

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

LAB MANUAL

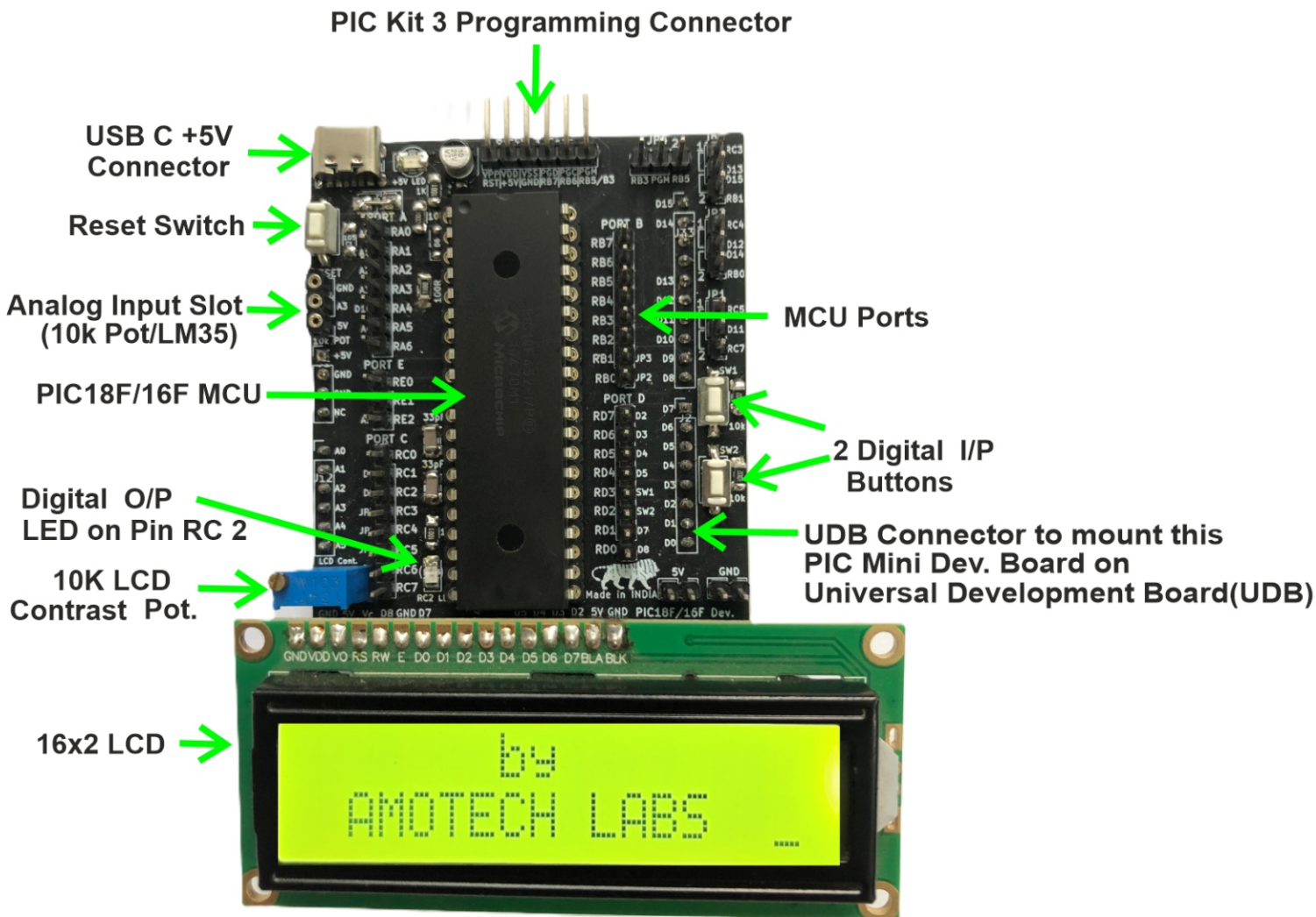
AMOTECH LABS

Embedded systems | Industrial Automation | Robotics

Contents

Overview of PIC16F/18 MCU Development Kit.....	3
Introduction to PIC18F452 Microcontroller.....	4
Schematic	5
MPLAB X IDE for Programming.....	6
MPLAB IPE for Program Downloading.....	11
Lab 1. LED Blinking	14
Lab 2. LCD_Interfacing.....	16
Lab 3. ADC interfacing.....	19
Lab 4. Pulse Counter.....	24
Lab 5. PWM Program.....	28

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. 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.

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

Universal Development Board(UDB) kit

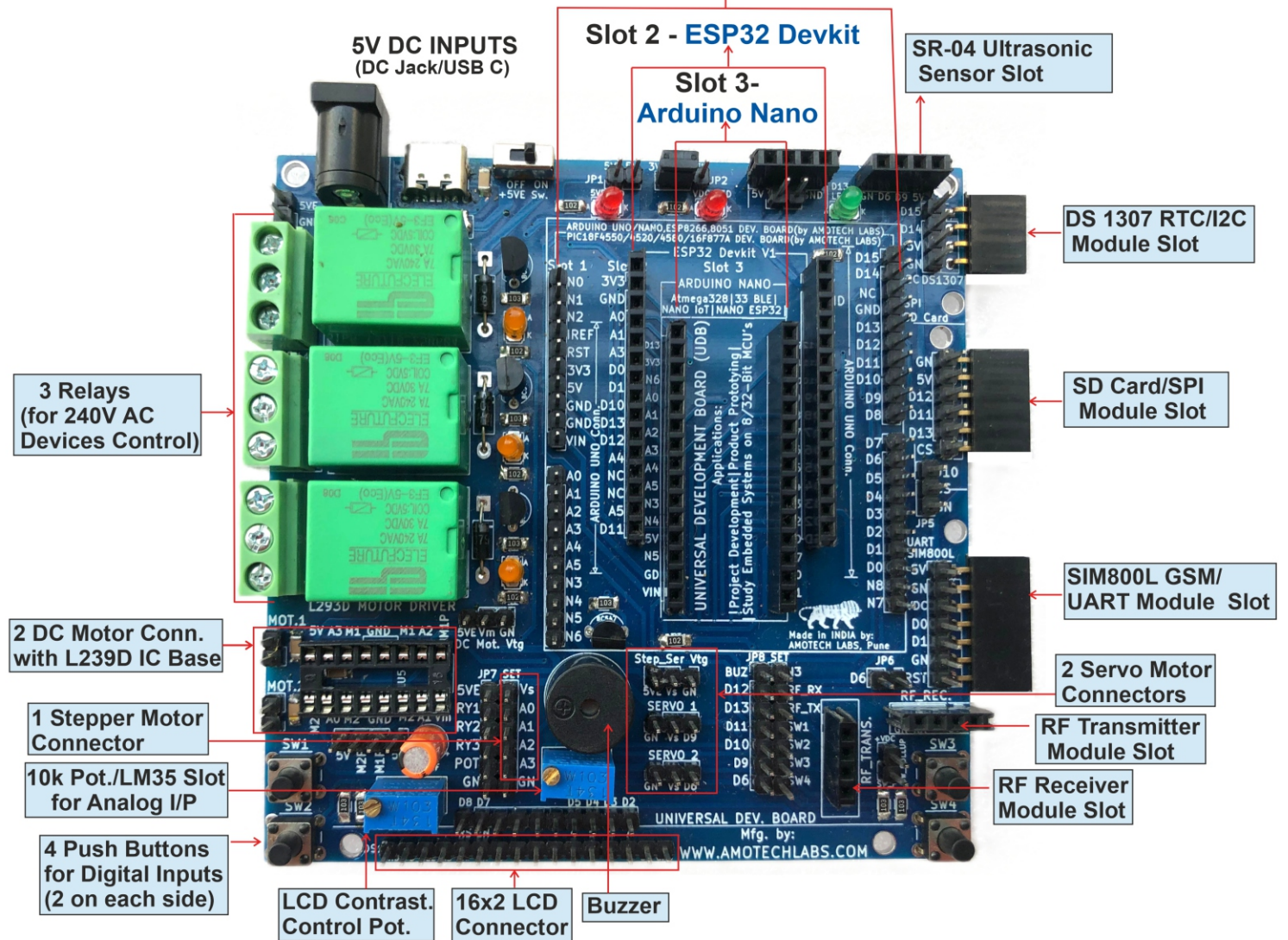
One Development for

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)

Slot 1- Arduino Uno Boards / 8051 & PIC Dev. Boards by Amotech Labs

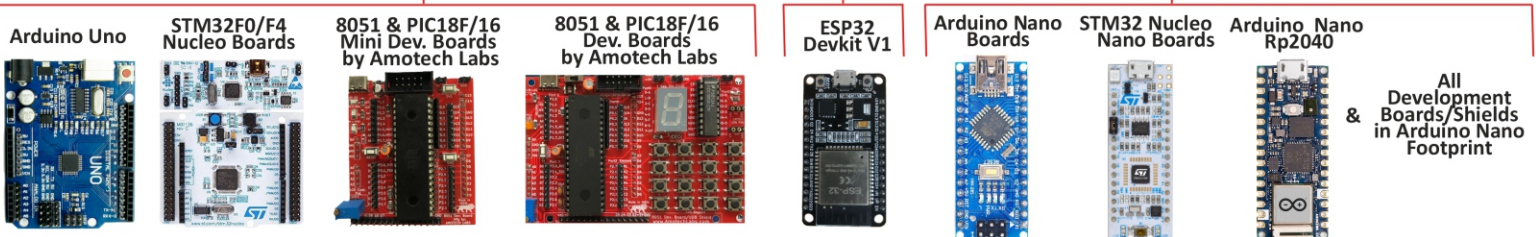


Below Micro-controller Shields/Boards can be inserted into UDB and interfaced with all above Peripherals on it

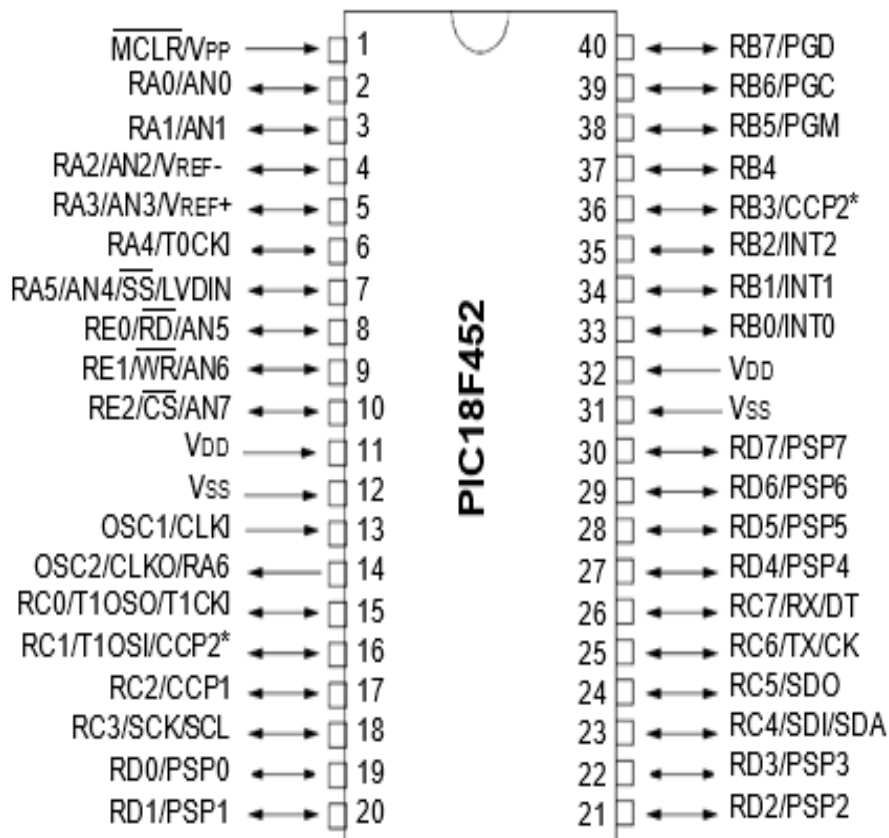
Slot 1

Slot 2

Slot 3



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

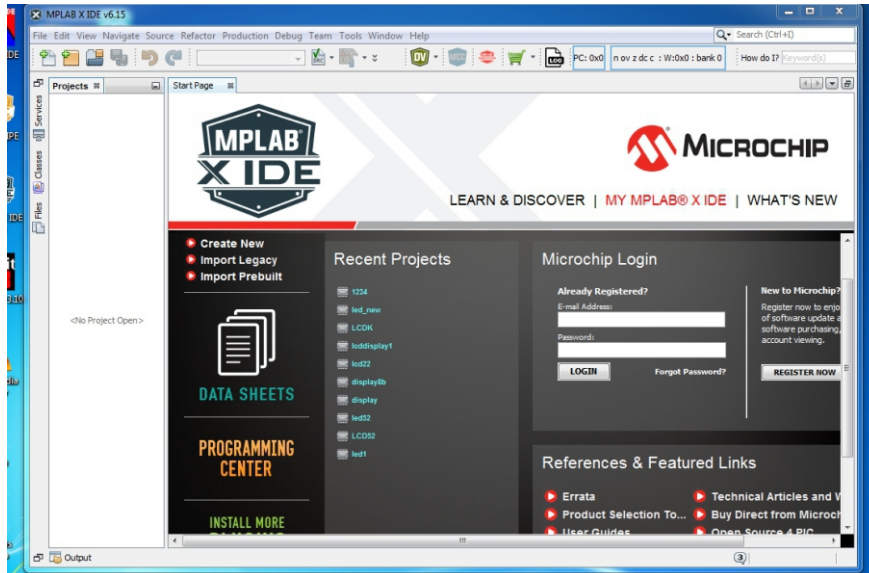
	0	5	4	3	2	1
1						
2						
3						
4						
5						



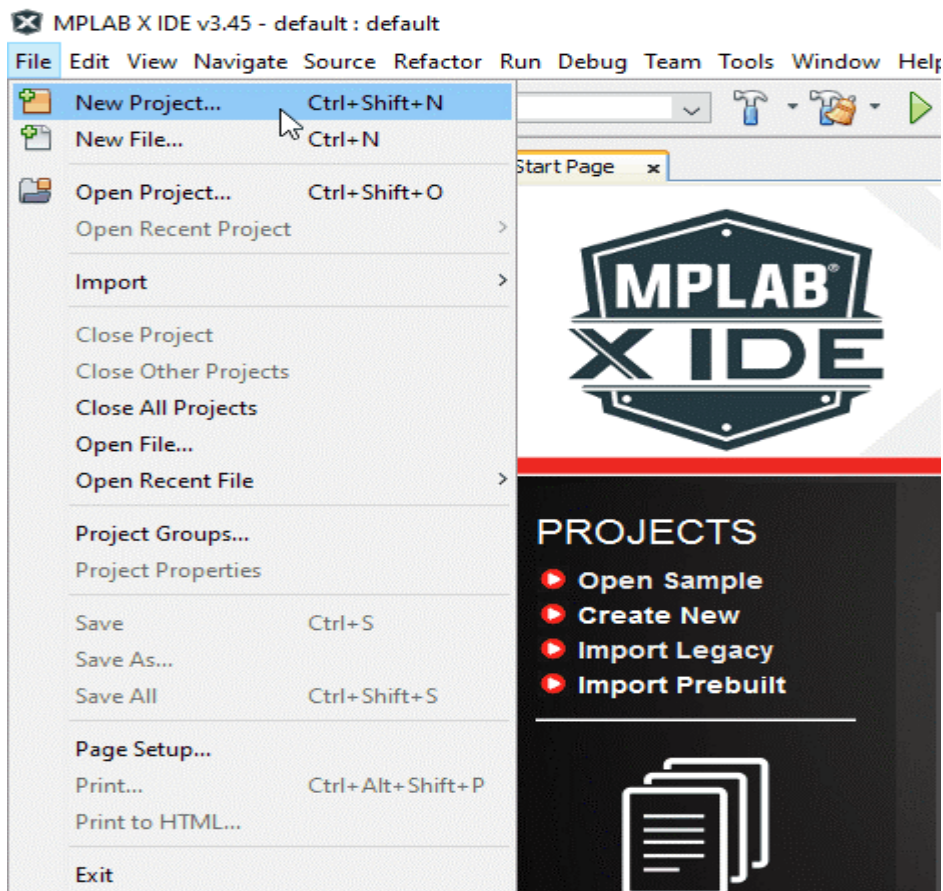
MPLAB X IDE for Programming PIC18F/16F MCU's

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

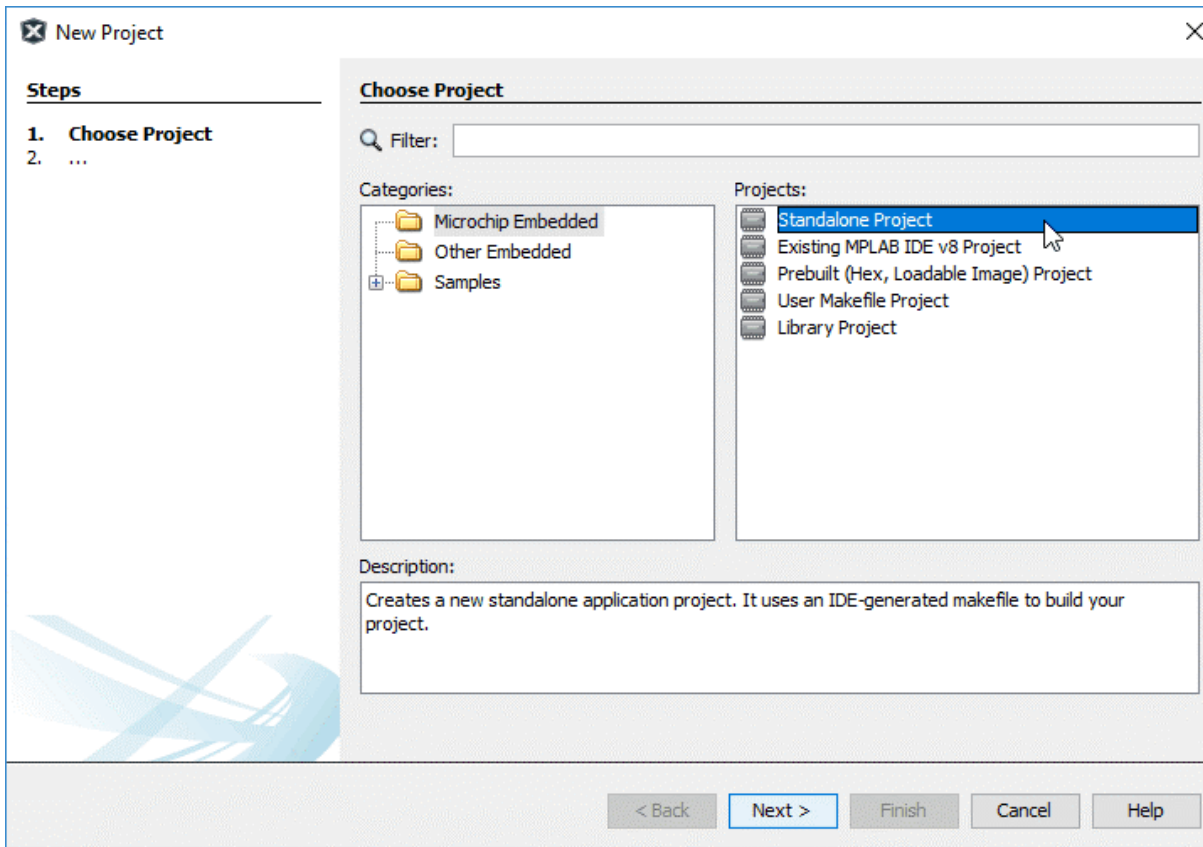
1. Double click on MPLAB X IDE icon on the desktop & Open MPLAB X IDE.



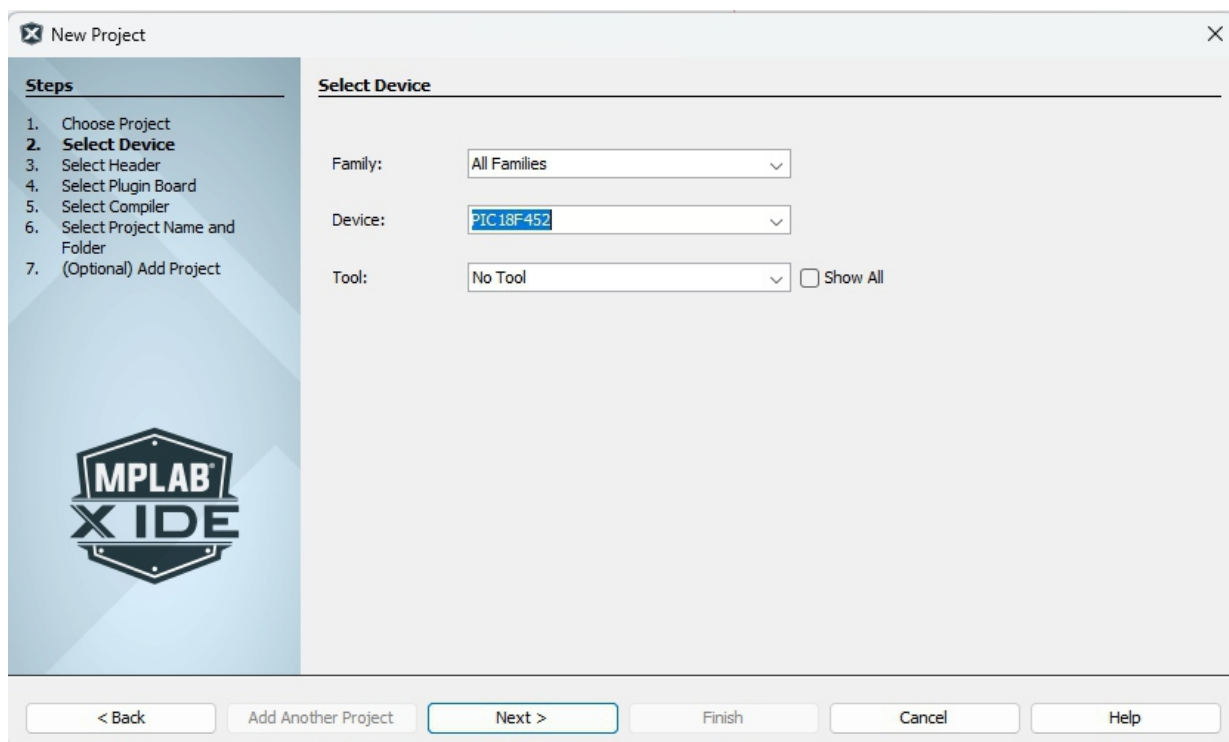
2. To create new project. Select File -> New Project.



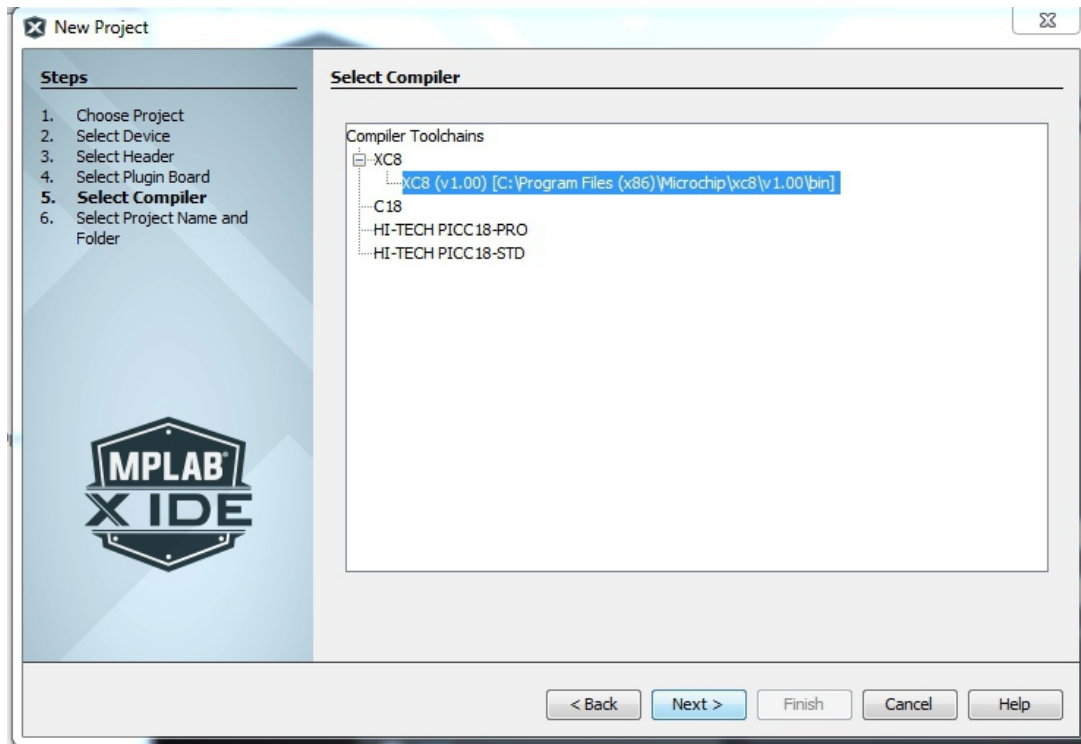
3. After that below window will appear. Select Microchip Embedded ->Standalone Project & Click next.



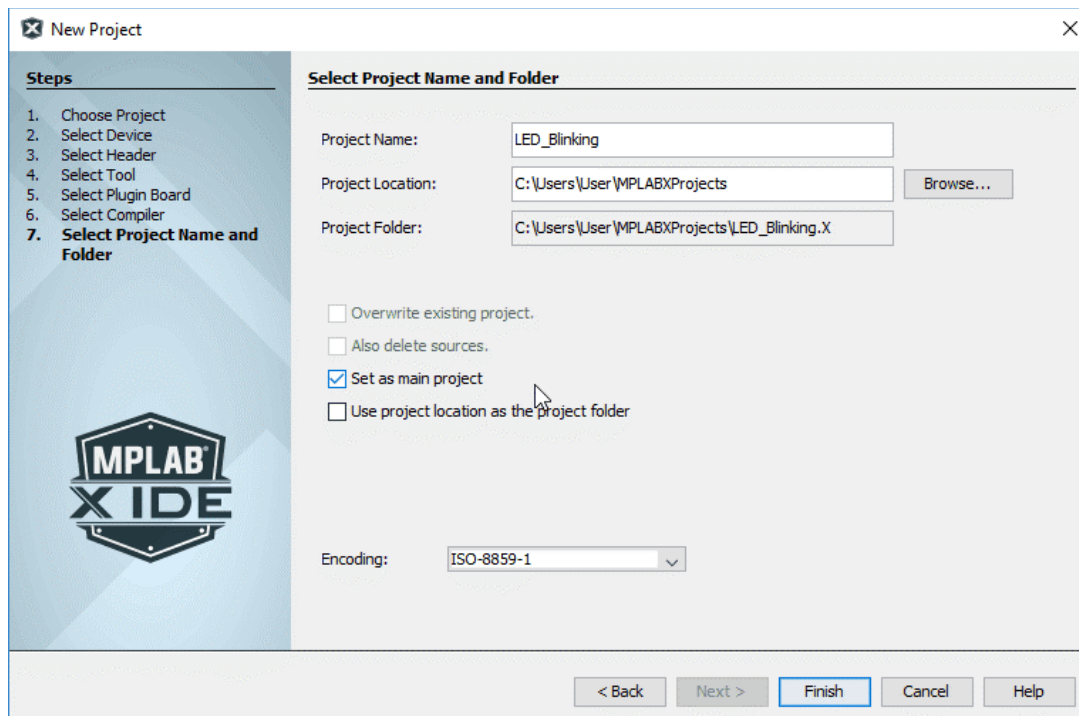
4. Select Family >Advanced 8-bit MCUs(PIC18) then select Device>PIC18F452 or any other MCU. In Tool you can select No Tool.



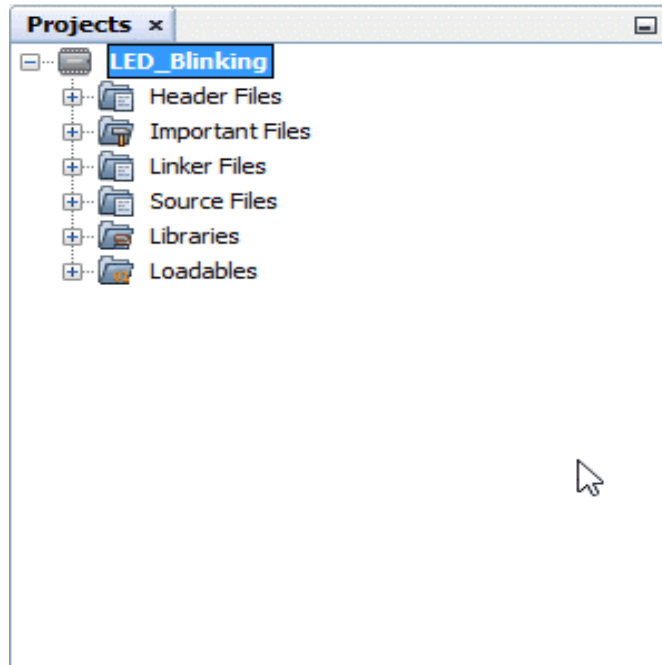
5. Now select the previously installed latest version of XC8 compiler and then click Next.



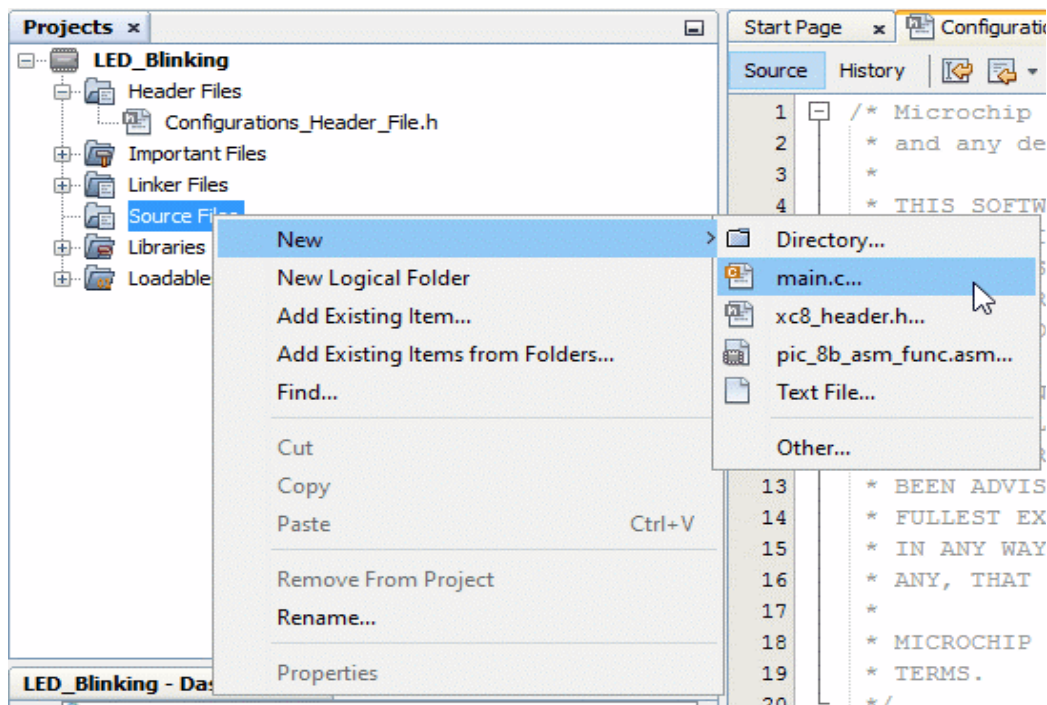
6. Enter your Project name > Click on Browse and Choose Project Folder location > Click on Finish.



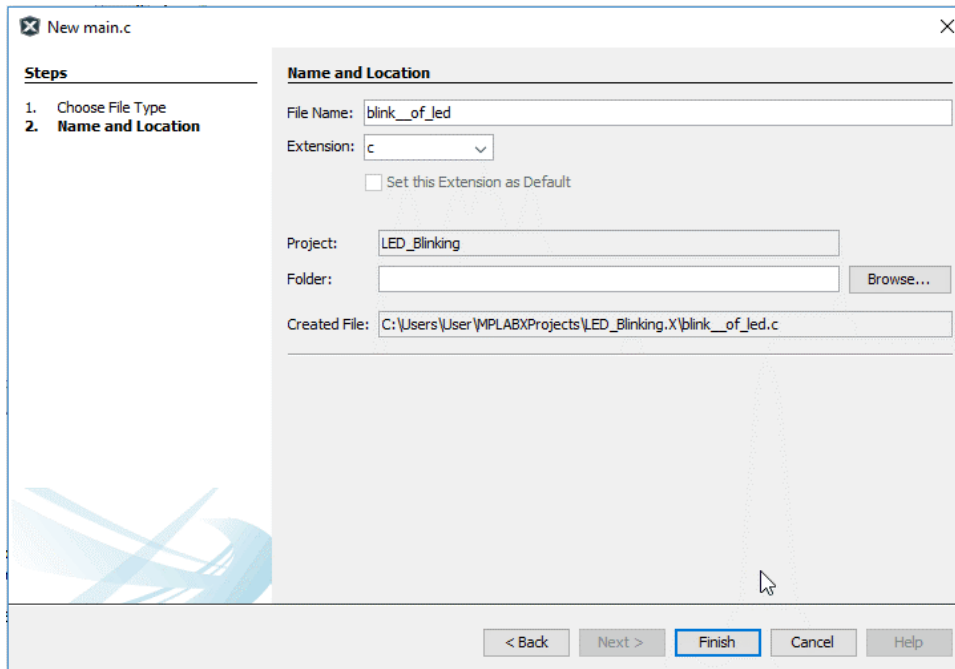
7. Now our project is displayed in Project window.



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

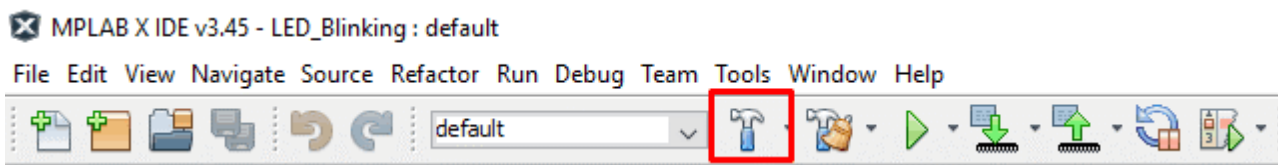


9. Give name to C file and click on finish.

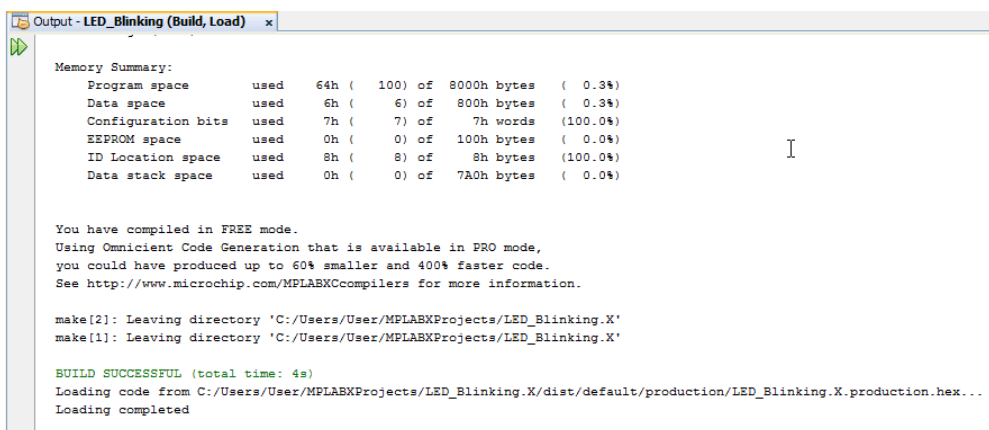


10. Write your program.

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



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



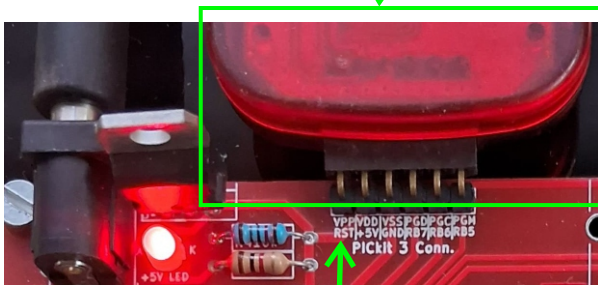
12. After BUILD SUCCESSFUL, Hex file is created in the project folder. In project folder Hex file is created in dist/default/production. You can also see a path to the folder under BUILD SUCCESSFUL.

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

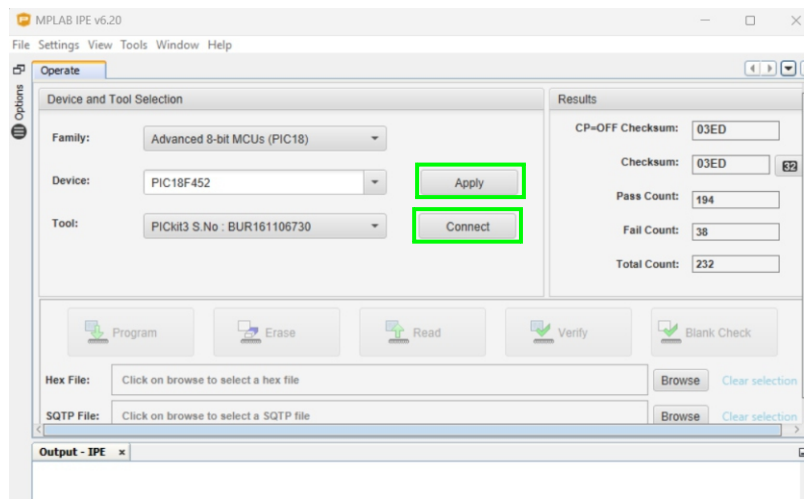
Below are the steps to program any PIC18F/16F MCU in MPLAB X IDE.

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/16F 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**'.

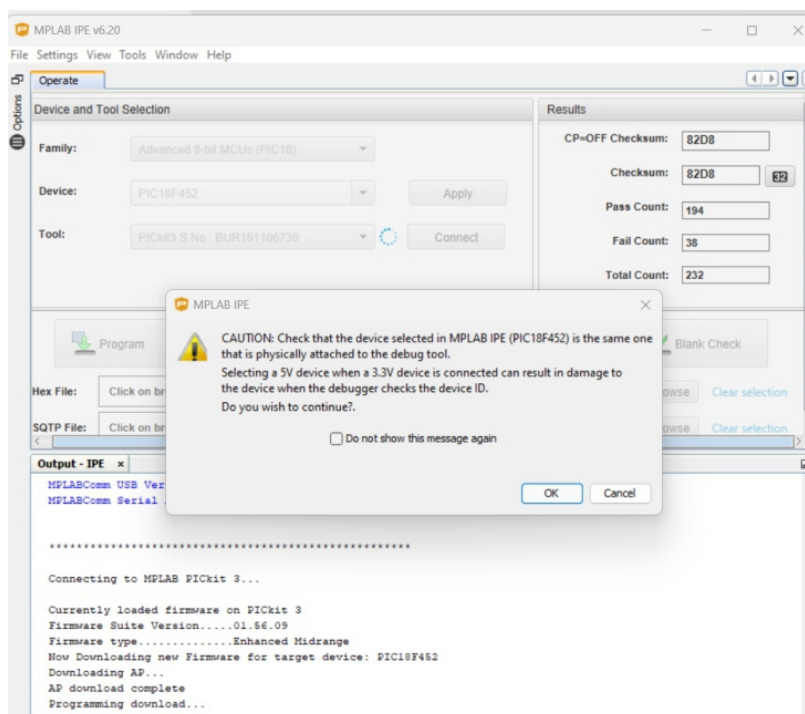
ICSP Conn & PICKit 3 Programmer.



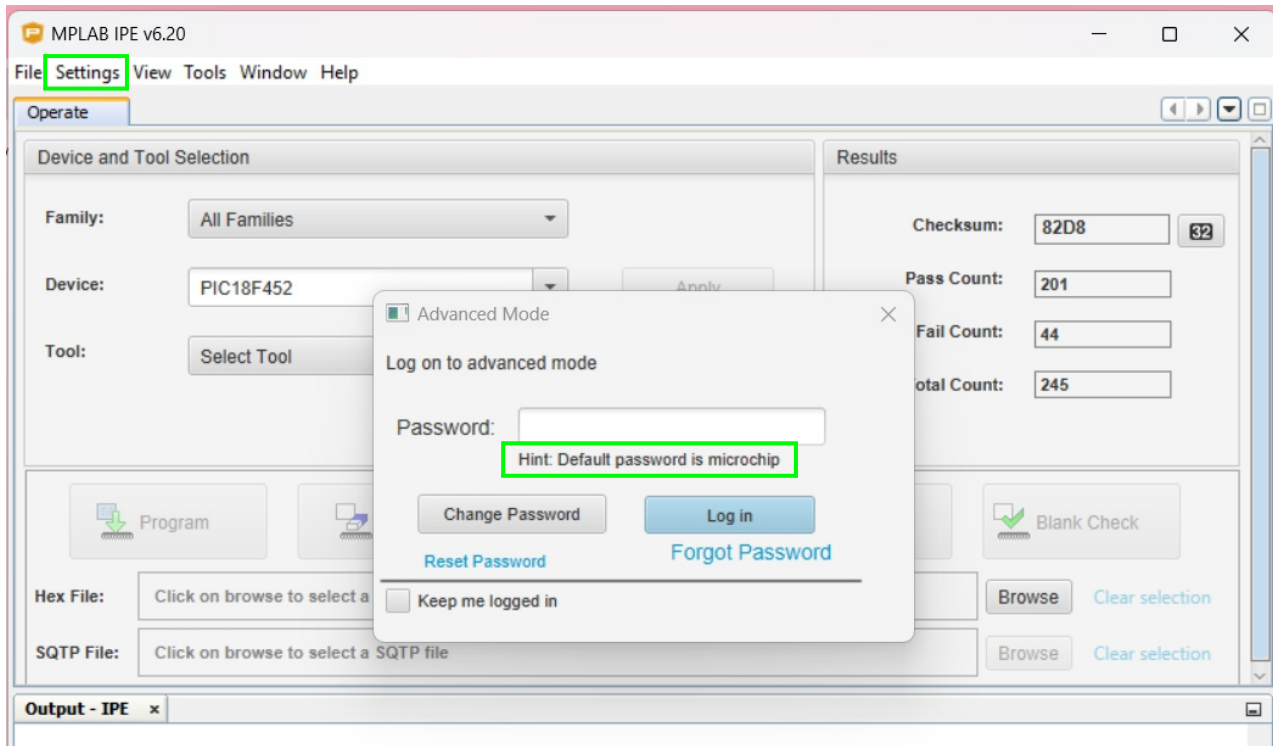
**Reset Pin
of ICSP Conn. on PIC18F/16F Kit**



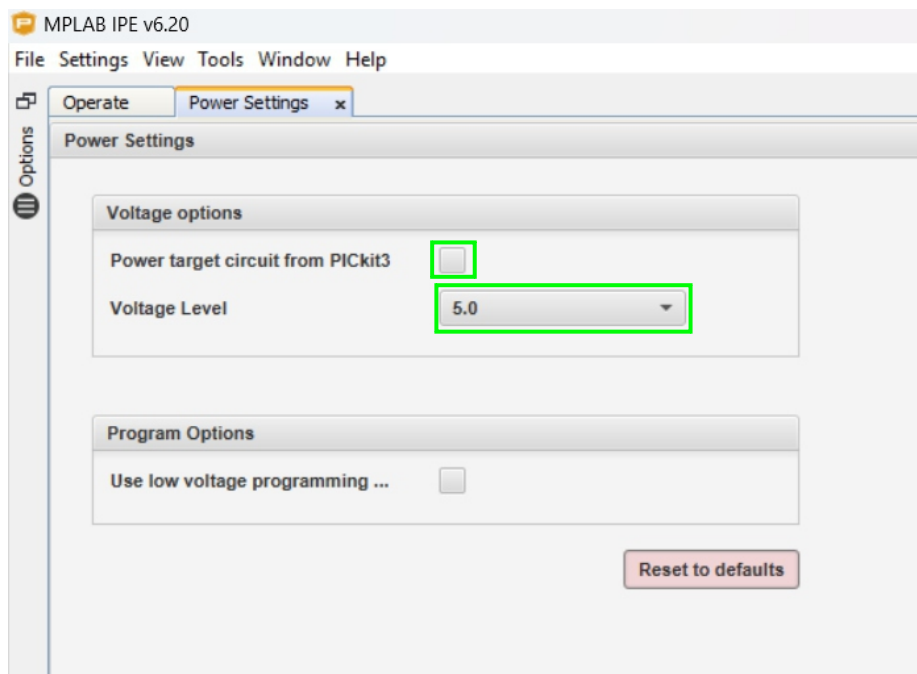
2. Once you click on Connect, below pop-up window may appear if you have not done Programming Power Source and Voltage level settings. Programming voltage and Power settings should be done before downloading the Program.



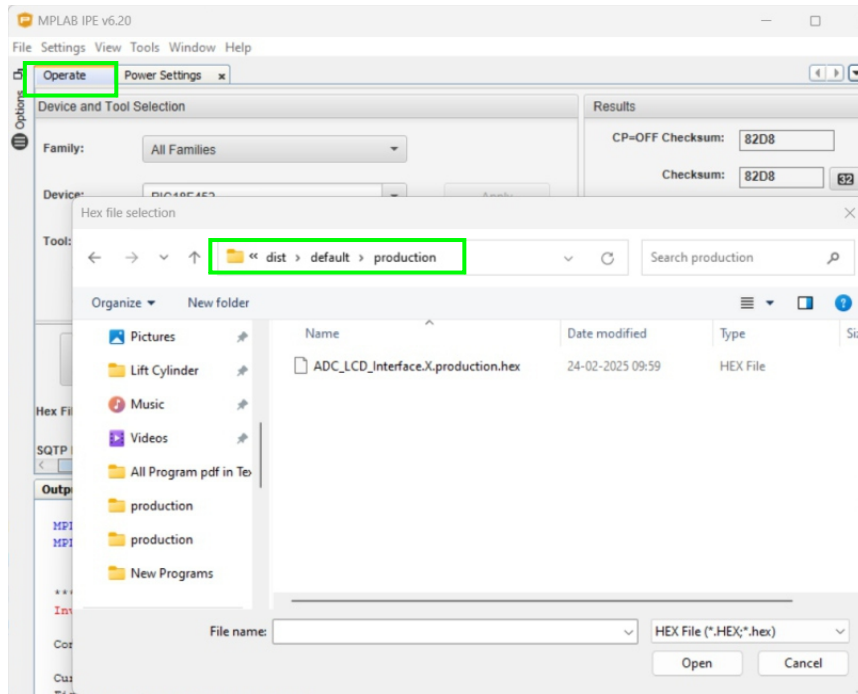
3. To confirm if correct Programming Voltage and Power Source is selected, we need to click on Setting and select Advance Mode. Then below Pop-Up window will appear. Enter the default password '**microchip**' as given in the Hint and press Log in. Now you go in Advanced mode.



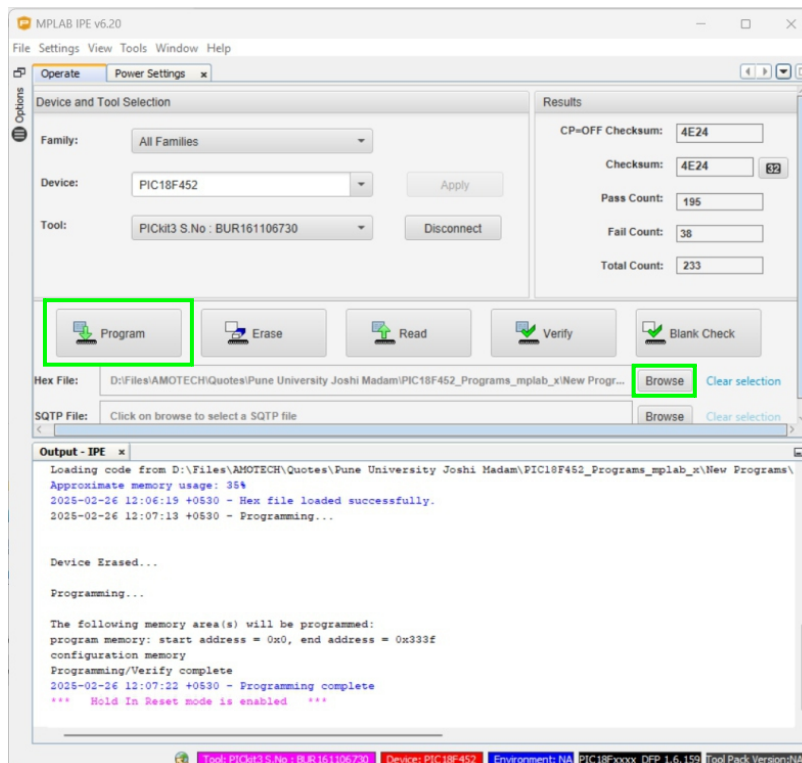
4. Once you are in Advance Mode, you will see the '**Options**' tab on the left. Click on it and select '**Power**' tab, then below window will appear. Un tick the below check box and select the Voltage level as **5.0** Volt. Now, must be powered by external Power Supply before downloading the program.



- 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.

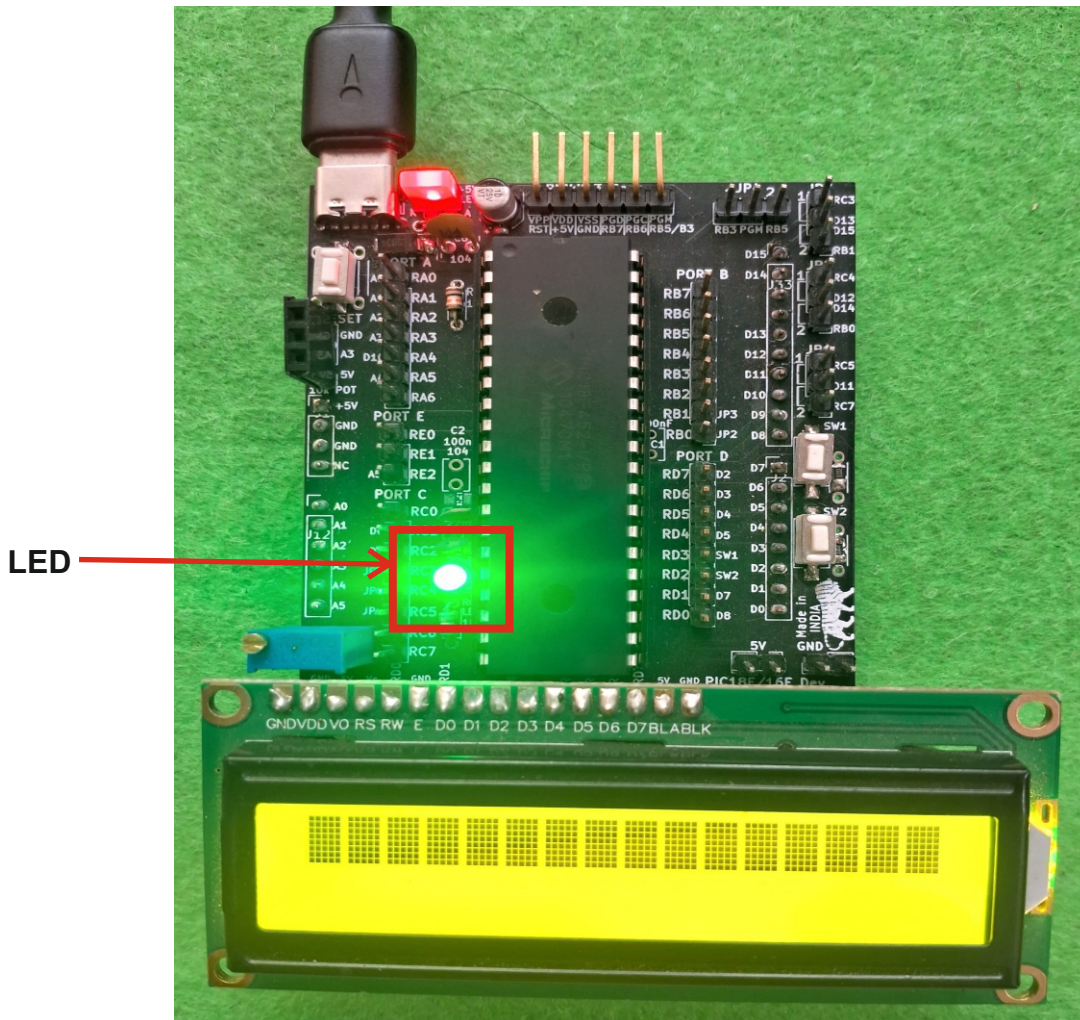


- Once you select the Hex file you will see the notification Hex File Loaded Successfully in 'Output IPE'. Click on 'Program' button. You will see PICKit 3 LED's flashing and '**Programming...**' in Output-IPE. And, finally '**Programming complete**' once downloaded. Now remove PICKit 3 and see your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



Lab 1 - LED Blinking

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



Procedure:

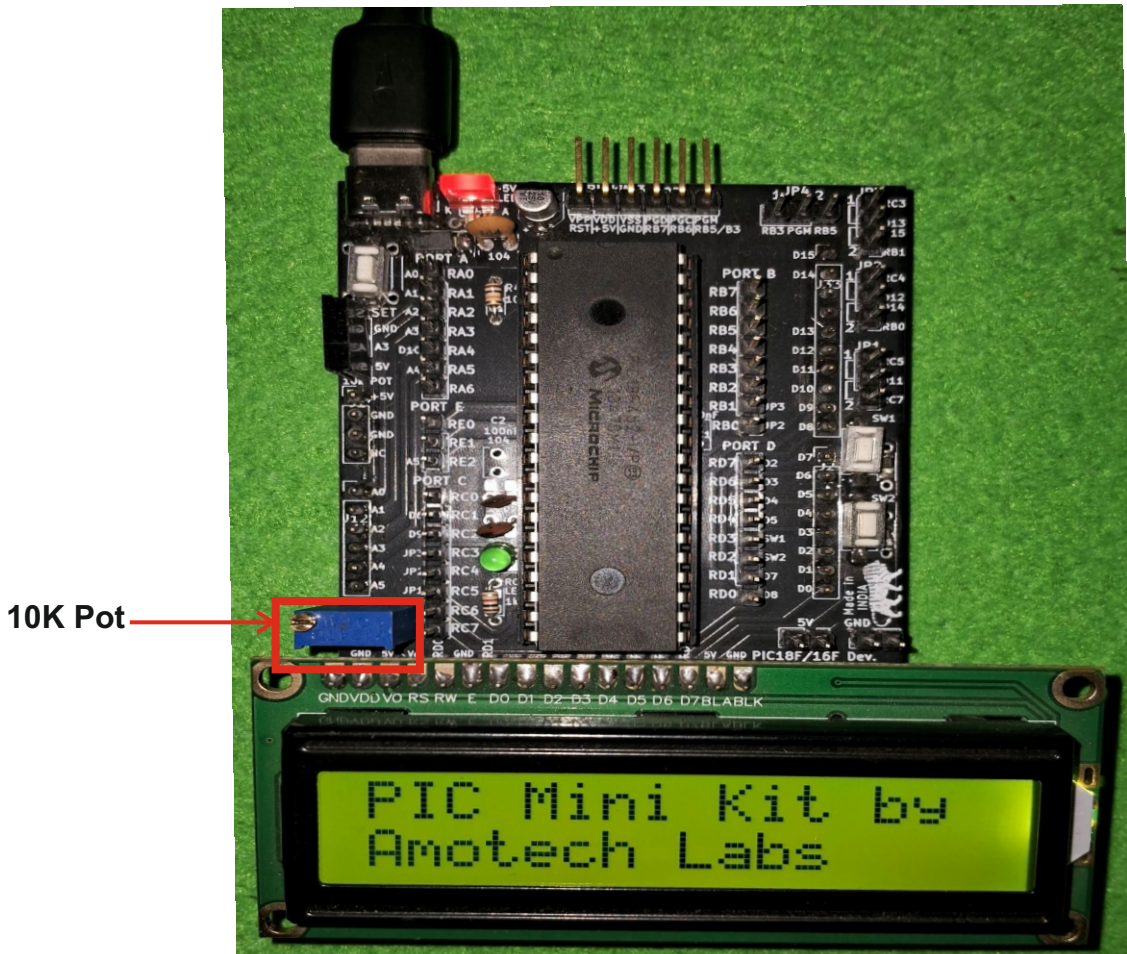
1. Read the LED Blink Program on next page with all the comments which explains the program.
2. Open the Project folder for the program in MPLAB X IDE or write it by making new project as per MPLAB X IDE procedure explained in the manual. Build and generate the Hex file for the same.
3. Open MPLAB IPE and follow the procedure explained to download the Hex file for the above program.
4. Use PICKIT3 programmer 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 void main ()
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.

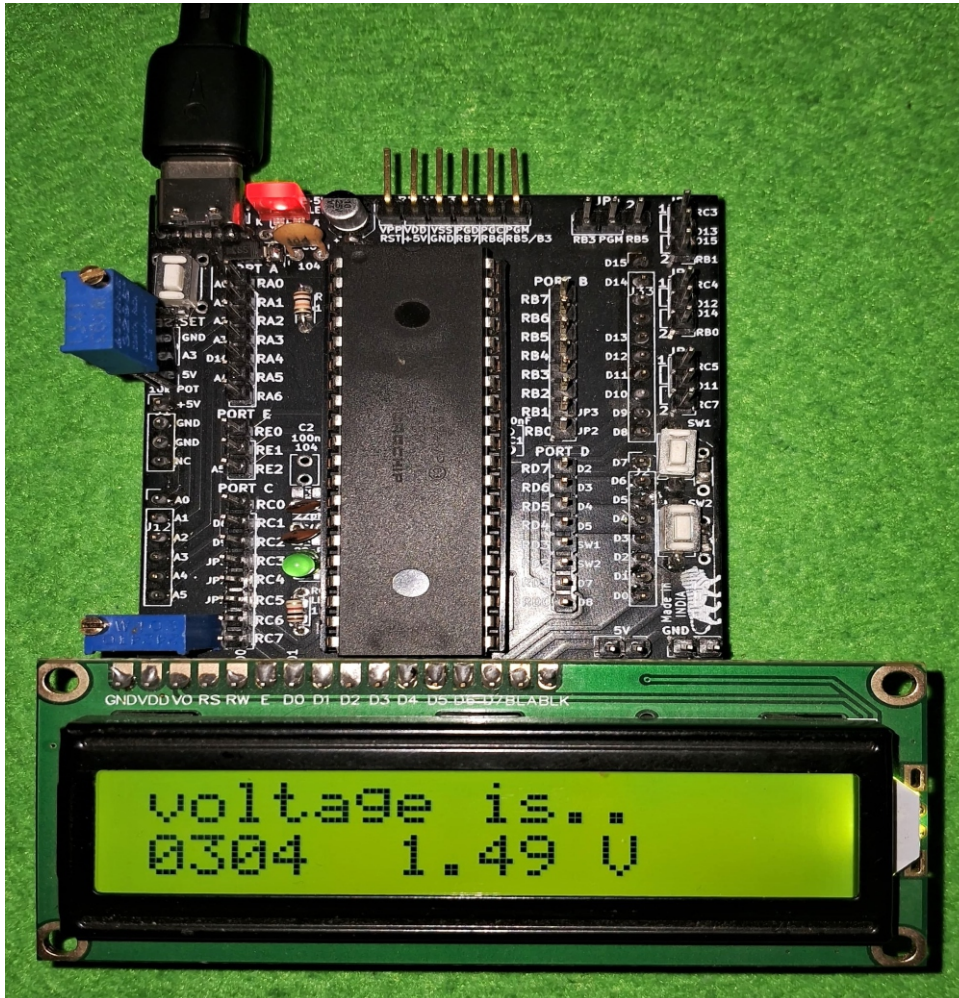
Program:

```
1 #include<xc.h>
2 #define _XTAL_FREQ 20000000 // Define the crystal frequency (8 MHz)
3 #include<pic18F452.h>
4 #include<stdio.h>
5 #include<string.h>
6 #include<stdlib.h>
7 #pragma config OSC = HS
8 #pragma config WDT = OFF
9 #pragma config LVP = OFF
10 #pragma config BOR = OFF // Brown-out Reset disabled
11 // LCD Pins
12 #define RS LATDbits.LATD0 // RS connected to RE0
13 #define EN LATDbits.LATD1 // EN connected to RE1
14 #define D4 LATDbits.LATD4 // Data line 4
15 #define D5 LATDbits.LATD5 // Data line 5
16 #define D6 LATDbits.LATD6 // Data line 6
17 #define D7 LATDbits.LATD7 // Data line 7
18 void LCD_Init(); //Initialize LCD*/
19 void LCD_Command(unsigned char ); /*Send command to LCD*/
20 void LCD_Char(unsigned char x); /*Send data to LCD*/
21 void LCD_String(const char *); /*Display data string on LCD*/
22 void LCD_String_xy(char, char , const char *);
23 void LCD_Clear(); /*Clear LCD Screen*/
24 int main(void)
25 {
26     /* OSCCON = 0x72; //Use internal oscillator and
27        // set frequency to 8 MHZ*/
28     TRISD = 0x00; // Set PORTD as output for LCD data
29     LCD_Init(); /*Initialize LCD to 5*8 matrix in 4-bit mode*/
30     LCD_String_xy(1,0,"PIC Mini Kit by"); /*Display string on 1st row*/
31     __delay_ms(1000);
32     LCD_String_xy(2,0,"Amotech Labs"); /*Display string on 2nd row,*/
33     __delay_ms(1000);
34     while(1);
35 }
36 /*****Functions*****/
37 void LCD_Init()
38 {
39     LCD_Command(0x02); // Initialize LCD in 4-bit mode
40     LCD_Command(0x28); // 2 lines, 5x7 matrix
41     LCD_Command(0x0C); // Display ON, Cursor OFF
42     LCD_Command(0x06); // Increment cursor
43     LCD_Command(0x01); // Clear display
44     __delay_ms(2);
45 }
46 void LCD_Command(unsigned char cmd )
47 {
48     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.



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 MPLAB X IDE procedure explained in the manual. Build and generate the Hex file for the same.
2. Open MPLAB IDE and follow the procedure explained to download the Hex file for the above program.
3. Use PICkit3 programmer 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:

```
1 #include<xc.h>
2 #include<pic18F452.h>
3 #include<stdio.h>
4 #include<string.h>
5 #include <stdlib.h>
6 #define _XTAL_FREQ 20000000 // Define the crystal frequency (20 MHz)
7 #pragma config OSC = HS
8 #pragma config WDT = OFF
9 #pragma config LVP = OFF
10 #pragma config BOR = OFF // Brown-out Reset disabled
11 // LCD Pins
12 #define RS LATDbits.LATD0// RS connected to RE0
13 #define EN LATDbits.LATD1// EN connected to RE1
14 #define D4 LATDbits.LATD4// Data line 4
15 #define D5 LATDbits.LATD5// Data line 5
16 #define D6 LATDbits.LATD6// Data line 6
17 #define D7 LATDbits.LATD7// Data line 7
18 /*****Proto-Type Declaration*****/
19 void LCD_Init(); /*Initialize LCD*/
20 void LCD_Command(unsigned char ); /*Send command to LCD*/
21 void LCD_Char(unsigned char x); /*Send data to LCD*/
22 void LCD_String(const char *); /*Display data string on LCD*/
23 void LCD_String_xy(char,char , const char *);
24 void LCD_Clear(); /*Clear LCD Screen*/
25 void ADC_Init();
26 int ADC_Read(int);
27 #define vref 5.00
28 int main(void)
29 {
30     char data[10];
31     TRISD = 0x00; // Set PORTD as output for LCD data
32     unsigned int digital;
33     float voltage;
34     int k;
35     char ADC_Array[5];
36     // OSCCON = 0x72; // Use internal oscillator and
37     // set frequency to 8 MHZ
38     LCD_Init(); /*Initialize LCD to 5*8 matrix in 4-bit mode*/
39     ADC_Init();
40     LCD_String_xy(1,0,"voltage is..");/*Display string on 1st row,5th location*/
41     LCD_Command(0xC0);
42
43     while(1)
44     {
45         digital=ADC_Read(3);
46         /*Convert digital value into analog voltage*/
47         voltage= digital*((float)vref/(float)1023);
48         for(k=0; k<=3; k++) /* Convert the result into ASCII*/
```

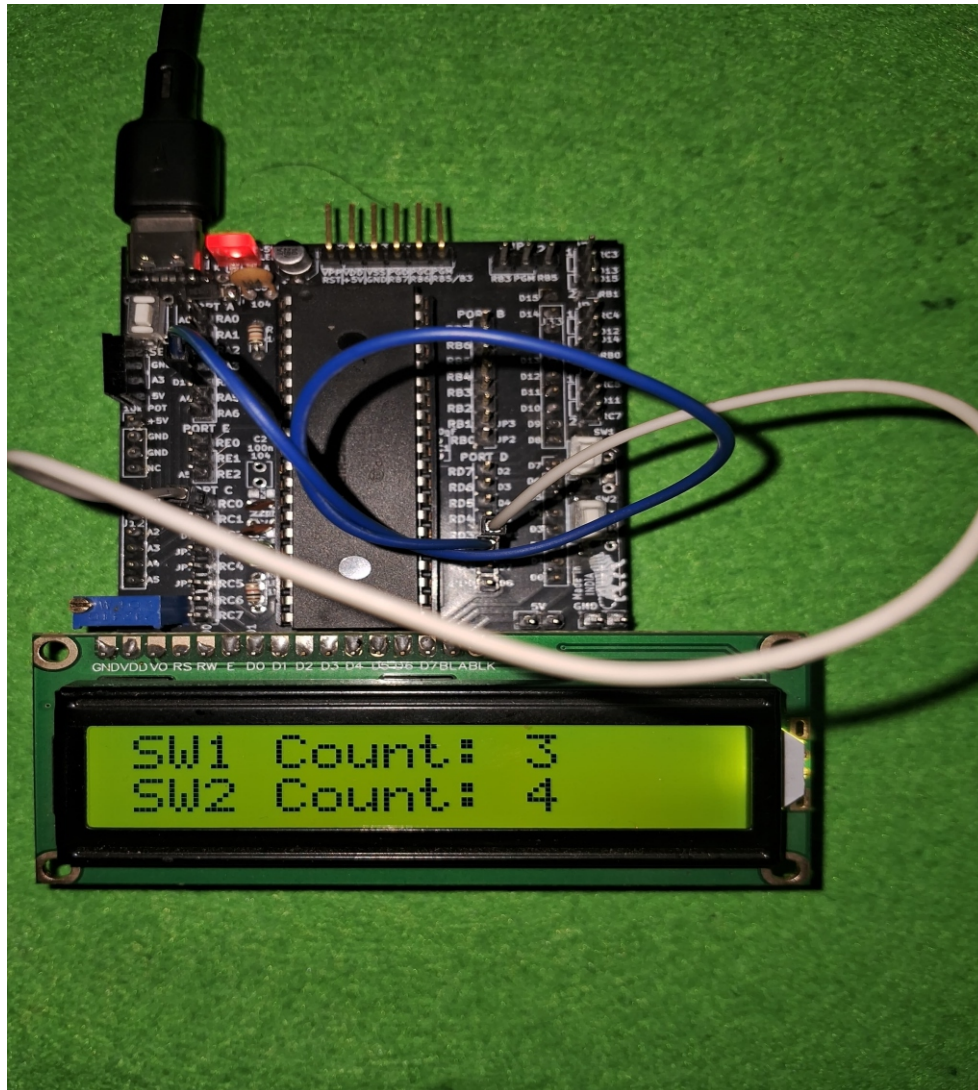
```
49 { /* Separate each digit of the integer */
50 ADC_Array[k] = digital%10+'0';
51 digital = digital/10;
52 }
53 LCD_Command(0XC0);
54 for(k=3; k>=0; k--)
55 {
56 LCD_Char(ADC_Array[k]); /* Display the result on LCD */
57 }
58 /*It is used to convert integer value to ASCII string*/
59 sprintf(data, "%.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:

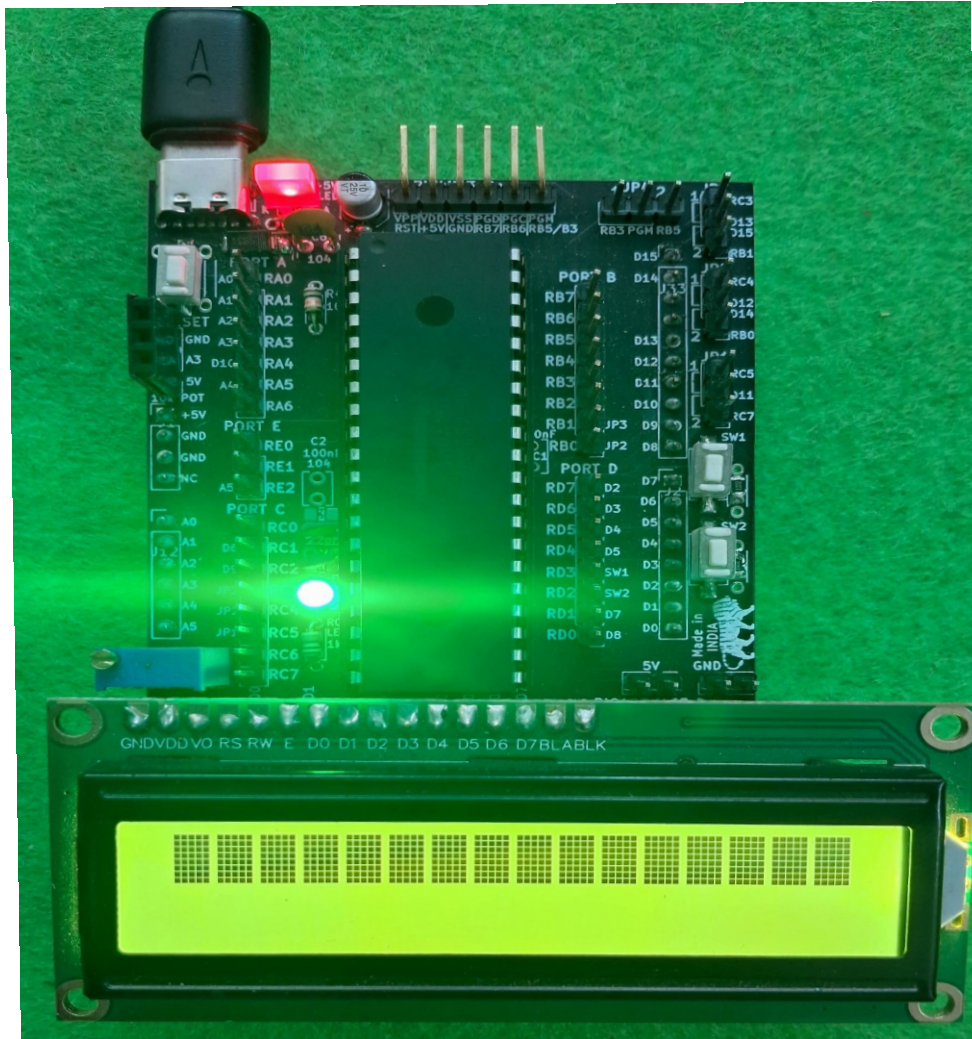
```
1 #include<xc.h>                                /*The #include <xc.h> directive is used in PIC microcontroller
2                                              programming with the MPLAB X IDE and XC8 compiler */
3 #define _XTAL_FREQ 20000000                  // Define the crystal frequency (20 MHz)
4 #include<pic18F452.h>
5 #include<string.h>
6 #include<stdio.h>                             // For printf
7 #pragma config OSC = HS
8 #pragma config WDT = OFF                     //Watchdog Timer is disabled.
9 #pragma config LVP = OFF                     /* LVP is disabled*/
10 #pragma config BOR = OFF                    // Brown-out Reset disabled
11 // LCD Pins
12 #define RS LATDbits.LATD0                   // RS connected to RE0
13 #define EN LATDbits.LATD1                   // EN connected to RE1
14 #define D4 LATDbits.LATD4                   // Data line 4
15 #define D5 LATDbits.LATD5                   // Data line 5
16 #define D6 LATDbits.LATD6                   // Data line 6
17 #define D7 LATDbits.LATD7                   // Data line 7
18 // Function prototypes
19 void LCD_Init(void);
20 void LCD_Command(unsigned char cmd);
21 void LCD_Char(unsigned char data);
22 void LCD_String(const char *str);
23 void LCD_Clear(void);
24 void LCD_SetCursor(unsigned char row, unsigned char column);
25 void setup();
26 void displayCount(unsigned int count);
27 // Global variables
28 volatile unsigned int pulseCount = 0;
29
30 void setup() {
31     // Configure Timer1 in Counter Mode
32     T1CON = 0x87;                            //Timer1 ON, External clock from RC2 (T1CKI),Prescaler 1:1
33     TRISCbits.TRISC0 = 0;                    // Set RC2/T1CKI as input
34     // Configure Timer0 in Counter Mode for Ra4
35     T0CON = 0xA8;                            // Timer0 ON, 8-bit counter mode,
36                                              //External clock source on RA4/T0CKI, no prescaler
37
38     TRISAbits.TRISA4 = 1;                    // Set RA4 as input
39     TRISD = 0x00;                            // Set PORTD as output for LCD data
40     LCD_Init();                              // Initialize the LCD
41     TMR0L = 0;                              // Clear Timer0 and Timer1 counts
42     TMR0H = 0;
43     TMR1H = 0;                              // Clear Timer1 High byte
44     TMR1L = 0;                              // Clear Timer1 Low byte
45 }
```

```
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;
```

```
96  D6 = (data >> 6) & 0x01;
97  D7 = (data >> 7) & 0x01;
98  EN = 1; __delay_ms(1); EN = 0;           // Latch the data
99
100 D4 = data & 0x01;
101 D5 = (data >> 1) & 0x01;
102 D6 = (data >> 2) & 0x01;
103 D7 = (data >> 3) & 0x01;
104 EN = 1; __delay_ms(1); EN = 0;           // Latch the data
105 }
106 void LCD_String(const char *str) {
107     while (*str) {
108         LCD_Char(*str++);
109     }
110 }
111 void LCD_Clear(void) {
112     LCD_Command(0x01);                     // Clear display
113     __delay_ms(2);
114 }
115 void LCD_SetCursor(unsigned char row, unsigned char column) {
116     unsigned char position;
117     if (row == 1) position = 0x80 + column - 1;
118     else if (row == 2) position = 0xC0 + column - 1;
119     LCD_Command(position);
120 }
121
```

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:

```
1 #include<xc.h>
2 #include<pic18F452.h>
3 #include<string.h>
4 // Configuration Bits
5 #pragma config OSC = HS      // High-speed oscillator
6 #pragma config WDT = OFF     // Watchdog Timer Disable
7 #pragma config LVP = OFF     // Low Voltage ICSP Disable
8 #pragma config BOR = OFF     // Brown-out Reset Disable
9 #define XTAL_FREQ 20000000  // Define system clock frequency (20 MHz)
10 void main()
11 {
12     unsigned int duty_cycle; /* Set CCP1 pin (RC2) as output for PWM */
13     TRISCbits.TRISC2 = 0;
14     PR2 = 124; /* Load period value in PR2 register for 100us period*/
15     CCPR1L = 1; /* Load initial duty cycle */
16     T2CON = 0x04; /* Timer2 ON, Prescaler = 4 */
17     CCP1CON = 0x0C; /* Set PWM mode */
18     TMR2 = 0; /* Clear Timer2 */
19     T2CONbits.TMR2ON = 1; /* Turn ON Timer2 */
20     while(1)
21     {
22         for(duty_cycle = 1; duty_cycle < 124; duty_cycle++)
23         {
24             CCPR1L = duty_cycle;
25             __delay_ms(20);
26         }
27         __delay_ms(500);
28
29         for(duty_cycle = 124; duty_cycle > 1; duty_cycle--)
30         {
31             CCPR1L = duty_cycle;
32             __delay_ms(20);
33         }
34         __delay_ms(500);
35     }
36 }
37
```