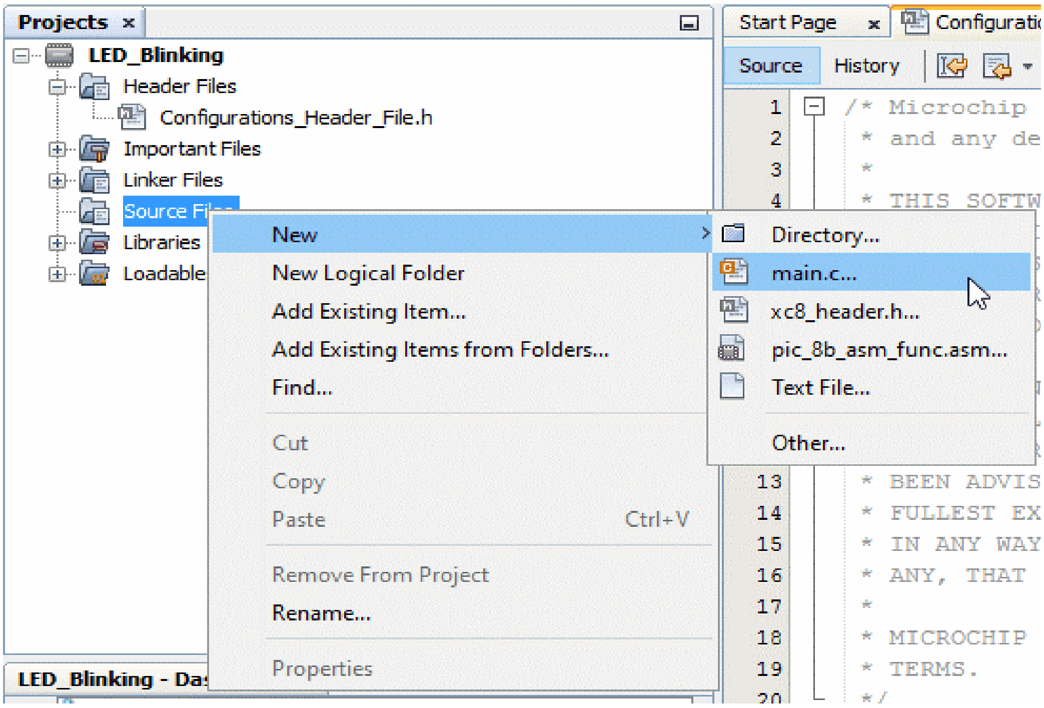
6. Enter your Project name > Click on Browse and Choose Project Folder location > Click on Finish.



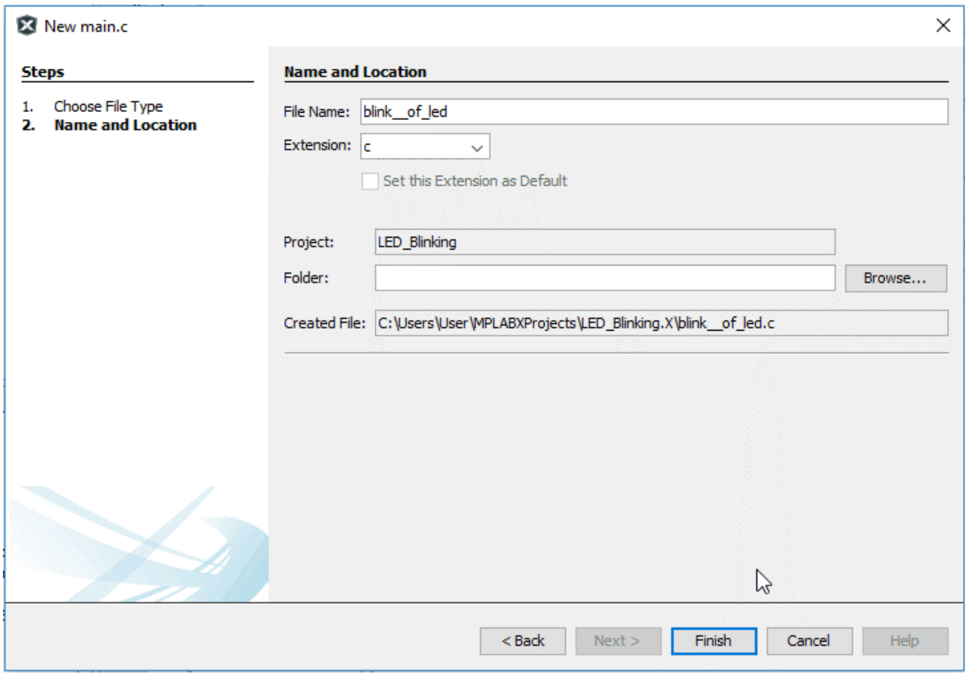
7. Now our project is displayed in Project window.



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

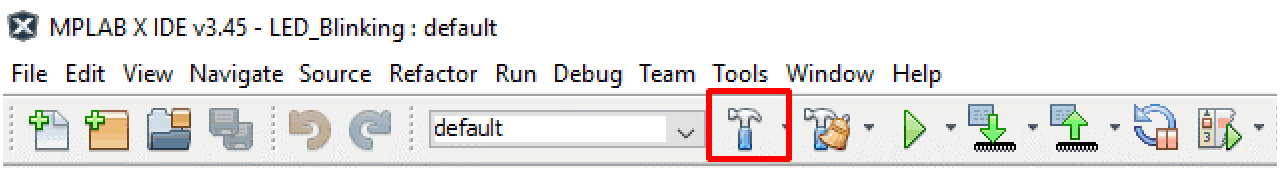


9. Give name to C file and click on finish.

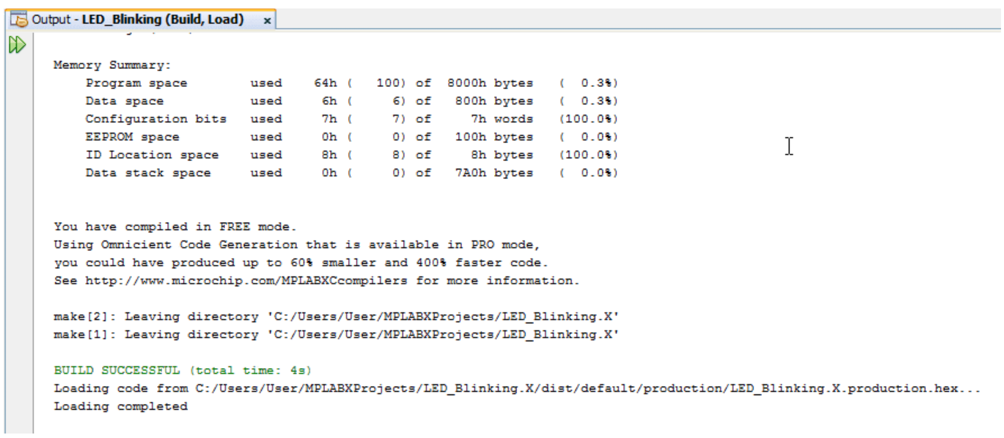


10. Write your program.

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.



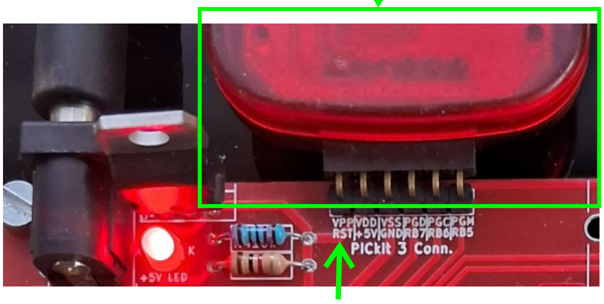
12. After BUILD SUCCESSFUL, Hex file is created in the project folder. In project folder Hex file is created in dist/default/production. You can also see a path to the folder under BUILD SUCCESSFUL.

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

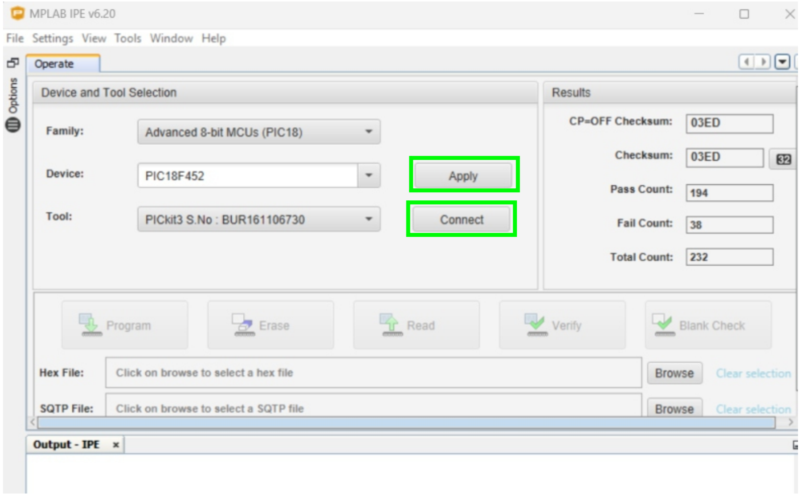
Below are the steps to program any PIC18F/16F MCU in MPLAB X IDE.

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'**.

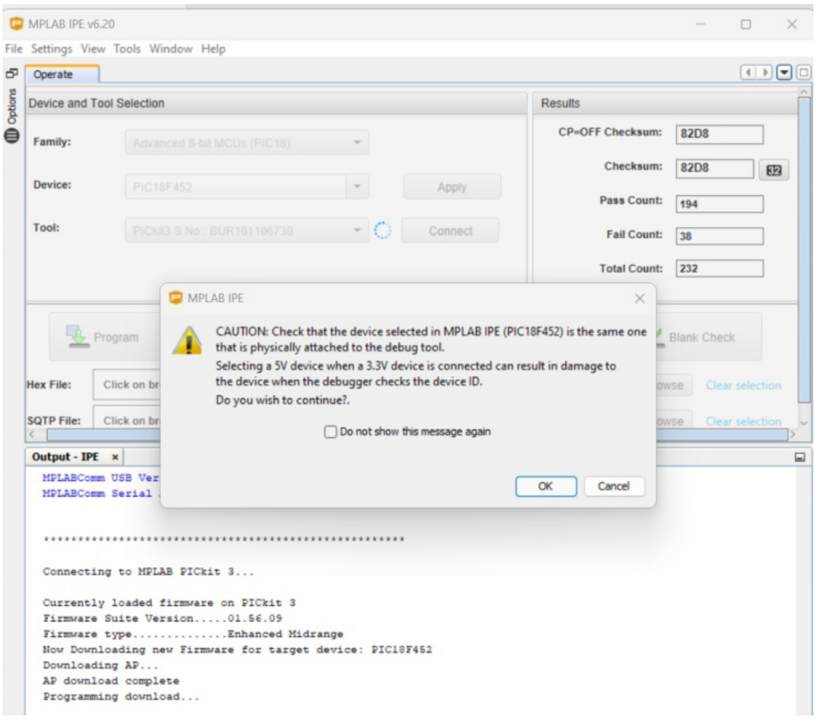
ICSP Conn & PICkit 3 Programmer.



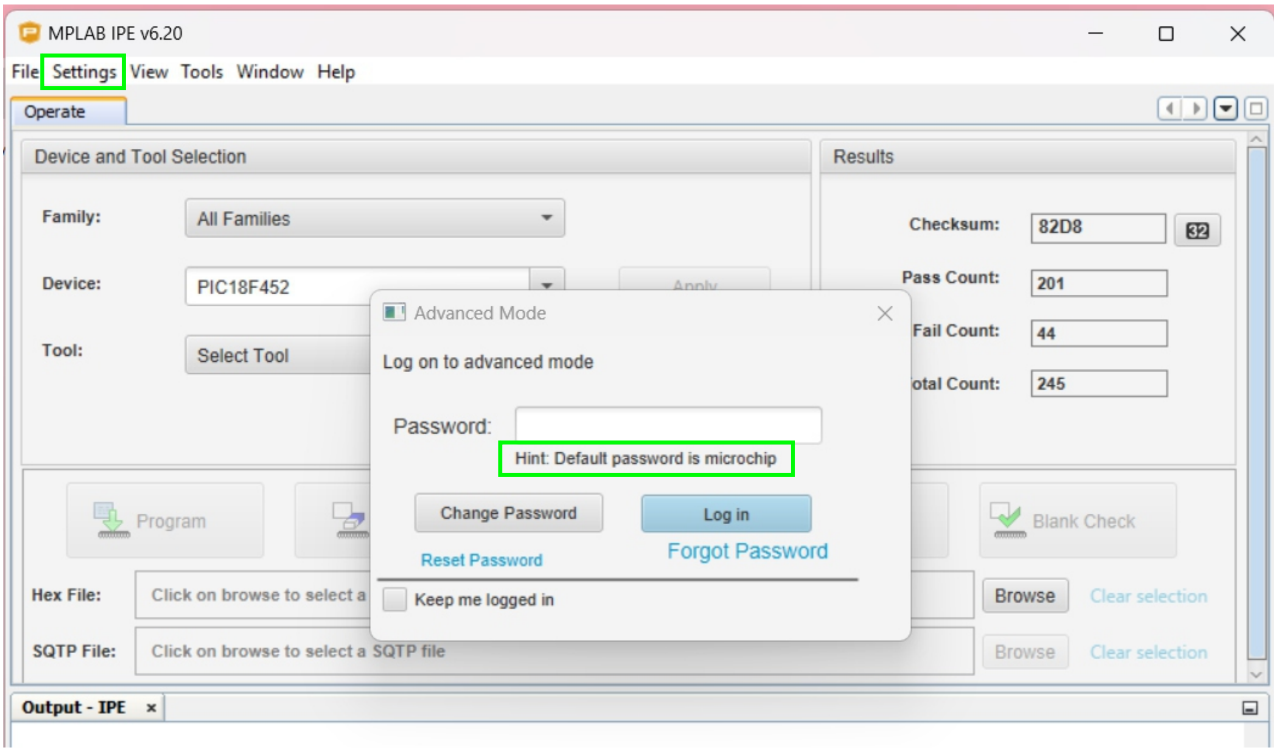
Reset Pin
of ICSP Conn. on PIC18F/16F Kit



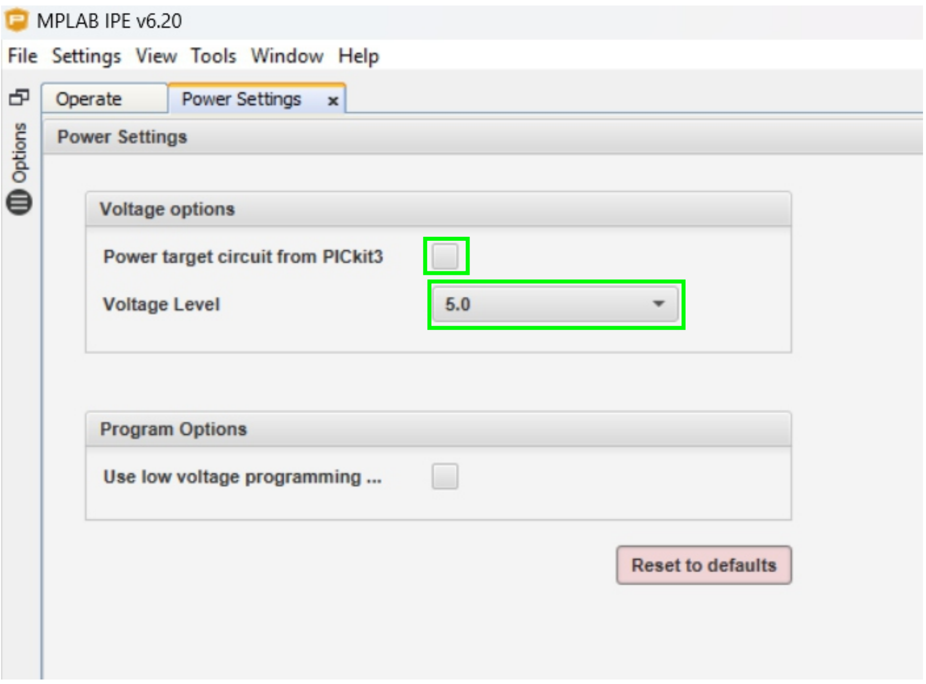
2. Once you click on Connect, below pop-up window may appear if you have not done Programming Power Source and Voltage level settings. Programming voltage and Power settings should be done before downloading the Program.



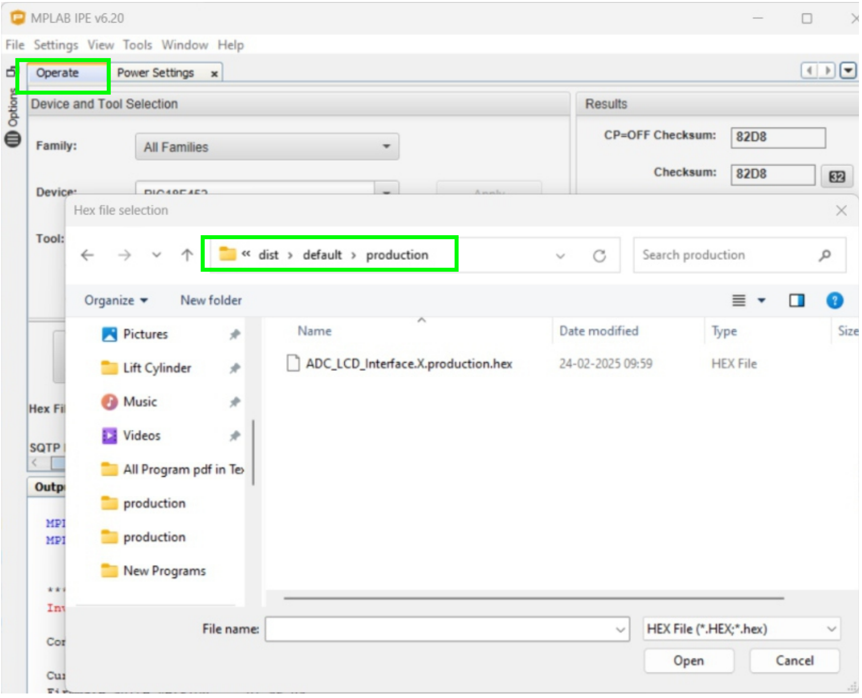
3. To confirm if correct Programming Voltage and Power Source is selected, we need to click on Setting and select Advance Mode. Then below Pop-Up window will appear. Enter the default password **‘microchip’** as given in the Hint and press Log in. Now you go in Advanced mode.



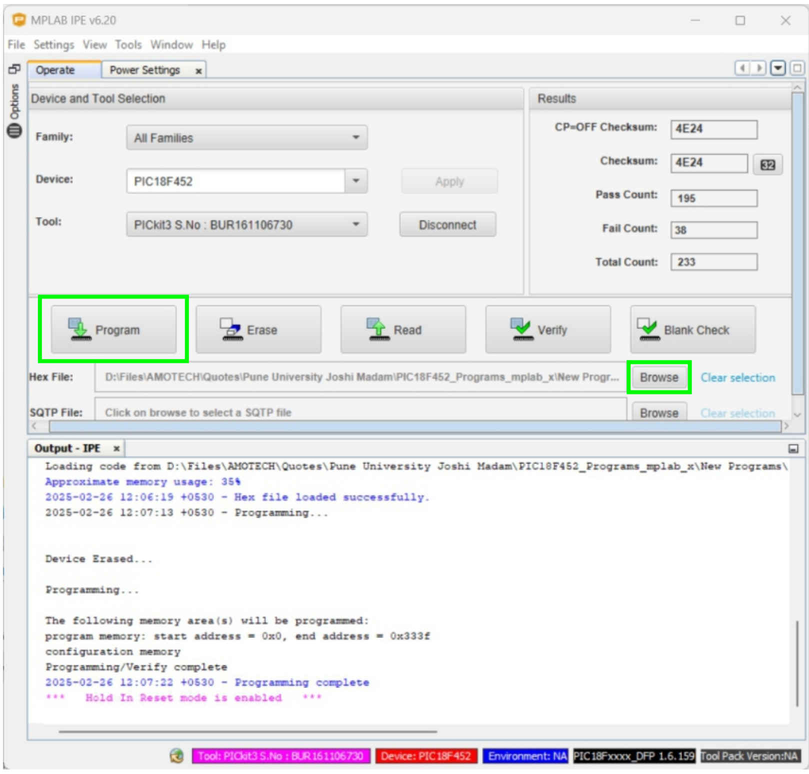
4. Once you are in Advance Mode, you will see the **‘Options’** tab on the left. Click on it and select **‘Power’** tab, then below window will appear. Un tick the below check box and select the Voltage level as **5.0** Volt. Now, must be powered by external Power Supply before downloading the program.



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.



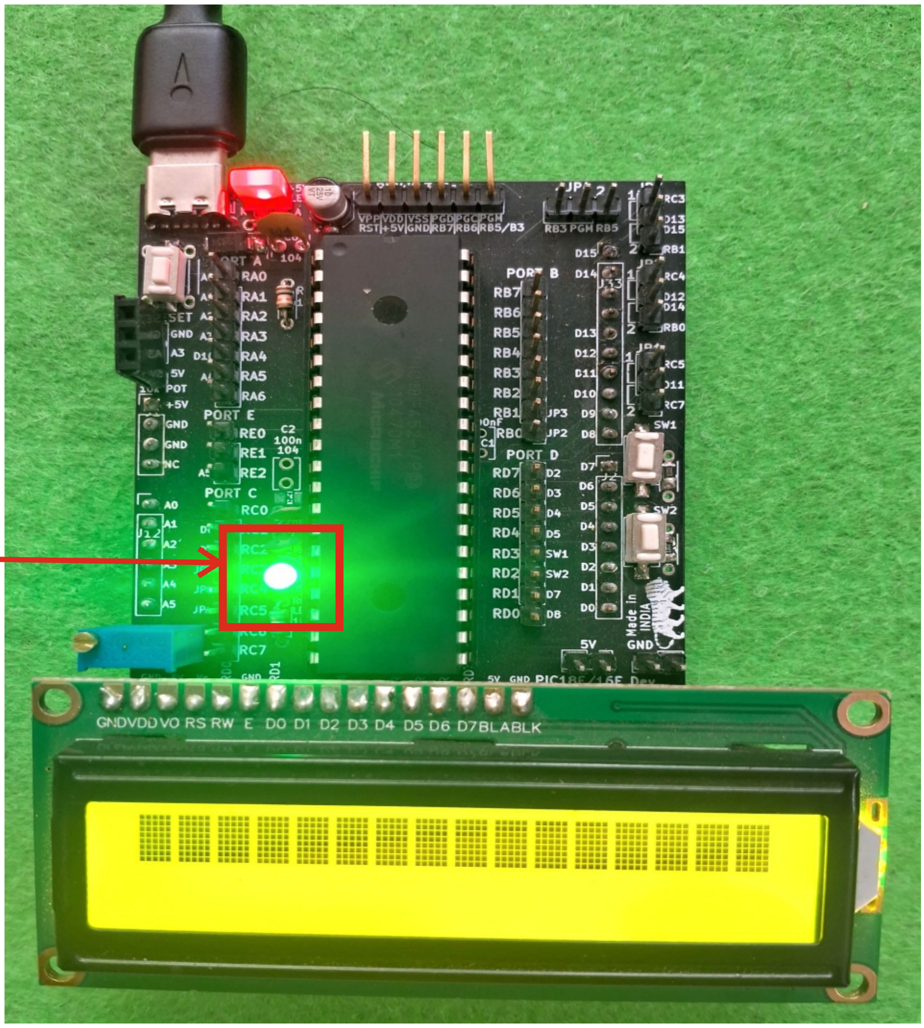
6. Once you select the Hex file you will see the notification Hex File Loaded Successfully in 'Output IPE'. Click on 'Program' button.You will see PICKit 3 LED's flashing and **'Programming...'** in Output-IPE. And, finally **'Programming complete'** once downloaded. Now remove PICKit 3 and see your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



Lab 1 - LED Blinking

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

LED



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.

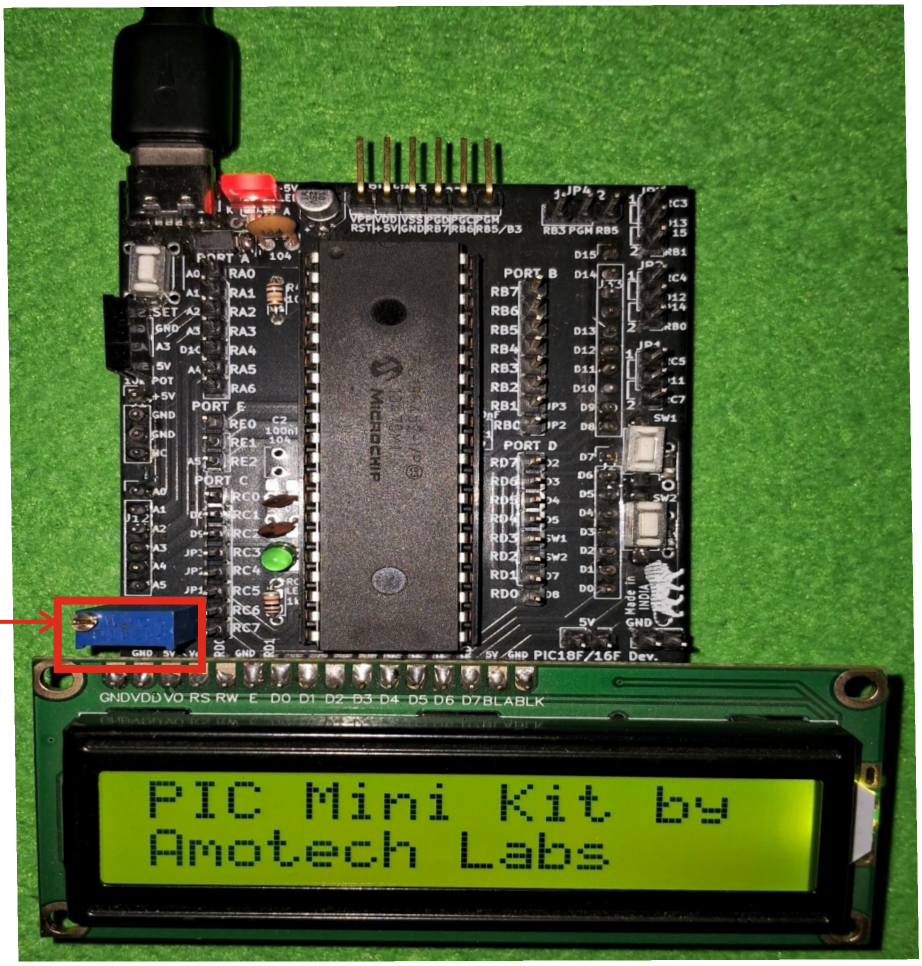
Program:

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

Lab 2 - LCD Interfacing

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

10K Pot



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 cha r cmd )
{
       RS = 0; // Command mode
```

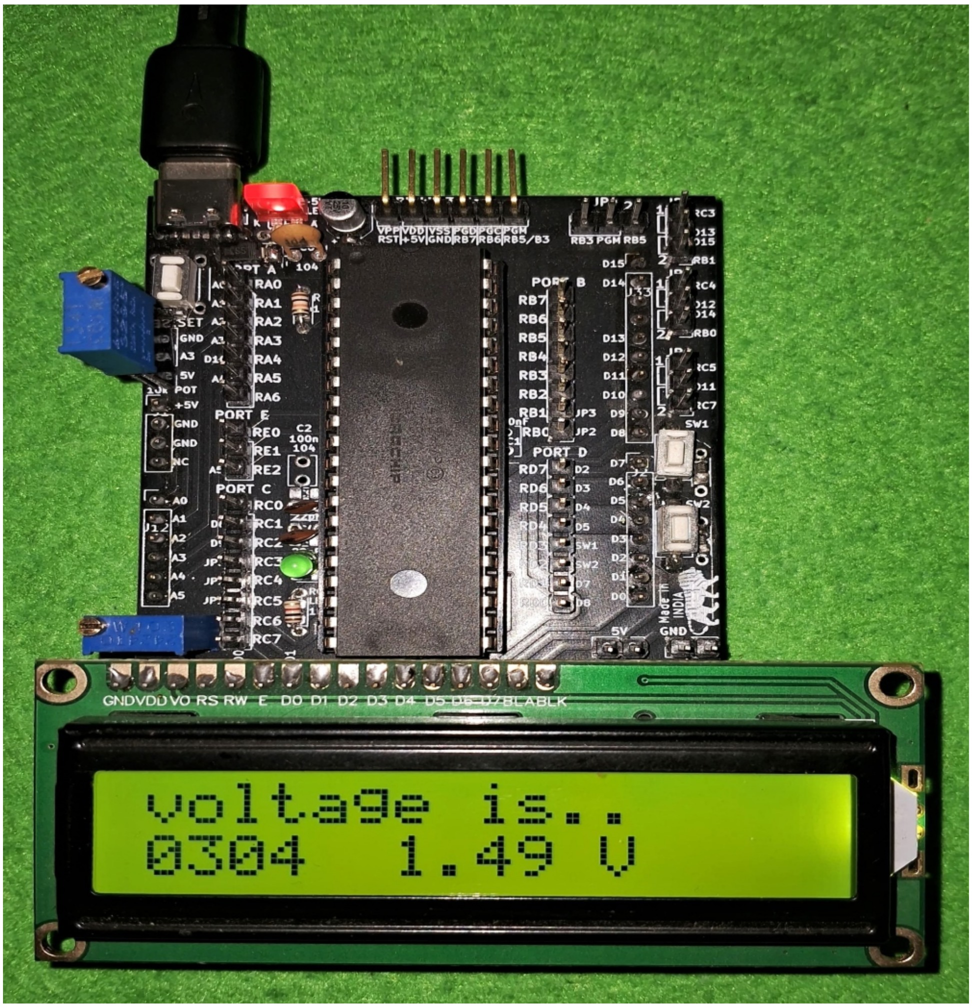
```
    D4 = (cmd >> 4) & 0x01;
    D5 = (cmd >> 5) & 0x01;
    D6 = (cmd >> 6) & 0x01;
    D7 = (cmd >> 7) & 0x01;
    EN = 1; __delay_ms(1); EN = 0; // Latch the command

    D4 = cmd & 0x01;
    D5 = (cmd >> 1) & 0x01;
    D6 = (cmd >> 2) & 0x01;
    D7 = (cmd >> 3) & 0x01;
    EN = 1; __delay_ms(1); EN = 0; // Latch the command
}
void LCD_Char(unsigned char data)
{
        RS = 1; // Data mode
    D4 = (data >> 4) & 0x01;
    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 *msg)
{    while((*msg)!=0) {
        LCD_Char(*msg);
          msg++;       }
 }
void LCD_String_xy(char row,char pos,const char *msg)  {
 char location=0;
 if(row<=1) {
 location=(0x80)|((pos)&0x0f);        /*Print message on 1st row and given column*/
 LCD_Command(location); }
 else {
  location=(0xC0)|((pos)&0x0f);        /*Print message on 2nd row and given column*/
  LCD_Command(location);
   }
  LCD_String(msg);
 }
 void LCD_Clear()
 {
    LCD_Command(0x01);  /*clear display screen*/
     __delay_ms(3);
  }
```

Lab 3 - ADC Interfacing

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



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 perMPLAB 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*/
```

```
 {                              /* Separate each digit of the integer */
  ADC_Array[k] = digital%10+'0';
  digital    = digital/10;
 }
 LCD_Command(0XC0);
 for(k=3; k>=0; k--)
 {
  LCD_Char(ADC_Array[k]);          /* Display the result on LCD */
 }
      /*It is used to convert integer value to ASCII string*/
      sprintf(data,"%.2f",voltage);

      strcat(data," V");      /*Concatenate result and unit to print*/
     LCD_String_xy(2,6,data);           /*Send string data for printing*/
      __delay_ms(250);
   }
}
void ADC_Init()
{
   TRISA = 0xff;              /*Set as input port*/
   ADRESH=0;                    /*Flush ADC output Register*/
   ADRESL=0;
}
int ADC_Read(int channel)
{
   unsigned int digital;
   ADCON0 =(ADCON0 & 0b11000111)|((channel<<3) & 0b00111000);
   /*channel 0 is selected i.e.(CHS3CHS2CHS1CHS0=0000)& ADC is disabled*/
  // ADCON0 |= ((1<<ADON)|(1<<GO));/*Enable ADC and start conversion*/
  ADCON0bits.ADON=1;
  ADCON0bits.GO_nDONE=1;
   /*wait for End of conversion i.e. Go/done'=0 conversion completed*/
   while(ADCON0bits.GO_nDONE==1);
   digital = ADRESH;
   digital <<= 8;  //Left shift the lower byte into the higher byte of tvalue
   digital += ADRESL; // Insert the TL0 byte into the lower byte of tvalue
   digital >>= 6;
   return(digital);
}
/****************************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
  D4 = (cmd >> 4) & 0x01;
  D5 = (cmd >> 5) & 0x01;
  D6 = (cmd >> 6) & 0x01;
  D7 = (cmd >> 7) & 0x01;
  EN = 1; __delay_ms(1); EN = 0; // Latch the command

  D4 = cmd & 0x01;
  D5 = (cmd >> 1) & 0x01;
  D6 = (cmd >> 2) & 0x01;
  D7 = (cmd >> 3) & 0x01;
  EN = 1; __delay_ms(1); EN = 0; // Latch the command
}
void LCD_Char(unsigned char data)
{
      RS = 1; // Data mode
  D4 = (data >> 4) & 0x01;
  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 *msg)
{
      while((*msg)!=0)
      {
        LCD_Char(*msg);
       msg++;
  }
}
void LCD_String_xy(char row,char pos,const char *msg)
{
  char location=0;
  if(row<=1)
  {
      location=(0x80) | ((pos) & 0x0f);  /*Print message on 1st row
                                          * and desired location*/
      LCD_Command(location);
  }
```

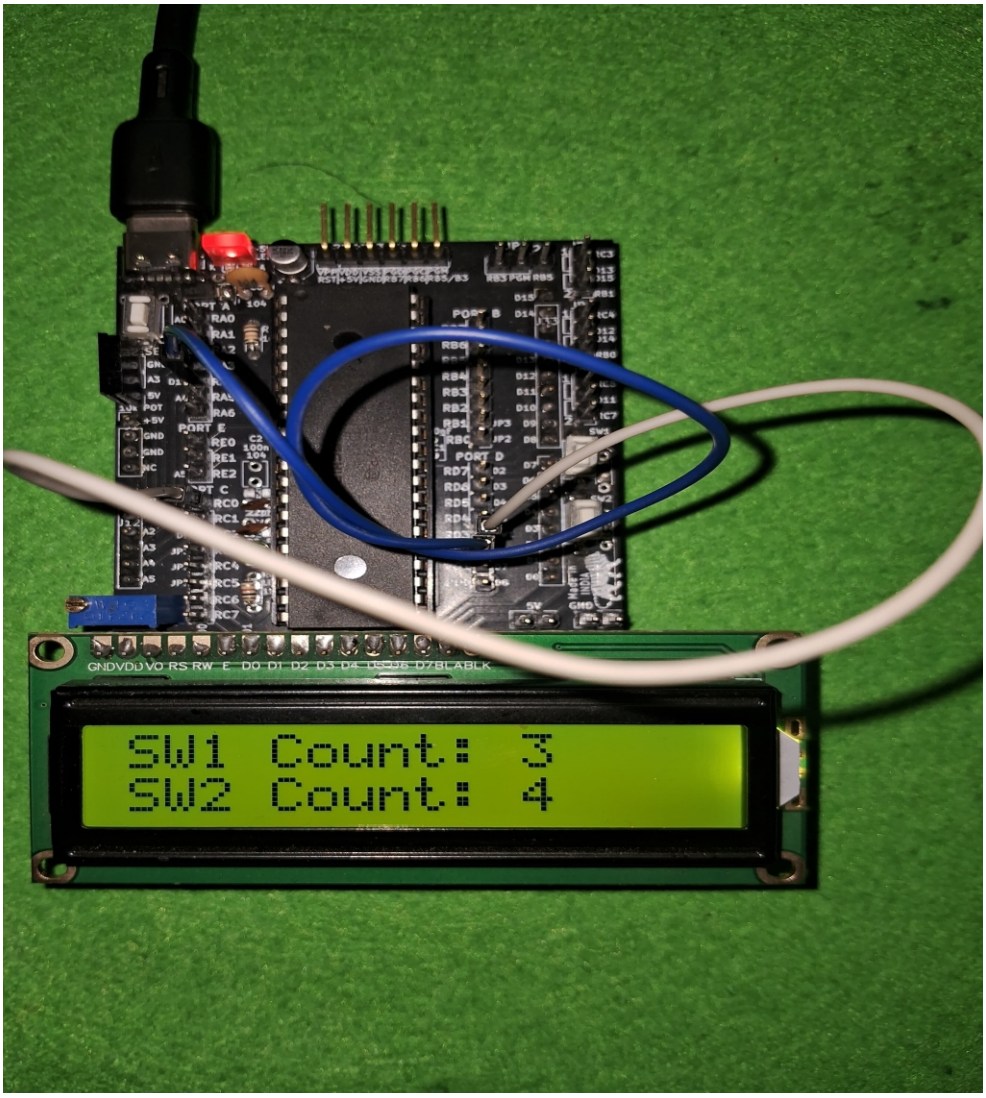
```
    else
    {
        location=(0xC0) | ((pos) & 0x0f);  /*Print message on 2nd row
                                         *  and desired location*/
        LCD_Command(location);
    }

    LCD_String(msg);
}
void LCD_Clear()
{
        LCD_Command(0x01);  /*clear display screen*/
   __delay_ms(3);
}


```

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
}
```

```
void main() {
    setup();
    LCD_Clear();
    while (1)
    {
     // Read the Timer1 counter value (pulse count)
     unsigned int t1PulseCount = (TMR1H << 8) | TMR1L;
     unsigned int t0PulseCount = (TMR0H << 8) | TMR0L;
     char t0buffer[16],t1buffer[16];
     sprintf(t1buffer, "SW1 Count: %u", t1PulseCount);          //Format the count value
     sprintf(t0buffer, "SW2 Count: %u", t0PulseCount);          //Format the count value
     // LCD_Clear();
     LCD_SetCursor(1,1);
     LCD_String(t1buffer);                                       // Display on LCD
     LCD_SetCursor(2,1);
     LCD_String(t0buffer);                                      // Display on LCD
     __delay_ms(5);                                             // Update the display every 500 ms
    }
}
void displayCount(unsigned int count) {
    char buffer[16];
    LCD_Clear();
    sprintf(buffer, "Count: %u", count); // Format the count value
    LCD_String(buffer); // Display on LCD
}
void LCD_Init(void) {
    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
    D4 = (cmd >> 4) & 0x01;
    D5 = (cmd >> 5) & 0x01;
    D6 = (cmd >> 6) & 0x01;
    D7 = (cmd >> 7) & 0x01;
    EN = 1;
     __delay_ms(1); EN = 0;                         // Latch the command
    D4 = cmd & 0x01;
    D5 = (cmd >> 1) & 0x01;
    D6 = (cmd >> 2) & 0x01;
    D7 = (cmd >> 3) & 0x01;
    EN = 1; __delay_ms(1); EN = 0;                  // Latch the command
}
void LCD_Char(unsigned char data) {
    RS = 1; // Data mode
    D4 = (data >> 4) & 0x01;
    D5 = (data >> 5) & 0x01;
```

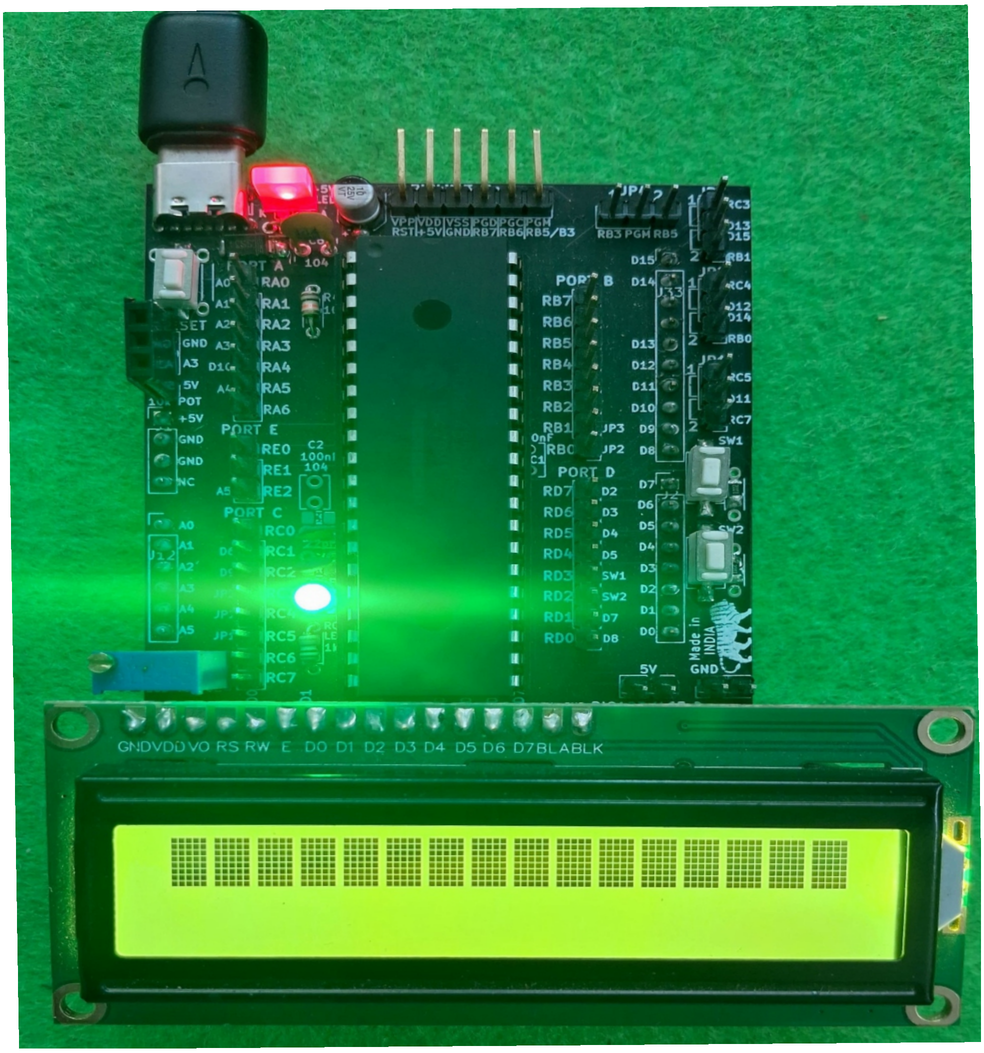
```
    D6 = (data >> 6) & 0x01;
    D7 = (data >> 7) & 0x01;
    EN = 1; __delay_ms(1); EN = 0;               // Latch the data

    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. 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:

```c
#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);
    }
}

```