POLITEKNIK MALAYSIA

KEMENTERIAN PENGAJIAN TINGGI

KOLEJ KOMUNITI MALAYSIA

USER MANUAL

PIC18F458 TEACHING KIT MODULE

EMBEDDED ROBOTIC

TUAN ROZILAAZAWANI BINTI TUAN MAT

USER MANUAL

PIC18F458 TEACHING KIT MODULE

EMBEDDED ROBOTIC

TUAN ROZILAAZAWANI BINTI TUAN MAT

First Publication: September 2021
©Copyright 2021

This ebook is the original work of
Tuan Rozilaazawani binti Tuan Mat

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior permission of author/s or publisher. The author also does not guarantee that the content is suitable for the reader, but all the content is through the author's own experience and expertise.

Published by:
Petrochemical Engineering Department,
Politeknik Tun Syed Nasir Syed Ismail,
Hab Pendidikan Tinggi Pagoh,
KM1 Jalan Panchor
84600 Panchor, Muar, Johor, Malaysia

e ISBN 978-967-2736-04-2
9 7 8 9 6 7 2 7 3 6 0 4 2

PREFACE

In the name of Allah, the Almighty who gives us the enlightenment, the truth, the knowledge and with regards to Prophet Muhammad S.A.W. for guiding us to the straight path. We thank Allah S.W.T. for giving us the strength to write this book. May Allah give us the ability to continue our good deeds to the community in this field.

This book is based on the latest syllabus prepared by the Polytechnic and College Community Department, Ministry of Higher Education. The contents are comprehensive and concise to help students learn this subject more effectively in order to achieve excellent results in continuous assessment.

This book consists of six practical works; Introduction to MPLAB IDE v8.85 and C18 Compiler, Input and Output Programming in C, Circuit Simulation on PIC, PIC Programming in C (7-Segment), PIC Programming in C (DC Motor) and Hardware LCD Interfacing. Each practical work discusses the theory concept followed by hands-on skill to demonstrate basic concepts. All concepts and skills presented for each practical work are accompanied by question and answer.

Upon completion of these laboratory and exercises, you will be able to apply suitable software and hardware development on PIC18F458 microcontroller system to interface with external devices using suitable internal chip features; design embedded system application based on PIC18F458 microcontroller effectively; construct and simulate real-time embedded system application based on PIC18F458 microcontroller effectively; and demonstrate the ability to lead a team to complete assigned project or practical work within a stipulated time frame.

We are happy to receive any comments and suggestions to improve the quality of this book. We hope that this book can be beneficial to the educationist and students. InsyaAllah.

AUTHOR

Tuan Rozilaazawani binti Tuan Mat

is a Lecturer of Electrical and Instrumentation at Department of Petrochemical Engineering at Politeknik Tun Syed Nasir Syed Ismail. Her first degree in Electrical Engineering from Kolej Universiti Teknologi Tun Hussein Onn. She finished her Master in Technical and Vocational Education in 2004. She had 17 years of experience in teaching Electrical and Electronic Engineering.

CONTENT

NO.	ITEMS	NO. OF PAGES
1	Introduction	6
2	Overall System Overview	7
3	PIC18F458 Microcontroller	7 - 8
4	Hardware and Software Preparation	8 - 10
5	Components Specification	10 - 11
6	PIC18F458 Teaching Kit	11
7	Teaching Kit Operational Procedure	12
8	Laboratory	13 - 14
	▪ Practical Work 1: Introduction to MPLAB IDE v8.85 and C18 Compiler.	15 - 24
	▪ Practical Work 2: Input and Output Programming in C.	25 - 30
	▪ Practical Work 3: Circuit Simulation on PIC.	31 – 39
	▪ Practical Work 4: PIC Programming in C (7-Segment).	40 - 46
	▪ Practical Work 5: PIC Programming in C (DC Motor).	47 - 51
	▪ Practical Work 6: Hardware LCD Interfacing.	52 - 56
9	Project Exercise	57
	▪ Delay Function	58
	▪ Send Value XX to Port X	58
	▪ Door Sensor	58
	▪ Timer Interrupt	58
	▪ PortB-Change Interrupt (RB4-RB5)	59
	▪ External Hardware Interrupt	59
	▪ Keypad	59
10	References	60

1.0 Introduction

Microcontroller technology that focuses on giving general knowledge on the varieties of the existing microcontrollers, the differences between these, and more theoretical knowledge about microcontroller principles. The innovative education based on microcontroller merges the innovative theory, innovative system, specialty knowledge, practical abilities and project design. The availability of multiple processing elements and large amounts of memory on a single chip will make it possible to create sophisticated multiprocessors for embedded applications.

Microchip C18 development environment, ensuring that students would program their algorithm using C and be able to compile this programming into the processor machine code. Teams were encouraged to independently add new sensors and adapt the kit while developing the original algorithm. It is designed to assist teaching many practical engineering skills that may often be left uncovered; innovation, design, knowledge integration and the ‘real’ problems of ‘real’ system. Educating our current engineering students in the best practices for real-time and embedded systems development is of great importance.

The embedded system course is application-oriented, learning embedded system is not only helpful for students to familiar with the methodology of system design, but also improve students’ ability to master and apply correlative knowledge. For the teaching of embedded systems, experiment is very important, and it is the key way for students to master the technique of embedded system design. For the students to develop the skills they need to design and program simple microcontroller-based projects while providing the working knowledge sufficient to be applied to more complex project systems in subsequent coursework.

2.0 Overall System Overview

You must be familiar with this manual in order to complete the exercises given within. With all these exercises, you will learn about:

i. MPLAB IDE v8.85 and C18 Compiler
ii. Input and Output Programming in C
iii. Circuit Simulation on PIC
iv. PIC Programming in C (7-Segment)
v. PIC Programming in C (DC Motor)
vi. Hardware LCD Interfacing

Upon completion of these laboratory and exercises, you will be able to apply suitable software and hardware development on PIC18F458 microcontroller system to interface with external devices using suitable internal chip features; design embedded system application based on PIC18F458 microcontroller effectively; construct and simulate real-time embedded system application based on PIC18F458 microcontroller effectively; and demonstrate the ability to lead a team to complete assigned project or practical work within a stipulated time frame.

3.0 PIC18F458 Microcontroller

These devices are available in 40-pin and 44-pin packages. PIC18F458 devices have twice the Flash program memory 32 Kbytes and data RAM 1536 bytes.

Features of the device:

- 256 bytes of EEPROM data memory
- 21 Interrupt Sources
- 2 Analog Comparators
- 8 input channels of 10-bit Analog-to-Digital Converter
- 4 Timers
- 2 Capture/Compare/PWM Modules.
- Serial Communications for MSSP, CAN and Addressable USART
- Self-programming

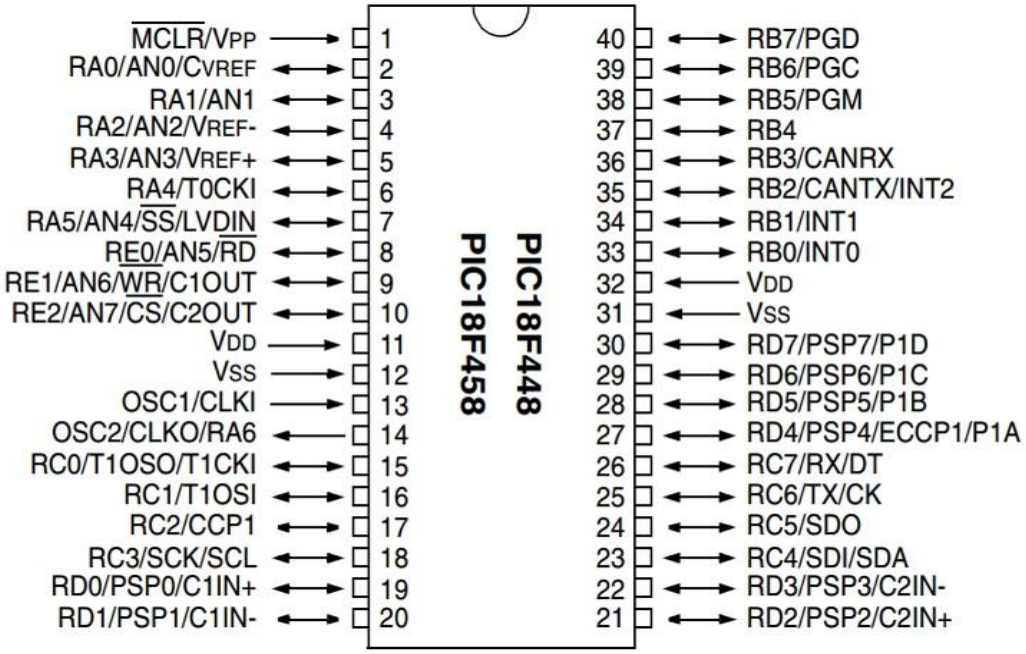


Figure 1: PIC18F458 Pin Diagrams

Figure 1 shows the pin diagram for PIC18F458. For more information about the PIC microcontroller, please refer to the datasheet. The data sheet can be found in Microchip website at http://www.microchip.com.

4.0 Hardware and Software Preparation

This PIC18F458 Teaching Kit Module Embedded Robotic covers the basic concept and application of microcontroller systems based on the Peripheral Interface Controller (PIC) microcontroller. Users will learn software and hardware development on PIC18F458 microcontroller development system such as MPLAB IDE and understand how to do interfacing with external devices using suitable internal chip features. Users are also exposed to the new Microcontroller Unit (MCU) simulation software such as Proteus.

Hardware Setup

The basic hardware in this teaching kit are target board, USB programmer and USB cable. Basic hardware connection to a computer as shown in Figure 2.

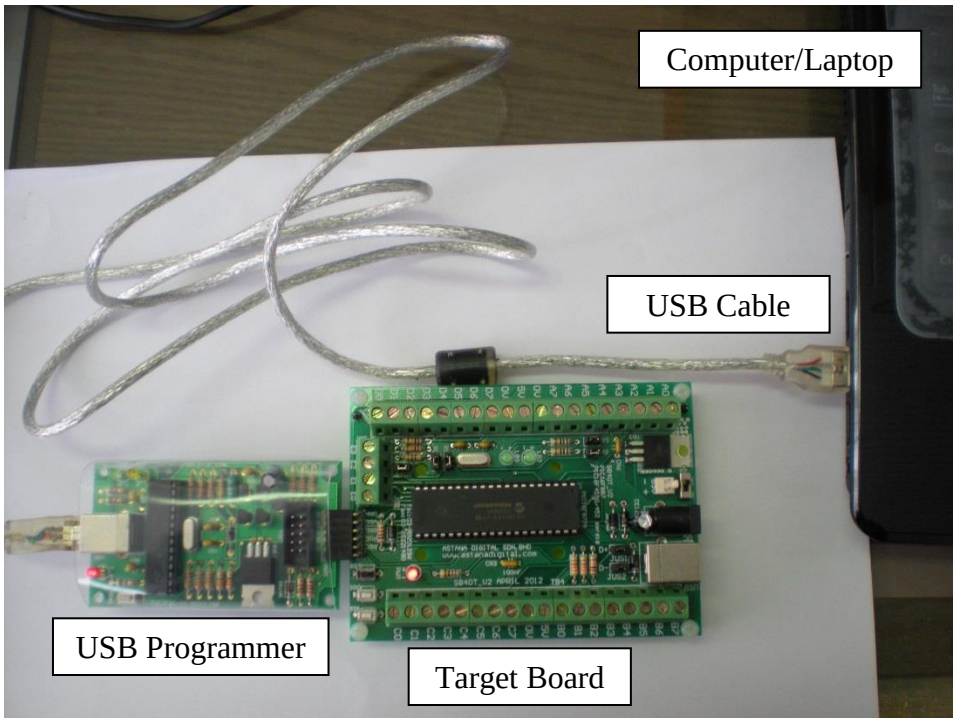


Figure 2: Basic Hardware Connection

MPLAB IDE v8.85

MPLAB Integrated Development Environment (IDE) is a comprehensive editor, project manager and design desktop for application development of embedded designs using MicrochipPIC micro and dsPIC microcontrollers. It is a free product of Microchip Inc. and is an effort to make source code development as smooth and comprehensive as possible. It is called an Integrated Development Environment, or IDE, because it provides a single integrated “environment” to develop code for embedded microcontrollers. MPLAB IDE provides a good platform for other compiler language tools to be integrated. MPLAB C17, MPLAB C18 and MPLAB C30 from microchip provide fully integrated, optimized code. Along with compilers from HI-TECH, IAR, micro Engineering Labs, CSC and Byte Craft, they are invoked by MPLAB IDE project manager to compile code.

The initial use of MPLAB IDE is covered here. This section shows the method to install MPLAB IDE. It is followed by a simple tutorial to create a project. Those who are unfamiliar with MPLAB IDE will get a basic understanding of using the system to develop an application. No previous knowledge is assumed, and comprehensive technical details of MPLAB IDE and its components are omitted in order to present the basic framework for using MPLAB IDE.

Proteus 7 Professional

Proteus is simulator software which is capable to simulate any circuit and scenario. Mainly, it is best for microcontrollers. Proteus professional design combines the ISIS schematic capture and ARES PCB layout programs to provide a powerful, integrated and easy to use tools suite for education and professional PCB Design. The Proteus Professional are software for automated design of electronic circuits. The package is a system of circuit simulation, based on the models of electronic components in SPICE. A distinctive feature of the package Proteus Professional is the possibility of modelling of the programmable devices: microcontrollers, microprocessors, DSP and others.

5.0 Components Specification

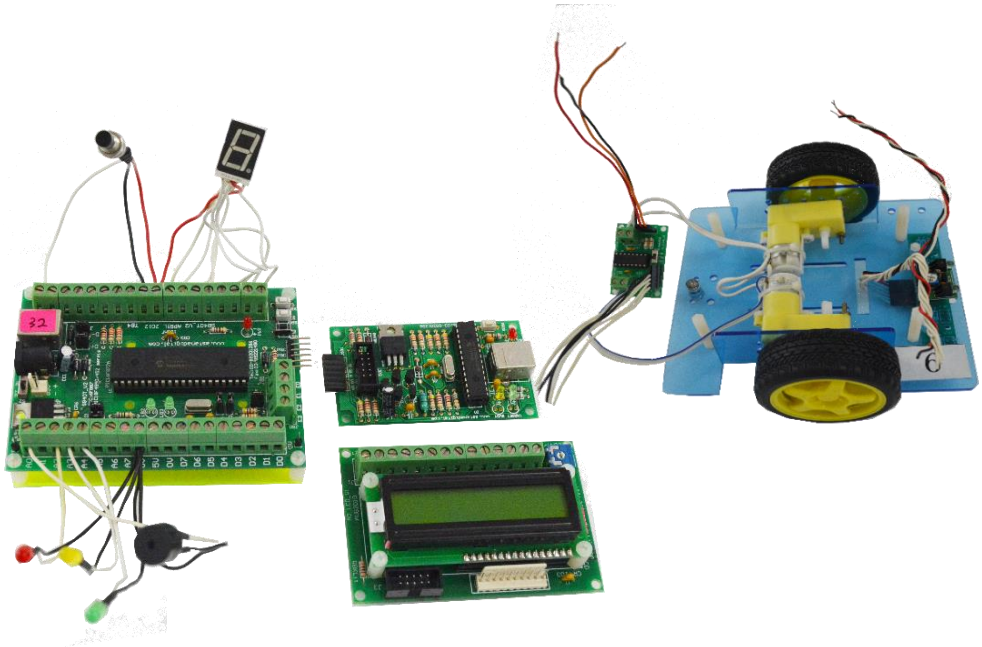
Table 1 shows the components specification for PIC18F458 Teaching Kit.

Table 1: PIC18F458 Teaching Kit Components Specification

No	Components	Model/Specification	Quantity
1	SB40 Target Board PIC18F458	SKU: 10001210 www.astanadigital.com	1
2	USB PIC Programmer	SKU: 1001067 www.astanadigital.com	1
3	USB Cable Type A to B	CSMUAB-1M Transfer rate up to 480Mbps www.I-com.com	1
4	LCD 16x2 Module	SKU: 10001025 www.astanadigital.com	1
5	Mobile Robot Base	SKU: 8369638 www.astanadigital.com	1
6	L293D H-Bridge Module Motor Driver	SKU: 1001027 www.astanadigital.com	1

7	Light Emitting Diode (LED) - Red, Yellow and Green	5mm diffused case www.rapidonline.com	10
8	Buzzer 5V	Apply 3V to 5V, Loud 2KHz www.rapidonline.com	1
9	Push Button	250mA 60V AC www.rapidonline.com	2
10	7-Segment	7-segment LED display component with decimal point	1
11	LDR	SK-LDR-5mm www.shelfkey.com	1
12	Super Bright LED	Blue 10mm, 30 degree viewing angle www.robotshop.com	1
13	Computer with MPLab IDE and Proteus.	-	1

6.0 PIC18F458 Teaching Kit



7.0 Teaching Kit Operational Procedure

i. Prepare all the equipment and material use such as target board, programmer, input-output (I/O) components, LCD module, robot mobile and USB cable.
ii. Plug one end of the USB cable into the USB programmer. Plug the other end into a USB port on your PC.
iii. Connect the USB programmer to the target board. The lights indicator on the target board and USB progammer will turn on. It shows that the equipment is in good condition.
iv. Open MPLAB IDE Workspace window. Select menu Programmer – Select Programmer – 9 PICKit2.
v. Then, the output window will display ‘PICKit2 Ready’. It’s means the hardware equipment is ready to communicate with the software.
vi. Follow the next procedure to complete the practical work.

LABORATORY

[CONTENT OF PRACTICAL WORK]

1. PW1 : INTRODUCTION TO MPLAB IDE v8.85 AND C18 COMPILER
2. PW2 : INPUT AND OUTPUT PROGRAMMING IN C
3. PW3 : CIRCUIT SIMULATION ON PIC
4. PW4 : PIC PROGRAMMING IN C (7-SEGMENT)
5. PW5 : PIC PROGRAMMING IN C (DC MOTOR)
6. PW6 : HARDWARE LCD INTERFACING

PRACTICAL WORK : 1

TITLE : INTRODUCTION TO MPLAB IDE v8.85 AND C18 COMPILER

OBJECTIVES

a) To do software installation of MPLAB IDE v8.85 and Microchip C18 Toolsuite.
b) To write some simple C code that will run on the PIC18.
c) To simulate results of the C program using MPLAB Tools.
d) To observe an output of C program using Program Memory and Disassembly Listing.

EQUIPMENTS

1. Microchip C18 Toolsuite
2. MPLAB IDE v8.85
3. Computer
4. PIC18F458 Training Kit

THEORY

MPLAB IDE

MPLAB IDE (Integrated Development Environment) is a professional software implemented by Microchip, compatible with Windows 7, Vista and XP.

MPLAB IDE is used as a powerful aid to the development of systems based on PIC microcontrollers. Its distribution is free. Download may be made from Microchip site.

Figure 1: MPLAB IDE v8.85 Shortcut

C18 COMPILER

The C18-MPLAB ® IDE, is a C compiler for PIC18 professional MCU, and a powerful tool for ANSI compatible PIC18 family of PICmicro ® 8-bit MCU. The C18 compiler is a 32-bit Windows ® environment fully integrated with Microchip MPLAB IDE and allows developing and debugging programs using their software tools.

MPLAB C18 C Compiler is a cross-compiler that runs on a PC and produces code that can be executed by the Microchip PIC18XXXX family of microcontrollers. Like an assembler, the MPLAB C18 compiler translates human-understandable statements into ones and zeros for the microcontroller to execute. Unlike an assembler, the compiler does not do a one-to-one translation of machine mnemonics into machine code.

MPLAB C18 takes standard C statements, such as “if(x==y)” and “temp=0x27”, and converts them into PIC18XXXX machine code. The compiler incorporates a good deal of intelligence in this process. It can optimize code using routines that were employed on one C function to be used by other C functions. The compiler can rearrange code, eliminate code that will never be executed, share common code fragments among multiple functions, and can identify data and registers that are used inefficiently, optimizing their access.

Code is written using standard ANSI C notation. Source code is compiled into blocks of program code and data which are then “linked” with other blocks of code and data, then placed into the various memory regions of the PIC18XXXX microcontroller. This process is called a “build,” and it is often executed many times in program development as code is written, tested and debugged. This process can be made more intelligent by using a “make” facility, which invokes the compiler only for those C source files in the project that have changed since the last build, resulting in faster project build times.

MPLAB C18 compiler and its associated tools, such as the linker and assembler, can be invoked from the command line to build a *.HEX file that can be programmed into a PIC18XXXX device. MPLAB C18 and its other tools can also be invoked from within the MPLAB IDE. The MPLAB graphical user interface serves as a single environment to write, compile and debug code for embedded applications. The MPLAB dialogs and project manager handle most of the details of the compiler, assembler and linker, allowing the task of writing and debugging the application to remain the main focus.

MPLAB C18 compiler makes development of embedded systems applications easier because it uses the C standard language. There are many books that teach the C language, and some are referenced in the Preface “Recommended Reading”. This guide will assume an understanding of the fundamentals of programming in C. The advantage of the C language is that it is widely used, is portable across different architectures, has many references and textbooks, and is easier to maintain and extend than assembly language. Additionally, MPLAB C18 can compile extremely efficient code for the PIC18XXXX microcontrollers.’

PROCEDURE A : MPLAB IDE v8.85 INSTALLATION

1) Run setup.exe of the MPLAB IDE v8.85 as Figure 2.

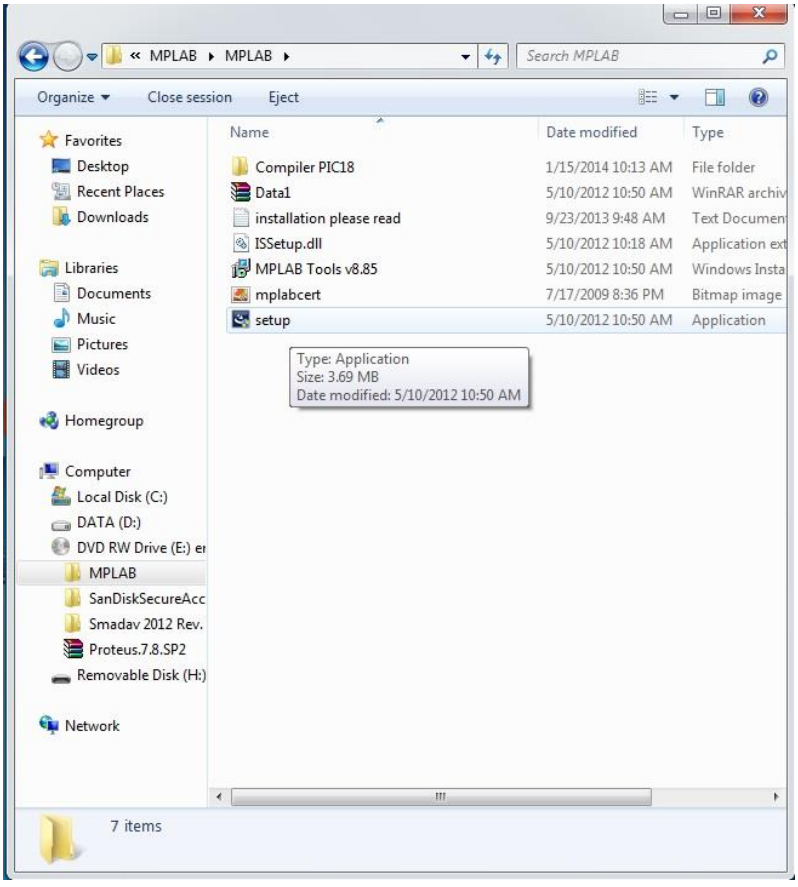


Figure 2: Setup.exe of MPLAB IDE v8.85

2) Complete the installation until Finish.

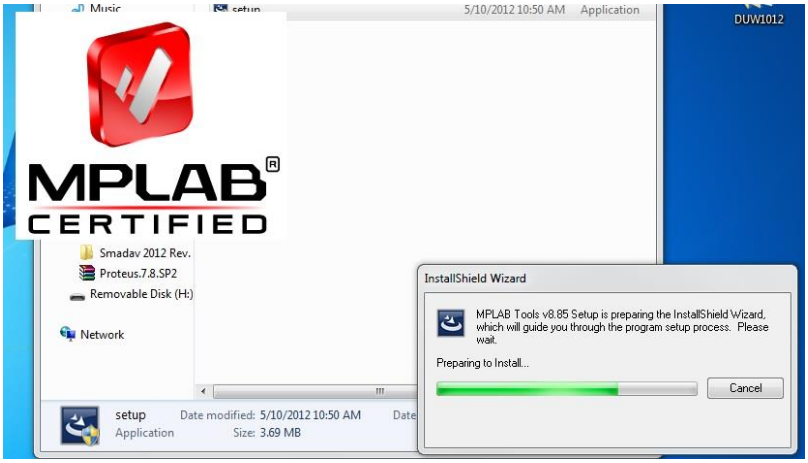


Figure 3: InstallShield Wizard Pop-up

3) When the installation is complete, the MPLAB IDE v8.85 shortcut will appear at your desktop.

PROCEDURE B : MPLABC18 v3.42 LITE INSTALLATION

1) Run MPLAB C18 v3.42 LITE installer.

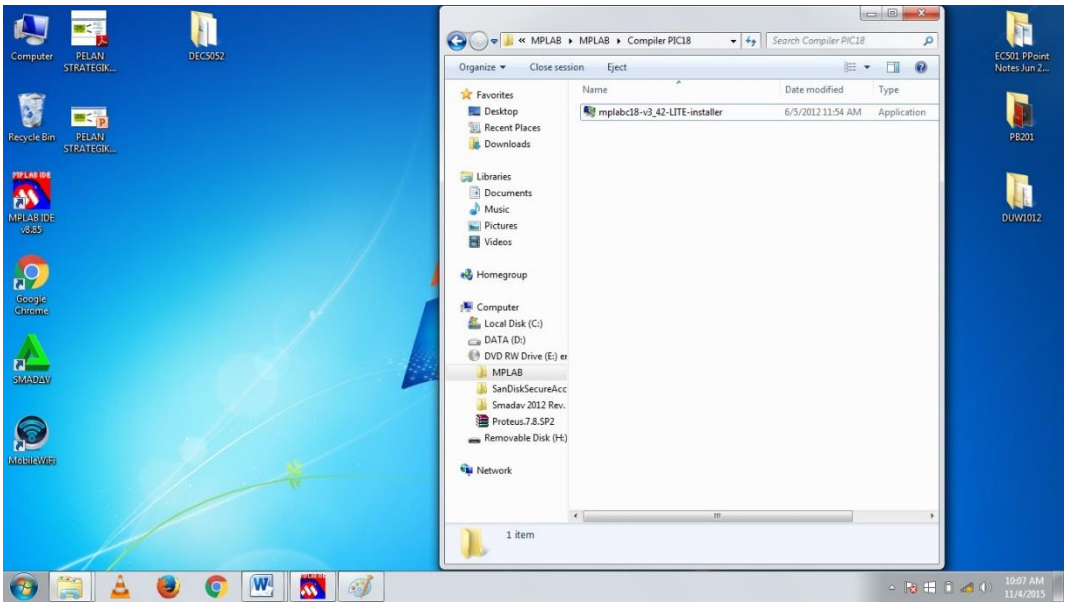


Figure 4: MPLAB C18 v3.42 LITE

2) Complete the installation until Finish.

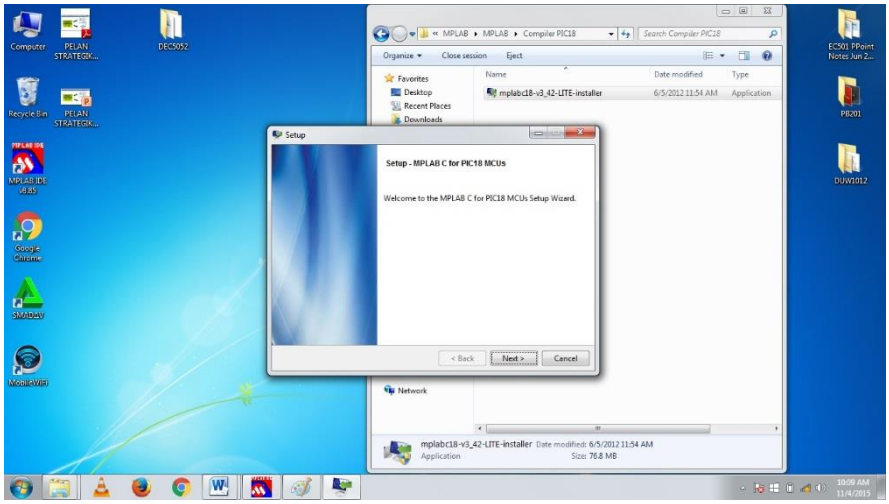


Figure 5 : Setup Window

3) Accept the agreement in License Agreement pop-up window.

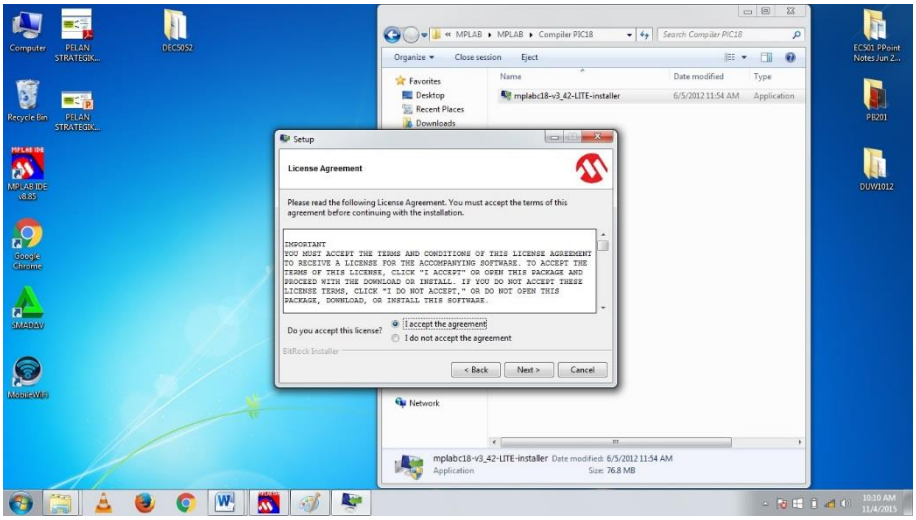


Figure 6 : License Agreement Window

4) Complete the installation until Finish.

PROCEDURE C : CREATE PROJECT WIZARD

1) Create New Folder at your desktop then run the MPLAB IDE v8.85.
2) Select menu Project – Project Wizard – Select Device : **PIC18F458** – Select Active Toolsuite as below then select Next >.

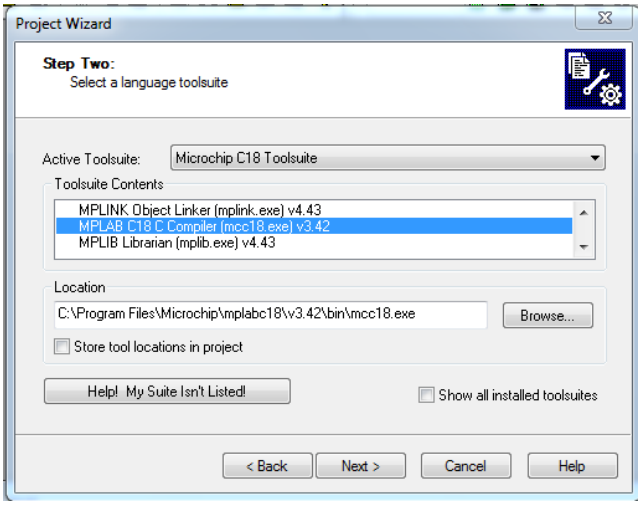


Figure 7 : Project Wizard Pop-up Window

3) Browse the folder that was created at the desktop and give file name as Project.mcp.
4) Click Next > button until Finish.

PROCEDURE D : MPLAB IDE v8.85 EDITOR

1) Double-clicks at MPLAB IDE shortcut and workspace window will display as below and then create a new file editor.

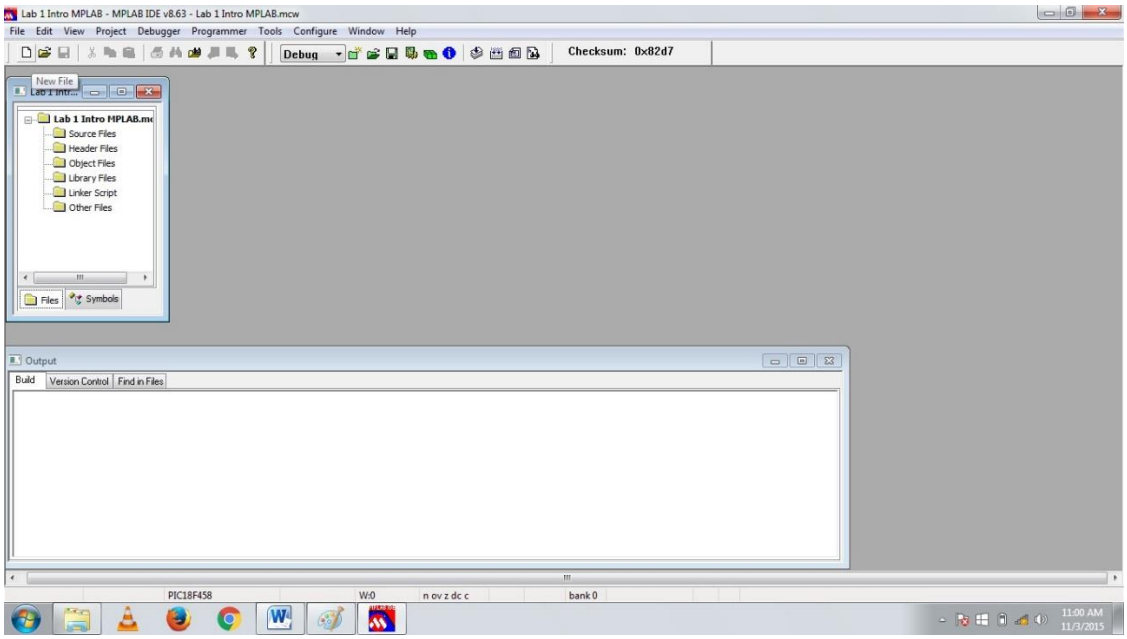


Figure 8 : MPLAB Workspace

2) Write a source code as below in the editor windows and save file as C file (*.c) in your folder at the desktop.

```
#include<p18f458.h>

void main (void)

{
                TRISB=1;                    //PORT B as Input
                TRISCbits.TRISC1=0;         //Pin C1 as Output
                PORTB=0;                    //Clear all Port B
                PORTC=0;                    //Clear all Port C

                PORTCbits.RC1=1;            //Pin C1 as ON

}
```

3) Right click at Source Files folder in Project.mcp and add your C file.

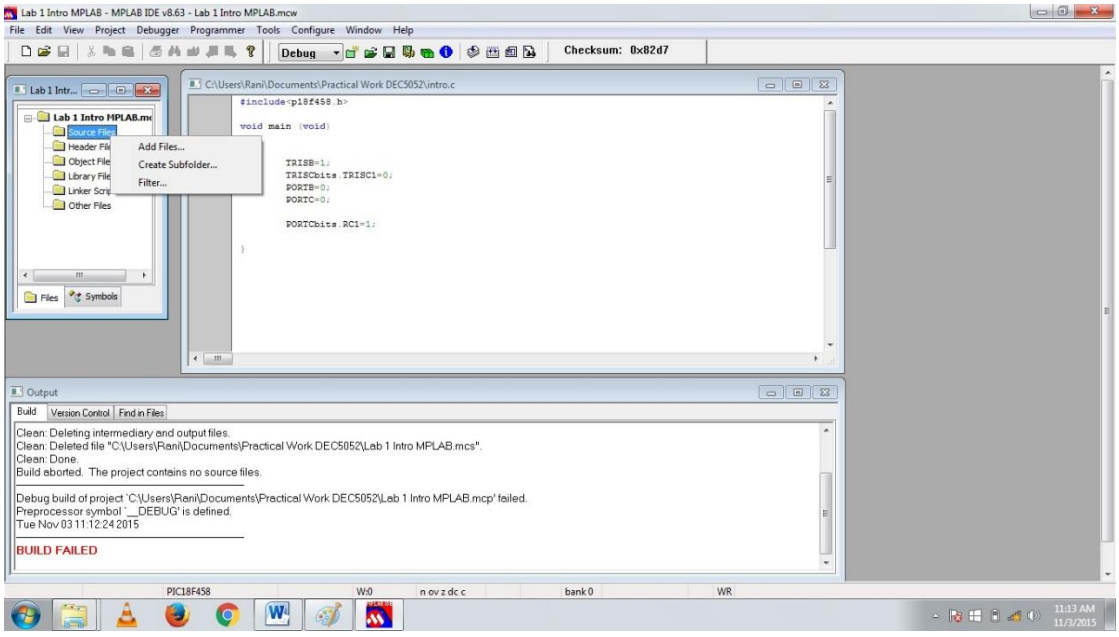


Figure 9 : Add Files in Source File

4) Compile (Build All) the source code until “Build Succeeded”.

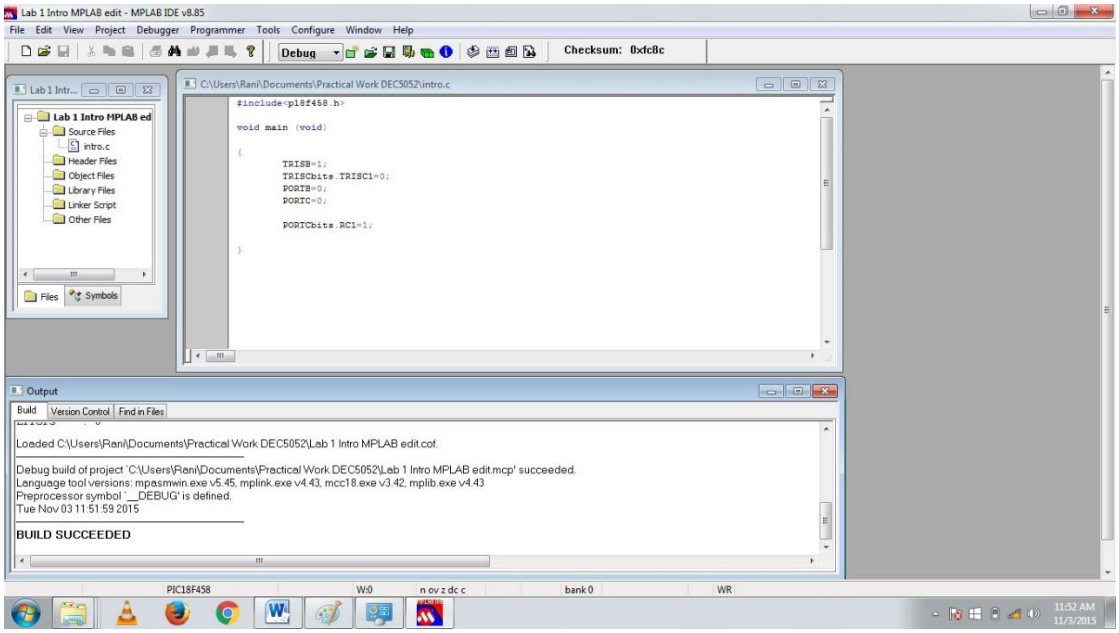


Figure 10 : Build Succeeded at Output Window

PROCEDURE E : PROGRAM MEMORY AND DISASSEMBLY LISTING

1) Select menu View – Program Memory and Disassembly Listing.
2) Program Memory window displays the address in hexadecimal and assembler source code.
3) Disassembly Listing window displays the program code in C program and assembler program. It also displays the address of the program code.

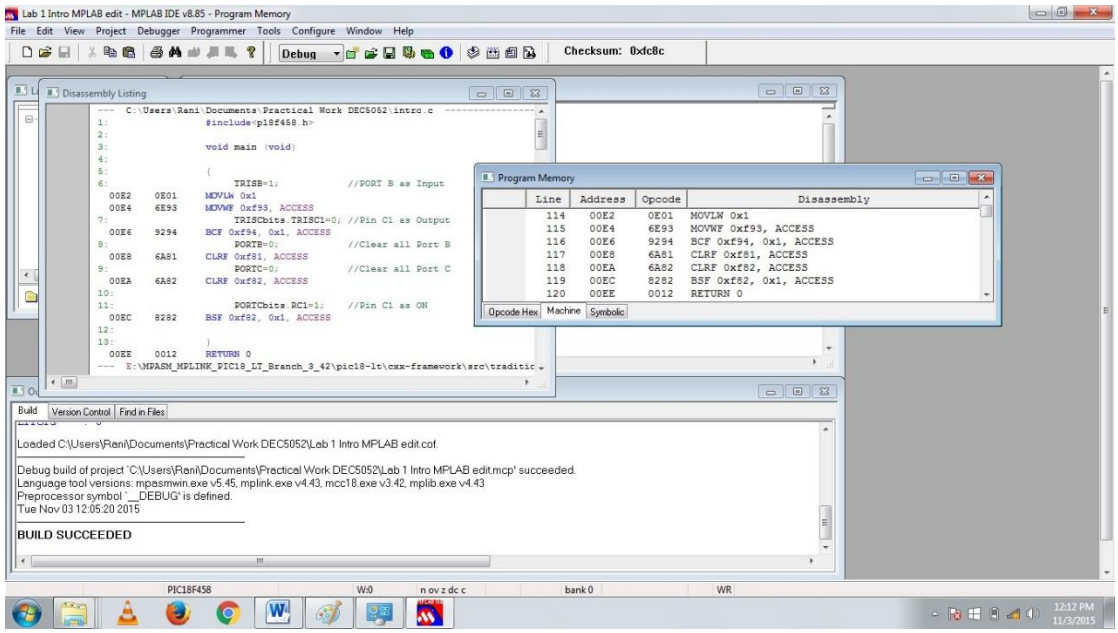


Table 1 : Program Memory and Disassembly Listing Popup Windows

PROCEDURE F : MPLAB IDE v8.85 WORKSPACE

1) Minimize the MPLAB IDE v8.85 Workspace and open your folder at the desktop.
2) The folder should contain C file, object file, HEX file, COF file, MCS file, MCP file and MCW file.

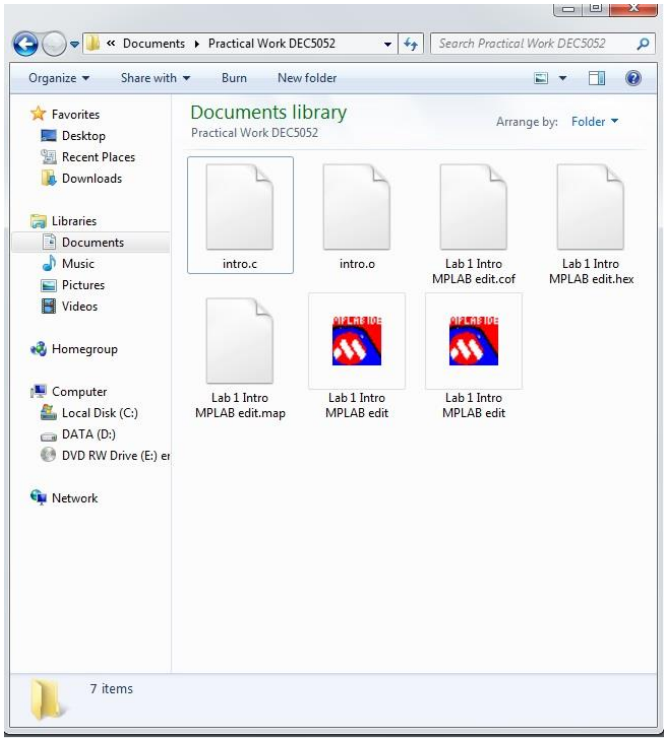


Table 2 : Project Workspace Contains

EXERCISES

1) Use the source code of the Procedure D to observe the contains of the Program Memory and fill the table below.

Line	Address	Opcode	Disassembly

Table 1 : Program Memory

2) List the contains of your folder at the desktop.

No.	File Name

Table 2 : Project Workspace Contains

PRACTICAL WORK : 2

TITLE : INPUT AND OUTPUT PROGRAMMING IN C

OBJECTIVES

a) To connect interfacing for LEDs, buzzer and push button.
b) To modify a C language program and observe results on target board.
c) To use the MPLAB and C18 Compiler to program PIC18 with machine code defined in a hex file.
d) To write C program to define TRIS and PORT Registers.

EQUIPMENTS

1. Microchip C18 Toolsuite
2. MPLAB IDE v8.85
3. Computer
4. PIC18F458 Training Kit

THEORY

PIC18

For PIC18F4550 chip set, there are up to five ports available. Based on Figure 1, there are PORTA, PORTB, PORTC, PORTD and PORTE. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Each port has three registers for its operation. These registers are:
i. TRIS register = data direction register.
ii. PORT register = reads the levels on the pins of the device.
iii. LAT register = output latch (The Data Latch register (LAT A) is useful for read modify-write operations on the value driven by the I/O pins.)

40-Pin PDIP

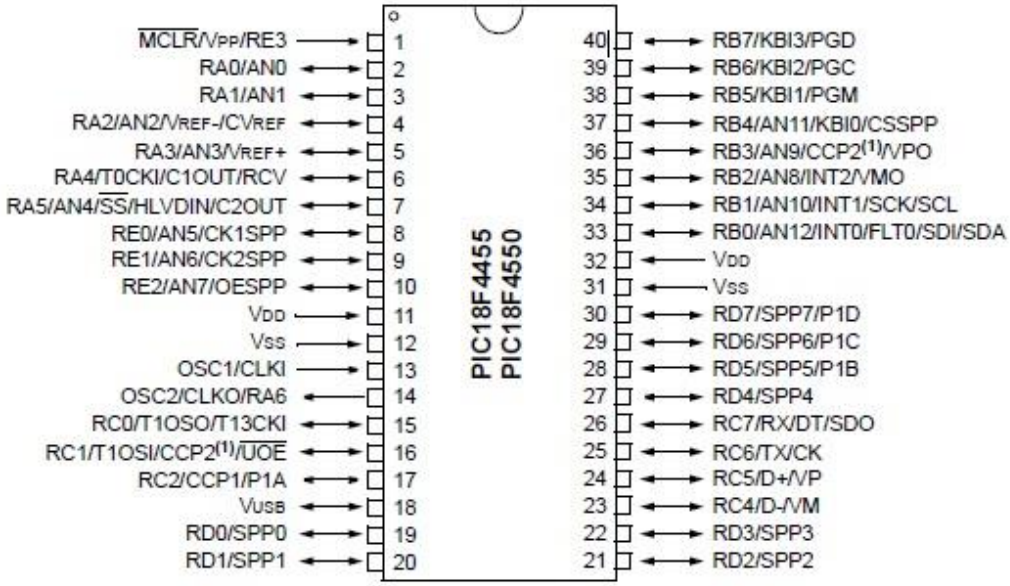


Figure 1 : PIC18 I/O Pin

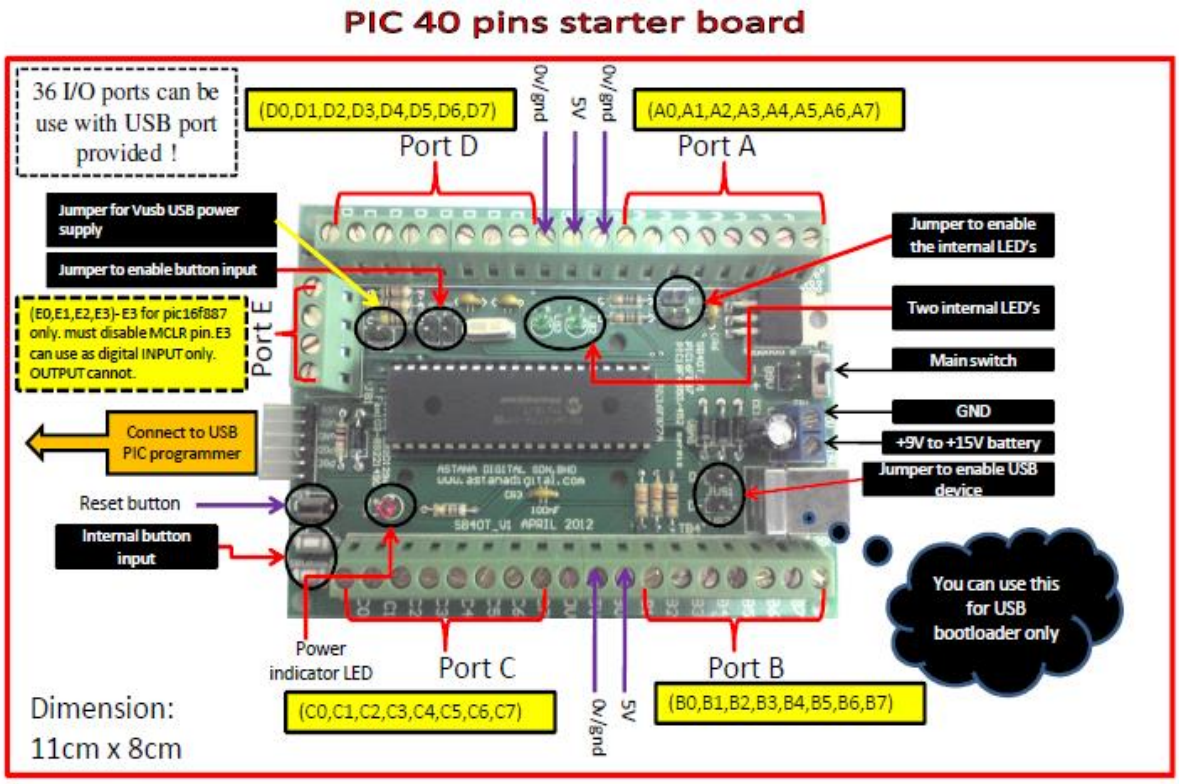


Figure 2 : Junior Pack Starter Board Layout

PROCEDURE A : SOFTWARE & HARDWARE INSTALLATION

1) Create New Folder at your desktop then run the MPLAB IDE v8.85.
2) Select menu Project – Project Wizard – Select Device : **PIC18F458** – Select Active Toolsuite as below then Next :

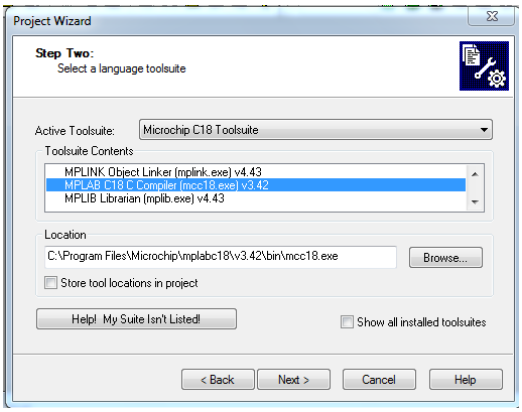


Figure 3 : Project Wizard Pop-up Window

3) Browse the folder that was created at the desktop and name the file as Project.mcp.
4) Click Next button until Finish.
5) Complete an installation of Target Board SB40T, PICKit 2 Programmer, USB Connector and PC/Laptop.

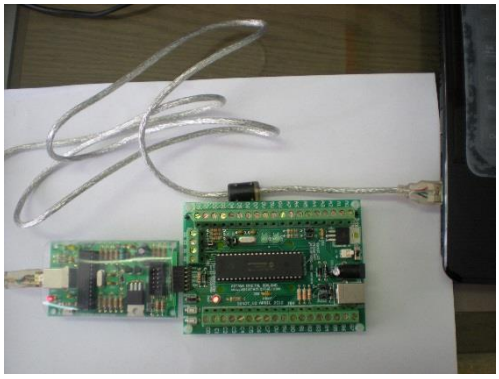


Figure 4 : PIC18 Junior Pack Hardware Connection

6) I/O connection :
    a. Red LED – B4 White and 0V Black
    b. Push Button – B0 White, 5V Black and 0V Red

7) Open MPLAB IDE v8.85. Select menu Programmer – Select Programmer – PICKit 2.
8) Re-establish PICKit 2 connection until PICKit 2 toolbar ready.

Figure 5 : PICKit2 Toolbar Ready

PROCEDURE B : LED & PUSH BUTTON

1) Write a source code as below in the editor windows and save file as Lab2A.c in your folder at the desktop.
2) Right click at Source Files folder in Project.mcp and add Lab2A.c.
3) Select Configure – Configure Bits – Untick Configuration Bits set in code – Change OSC (HS Oscillator), Disabled BOR (Brown-out Reset) and Disabled WDT (Watchdog Timer).
4) Select Project – Build All and if there are errors, reedit the source code Lab2A.c.
5) Compile the source code until “Build Succeeded”.
6) Then, select (Program target device) and (Bring target.....).
7) Observe the output at the target board.

```
//Lab2A.c

#include <p18f458.h>
void main(void)
{
TRISBbits.TRISB0=1;          // declare Push Button  as input
TRISBbits.TRISB4=0;          // declare  LED as output
PORTB=0;                     // clear content of PORTB

here:                        // position to jump
if(PORTBbits.RB0==0)
{
PORTBbits.RB4=1;
}
PORTBbits.RB4=0;
goto here;                   // jump to here

}
```

PROCEDURE C : LED, PUSH BUTTON AND BUZZER

1) Write the source code as below in the editor windows.
2) Save file as Lab2C.c in your folder at the desktop.
3) Right click at Source Files folder in Project.mcp and add Lab2C.c.
4) Compile the source code until “Build Succeeded”.
5) Then, select (Program target device) and (Bring target.....).
6) Observe the output at the target board.

```
//Lab2C.c

#include <p18f458.h>
#include <delays.h>
#define XTAL_FREQ 20Mhz

void LED_On(void);
void Buzzer_On (void);

void main(void)
{
        TRISBbits.TRISB0=1;   // push button as input
        TRISBbits.TRISB2=0;   // buzzer as output
        TRISBbits.TRISB4=0;    //led as output

        PORTB=0;               //clear content of PORTB

here:                          // position to jump

        if (PORTBbits.RB0==0)LED_On();   //function call
        if (PORTBbits.RB0==1)Buzzer_On();
        goto here;
}

void LED_On(void)
{
        PORTBbits.RB4=1;

}

void Buzzer_On(void)
{
        PORTBbits.RB2=1;
}
```

EXERCISES

1) Modify the Lab2A source code to make the push button;
    a. Press – ON
    b. Release – OFF

2) Modify the connection and Lab2C source code to make the push button;
    a. Press – LED and Buzzer ON

3) Modify Lab2C.c to create source code to make the toggle output for 8 LEDs in Port B as below :
    a. Press Button : LED , RB7, RB5, RB3, and RB1 are ON.
    b. Release Button : LED RB6, RB4, RB2, and RB0 are ON
    c. Write down and state the result for a new source code in answering form.

PRACTICAL WORK : 3

TITLE : CIRCUIT SIMULATION ON PIC

OBJECTIVES

a) To use a test program to verify operation of LCD.
b) To draw a schematic PIC18 diagram using Proteus 7.
c) To use hex file to simulate the PIC18 diagram in Proteus 7.

EQUIPMENTS

1. C18 Compiler
2. Proteus 7
3. Computer

THEORY

MPLAB

MPLAB Integrated Development Environment (IDE) is a comprehensive editor, project manager and design desktop for application development of embedded designs using MicrochipPIC micro and dsPIC microcontrollers. It is a free product of Microchip Inc. and is an effort to make source code development as smooth and comprehensive as possible. It is called an Integrated Development Environment, or IDE, because it provides a single integrated “environment” to develop code for embedded microcontrollers. MPLAB IDE provides a good platform for other compiler language tools to be integrated. MPLAB C17, MPLAB C18 and MPLAB C30 from microchip provide fully integrated, optimized code. Along with compilers from HI-TECH, IAR, micro Engineering Labs, CSC and Byte Craft, they are invoked by MPLAB IDE project manager to compile code. The initial use of MPLAB IDE is covered here. This section shows the method to install MPLAB IDE. It is followed by a simple tutorial to create project. Those who are unfamiliar with MPLAB IDE will get a basic understanding of using the system to develop an application. No previous knowledge is assumed, and comprehensive technical details of MPLAB IDE and its components are omitted in order to present the basic framework for using MPLAB IDE.

Programming and Compiler

C18 is a very widely used compiler for PIC18. It offers programming of almost all the PIC Microcontrollers, along with its appealing and friendly interface

Setting input and output port

TRIS register is vital part in order to control the PORT in PIC18. The value in register will determine the port become as input or output port. For example value TRISB = FFH or TRISB = B '11111111' will set PORTB as input port and TRISB = 00H or TRISB = B '00000000' will turn PORTB as output port. Value 1 or 0 will determine the state of port. Sometimes, PORT can also be as input and output port. For example TRISB = F0H or TRISB = B '11110000' will make the pins RB0 – RB3 as output port and the rest pins RB4 – RB7 as input port. This technique applicable for all PORTA, PORTC, PORTD and PORTE.

Circuit Simulation

A common tool for the electrical/electronic designer is the circuit simulator. Although it is usually called a *simulator*, it is a software application that typically boasts many functions beyond electrical signal simulation, including schematic capture, printed circuit board layout, and bill of materials generation.

These tools are available in both the public domain and commercially. The simulator used in our electrical labs is called Proteus as shown in Figure 1.

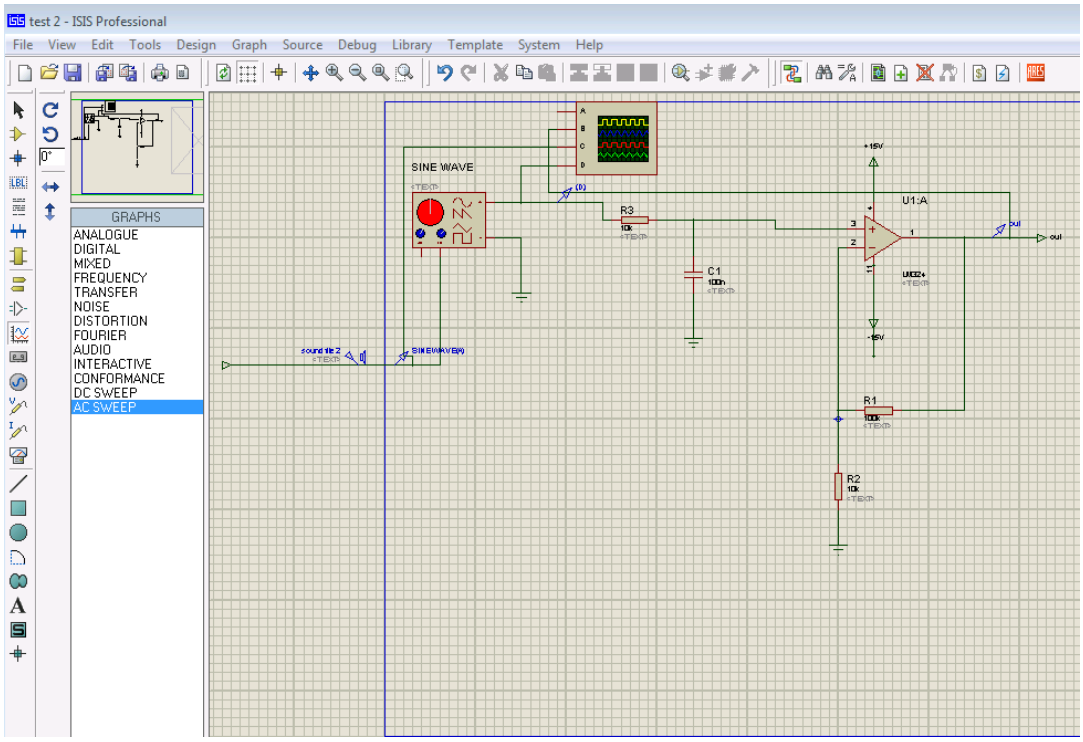
PROTEUS Program

Proteus contains everything you need to develop, test and virtually prototype your embedded system designs based around the Microchip Technologies™ PIC18 series of microcontrollers. The unique nature of schematic based microcontroller simulation with Proteus facilitates rapid, flexible and parallel development of both the system hardware and the system firmware. This design synergy allows engineers to evolve their projects more quickly, empowering them with the flexibility to make hardware or firmware changes at will and reducing the time to market. Proteus VSM models will fundamentally work with the exact same HEX file as you would program the physical device with, binary files (i.e. Intel or Motorola Hex files) produced binary assembler or compiler.

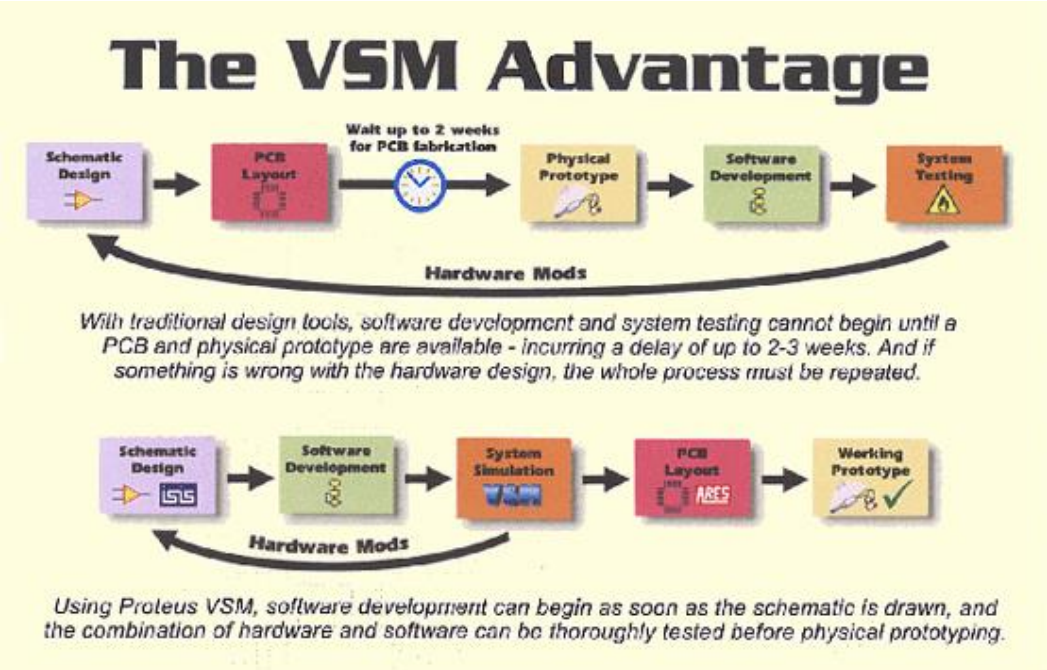
The Proteus Design Suite is wholly unique in offering the ability to co-simulate both high and low-level micro-controller code in the context of a mixed-mode SPICE circuit simulation. With this Virtual System Modelling facility, you can transform your product design cycle, reaping huge rewards in terms of reduced time to market and lower costs of development.

If one person designs both the hardware and the software then that person benefits as the hardware design may be changed just as easily as the software design. In larger organisations where the two roles are separated, the software designers can begin work as soon as the schematic is completed; there is no need for them to wait until a physical prototype exists.

In short, Proteus VSM improves efficiency, quality and flexibility throughout the design process as shown in Figure 2.



Figures 1 : Proteus ISIS Workspace



Figures 2 : VSM Advantage

LCD Display

The 2x16 character LCD offers character display for embedded system. It can be used to display numerical information, text message and also special symbol. We can control a LCD using either 8 pins (8-bit interface) or 4 pins (4-bit interface), depending on the I/O pins that we have.

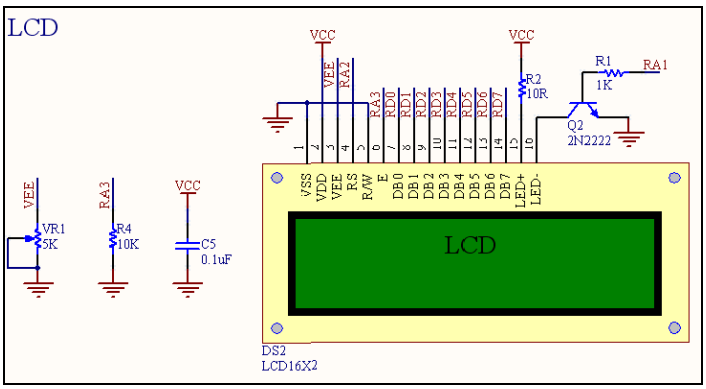


Figure 3 : Connection of a 2x16 character LCD

PROCEDURE A

1) Write the source code as below (**save as LCD.c)**;

```
#pragma config OSC = HS, OSCS = OFF
#pragma config PWRT= OFF , BOR=ON, BORV = 45
#pragma config WDT = OFF
#pragma config DEBUG = OFF ,LVP =OFF , STVR =OFF
#include <p18f458.h>
#include "lcd16.h"

void main()
{
        TRISD=0;
        TRISC=0;
        EN=0;
        lcdinit();
        gotoXy(0,0); // Lcd line
        prints("Hello Word!!");

while(1)
{
}
}
```

2) By using MPLAB compile the source code with external header as Figure 4 to get the hex file.

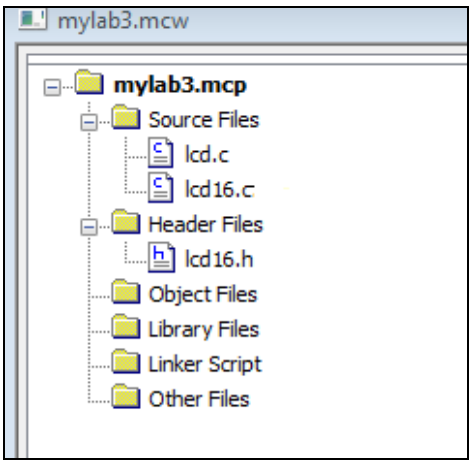


Figure 4 : Header File

3) Finally the hex file was created as

PROCEDURE B

1) Design circuit below by using Proteus.

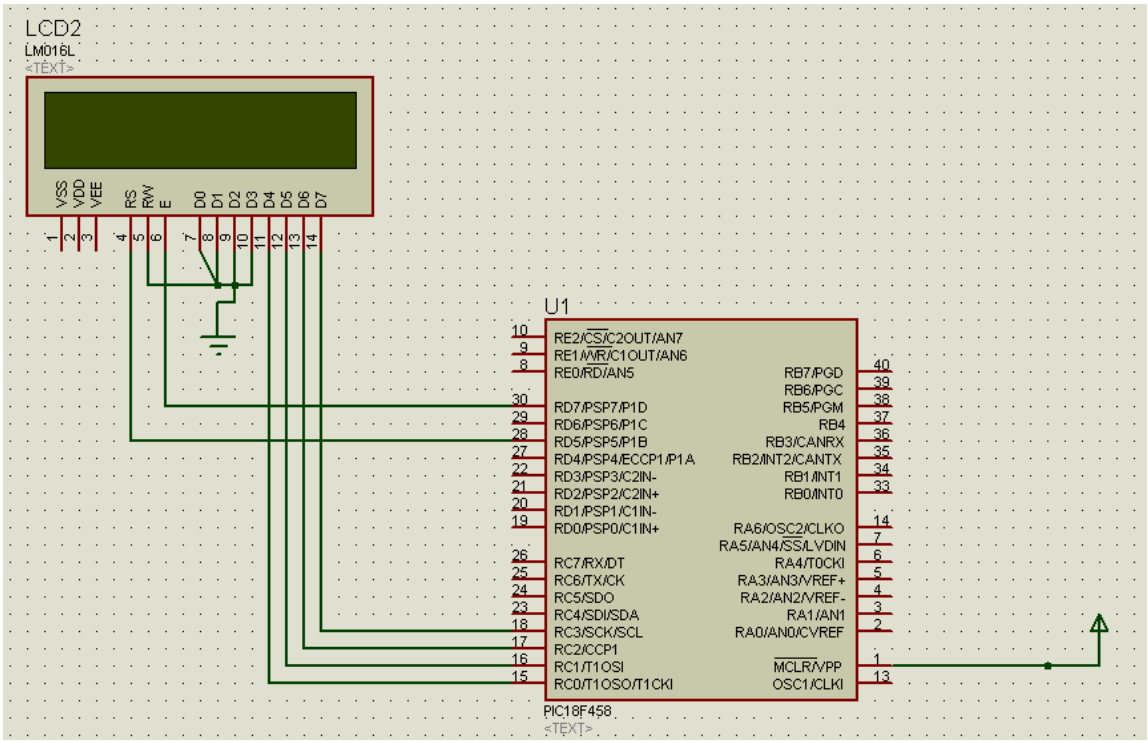


Figure 5 : PIC18F458 LCD Interfacing

2) Run Proteus and get familiarized with the tools around.
3) Press ‘P’ and it show a window to select device use. First you need to add controller i.e PIC18F458. Type ‘**PIC18F458’** in keywords text box. It will automatically show the controller results. Select it and press OK. Repeat the same procedure to add another component.
4) As in actual controller you need to download the HEX file generated in **Procedure A**. Same way will give that HEX file to Proteus to simulate the program. Double-click controller and click browse icon in front of ‘Program File’ text box.
   - Select the compiler generated HEX file and press OK.
   - Now everything is set to go. Press ‘F12’ to start the simulation.
   - or Insert the **HEX file** (**mylab3.hex**) into the PIC18F458 LCD interfacing circuit.

   Run the circuit and if have error need to correct it.
5) Observe the result that you get on LCD display.

PROCEDURE C

1) Write the source code as below (**save as LED.c)**;

```
#include<p18f458.h>
#define XTAL_FREQ 20MHz
#define BUTTON1 PORTAbits.RA0
#define BUTTON2 PORTAbits.RA1
#define BUTTON3 PORTAbits.RA2
#define BUTTON4 PORTAbits.RA3

void pattern1(void)                    //function for pattern 1
{
PORTB = 0b11111111;
}
void pattern2(void)                    //function for pattern 2
{
PORTB = 0b10101010;
}
void pattern3(void)                    //function for pattern 3
{
PORTB = 0b11110000;
}
void pattern4(void)                    //function for pattern 4
{
PORTB = 0b00001111;
}

void main(void)
{
ADCON1 = 0b00000110;                   //Set ADCON1 for Port A as a digital input
TRISA = 0b11111111;                    //Port A as input port
TRISB = 0b00000000;                    //Port B as output port
PORTB = 0b00000000;                    //Clear PORTB at start up

while(1)                               //endless loop
{
if (BUTTON1 == 1)                      //Test button at PORTA bit 0 if pressed.
pattern1();                            //call function pattern1
else if (BUTTON2 == 1)
pattern2();                            //call function pattern2
else if (BUTTON3 == 1)
pattern3();                            //call function pattern3
else if (BUTTON4 == 1)
pattern4();                            //call function pattern4
else


PORTB = 0b00000000;                    //All LED off if no button pressed

}                                      //end of while (endless loop)
}                                      //end of main function
```

2) By using MPLAB compile the source code to get the hex file.

PROCEDURE D

1) Design circuit below by using Proteus.

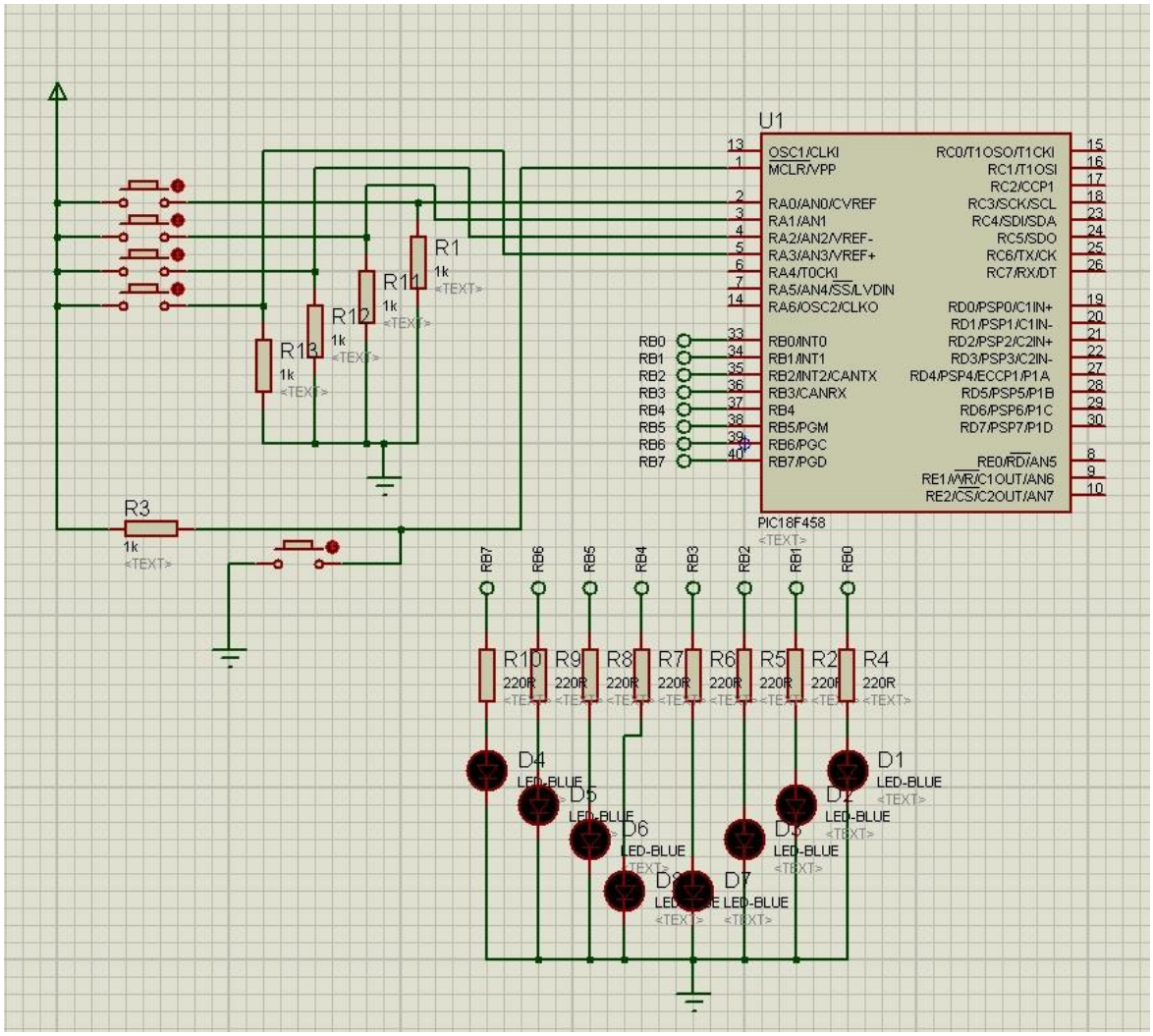


Figure 6 : PIC18F458 LED Interfacing

2) Load HEX file to Proteus to simulate the program.
3) Observe the result that you get on LED display.

EXERCISE

1) Write the source code to display ouput as below ;

*Embedded System ****DEC5052*****

2) Write a source code to display output as blinking sentences**;**

3) Write a source code to display output as running sentences.

PRACTICAL WORK : 4

TITLE : PIC PROGRAMMING IN C (7-SEGMENT)

OBJECTIVES

a) To use a PIC18 to display a number on 7-segment display.
b) To write and program using C language subroutine using Delay.
c) To connect interfacing for 7-segments.
d) To observe an output of C program on the 7-segment.

EQUIPMENTS

1. Microchip C18 Toolsuite
2. MPLAB IDE v8.85
3. Computer
4. PIC18F458 Training Kit

THEORY

SEVEN-SEGMENT DISPLAY

A seven-segment display (SSD), or seven-segment indicator, is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot-matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, and other electronic devices for displaying numerical information.

The seven elements of the display can be lit in different combinations to represent the Aarabic numerals. Often the seven segments are arranged in an*oblique* (slanted) arrangement, which aids readability. In most applications, the seven segments are of nearly uniform shape and size (usually elongated hexagons, though trapezoids and rectangles can also be used), though in the case of adding machines, the vertical segments are longer and more oddly shaped at the ends in an effort to further enhance readability. The numerals 6, 7 and 9 may be represented by two or more different glyphs on seven-segment displays. The seven segments are arranged as a rectangle of two vertical segments on each side with one horizontal segment on the top, middle, and bottom. Additionally, the seventh segment bisects the rectangle horizontally.

There are also fourteen-segment displays and sixteen-segment displays (for full alphanumerics); however, these have mostly been replaced by dot-matrix displays. The segments of a 7-segment display are referred to by the letters A to G, where the optional DP decimal point (an "eighth segment") is used for the display of non-integer numbers.

In a simple LED package, typically all of the cathodes (negative terminals) or all of the anodes (positive terminals) of the segment LEDs are connected and brought out to a common pin; this is referred to as a "*common cathode*" or "*common anode*" device. Hence a 7 segment plus decimal point package will only require nine pins (though commercial products typically contain more pins, and/or spaces where pins would go, in order to match standard IC sockets. For example, all the anodes of the A segments of each digit position would be connected together and to a driver pin, while the cathodes of all segments for each digit would be connected. To operate any particular segment of any digit, the controlling integrated circuit would turn on the cathode driver for the selected digit, and the anode drivers for the desired segments; then after a short blanking interval the next digit would be selected and new segments lit, in a sequential fashion. In this manner an eight digit display with seven segments and a decimal point would require only 8 cathode drivers and 8 anode drivers, instead of sixty-four drivers and IC pins. A single byte can encode the full state of a 7-segment-display. The most popular bit encodings are *gfedcba* and *abcdefg*, where each letter represents a particular segment in the display. In the *gfedcba* representation, a byte value of 0x06 would (in a common-anode circuit) turn on segments 'c' and 'b', which would display a '1'.

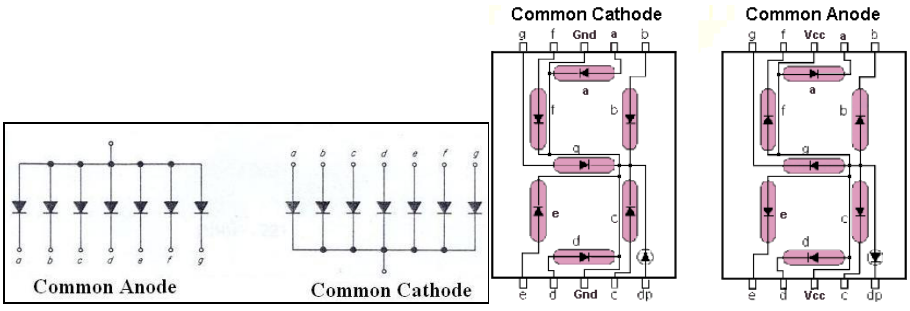
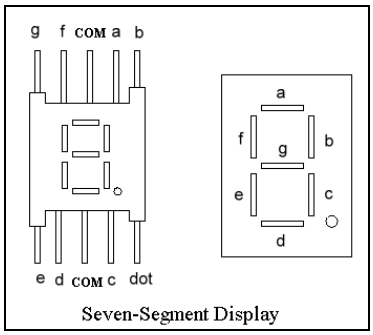


Figure 1 : Interfacing LED's to Microcontroller

Basically there are two types of 7-Segment displays:

1. Common Cathode where all the segments share the same Cathode.
2. Common Anode where all the segments share the same Anode.

Common Anode is order to turn ON a segment the corresponding pin must be set to 0. And to turn it OFF if set to 1. Whereas Common Cathode is order to turn ON a segment the corresponding pin must be set to 1. And to turn it OFF if set to 0. Figure 2 and 3 shown truth table for Seven Segment decoder outputs.



Hex No.	Seven Segment Conversion								7segment equivalent
	dot	g	f	e	d	c	b	a	
0	1	1	0	0	0	0	0	0	CO
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	0	0	0	0	0	B0
4	1	0	0	0	1	0	0	1	99
5	1	0	0	0	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	1	0	0	0	98

Figure 2: Common Anode (active Low) decoder outputs

Hex No.	Seven Segment Conversion								7segment equivalent
	dot	g	f	e	d	c	b	a	
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	1	1	1	1	6F

Figure 3: Common cathode (active High) decoder outputs

PROCEDURE A : SOFTWARE & HARDWARE INSTALLATION

1) Create New Folder or open an existing folder at your desktop then run the MPLAB IDE v8.85.
2) Select menu Project– Open – Browse your folder – Open an existing *.mcp file.
3) Complete an installation of Target Board SB40T, PICKit 2 Programmer, USB Connector and PC/Laptop.
4) I/O connection :
    a. Red LED – B4 White and 0V Black
    b. Buzzer – B2 White and 0V Black
    c. 7-segmen – C0 to C7 White and 0V Red

C0	a
C1	b
C2	c
C3	d
C4	e
C5	f
C6	g
C7	dp

5) Open MPLAB IDE v8.85. Select menu Programmer – Select Programmer – PICKit 2.
6) Re-establish PICKit 2 connection until PICKit 2 toolbar ready.
7) Set up the Configuration Bits in code for OSC, BOR and WDT.

PROCEDURE B : 7-SEGMENT DISPLAY

1) Write a source code as below in the editor windows and save file as Lab4A.c in your folder at the desktop.
2) Right click at Source Files folder in Project.mcp and add Lab4A.c.
3) Select Project – Build All.
4) If there are errors, reedit the source code Lab4A.c.
5) Compile and run the source code.
6) Observe the output at the target board.

```
//Lab4A.c

#include <p18f458.h>
#define XTAL_FREQ 20Mhz

void main(void)
{
unsigned char display[10]={0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
0x6F};                          //array 0-9

TRISBbits.TRISB2=0;             //buzzer
TRISBbits.TRISB4=0;             //led
TRISC=0b00000000;               //7segment

PORTB=0;                        //clear content of PORTB
PORTC=0;                        //clear content of PORTC


while(1)                        // while it's true
{                               //start of statement on while loop
PORTBbits.RB2=1;
PORTBbits.RB4=1;
PORTC=display[7];
}                               // end while loop
}
```

PROCEDURE C : 7-SEGMENT AND COUNTER

1) Write a source code as below in the editor windows.
2) Save file as Lab4B.c in your folder at the desktop.
3) Run and observe the output at the target board.

```
//Lab4B.c

#include <p18f458.h>
#include <delays.h>
#define XTAL_FREQ 20Mhz

void LEDBlink(void);
void BuzzerBeep (void);

void main(void)
{
unsigned char counter;                          //create a variable name counter
unsigned char display[10]={0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
0x6F};                                          //array 0-9

TRISBbits.TRISB2=0;                             //buzzer
TRISBbits.TRISB4=0;                             //led
TRISC=0b00000000;                               //seven segment
PORTB=0;                                        //clear content of PORTB
PORTC=0;
counter=0;                                      //give initial value to counter

while(1)
{                                               //start of while loop true statement
for(counter=0;counter<10;counter++)
{                                               //start for loop
PORTC=display[counter];
Delay10KTCYx(100);
LEDBlink(); }                                   // end for loop
BuzzerBeep();
}                                               //end while loop
}

void LEDBlink(void)
{
        PORTBbits.RB4=1;
        Delay10KTCYx(100);
        PORTBbits.RB4=0;
        Delay10KTCYx(100);
}

void BuzzerBeep(void)
{
        PORTBbits.RB2=1;
        Delay10KTCYx(100);
        PORTBbits.RB2=0;
        Delay10KTCYx(100);
}
```

EXERCISES

1) Modify the Lab4A source code to;
    a. use if statement control
    b. use do while statement control

2) Draw **THREE** flow chart for Lab4A.c for ;
    a. While loop
    b. If
    c. Do while loop

3) Modify Lab4B source code to counter down (from 9 to 0).

4) Modify Lab4B using **Do While** and **For** loop.

5) Draw THREE flowchart for ;
    a. Lab 4B
    b. Lab4B (counter down)
    c. Lab 4B using **Do While** and **For** loop

PRACTICAL WORK : 5

TITLE : PIC PROGRAMMING IN C (DC MOTOR)

OBJECTIVES

a) To use a C18 Compiler to produce hex code for mobile robot application.
b) To control the mobile robot using C program.
c) To connect interfacing for DC motor, LDR and super bright LED.
d) To write and program using C language subroutine.

EQUIPMENTS

1. Microchip C18 Toolsuite
2. MPLAB IDE v8.85
3. Computer
4. PIC18F458 Training Kit
5. Robot Base c/w Motor Driver

THEORY

DC MOTOR

A *DC motor* is a mechanically commutated electric motor powered from direct current (DC). The stator is stationary in space by definition and therefore so is its current. The current in the rotor is switched by the commutator to also be stationary in space. This is how the relative angle between the stator and rotor magnetic flux is maintained near 90 degrees, which generates the maximum torque.

DC motors have a rotating armature winding (winding in which a voltage is induced) but non-rotating armature magnetic field and a static field winding (winding that produces the main magnetic flux) or permanent magnet. Different connections of the field and armature winding provide different inherent speed/torque regulation characteristics. The speed of a DC motor can be controlled by changing the voltage applied to the armature or by changing the field current. The introduction of variable resistance in the armature circuit or field circuit allowed speed control. Modern DC motors are often controlled by power electronics systems called DC drives.

The introduction of DC motors to run machinery eliminated the need for local steam or internal combustion engines, and line shaft drive systems. DC motors can operate directly from rechargeable batteries, providing the motive power for the first electric vehicles. Today DC motors are still found in applications as small as toys and disk drives, or in large sizes to operate steel rolling mills and paper machines.

PHOTORESISTOR

A **photoresistor** or **light dependent resistor (LDR)** is a resistor whose resistance decreases within creasing incident light intensity; in other words, it exhibits *photoconductivity*. A photoresistor is made of a high resistance semiconductor. If light falling on the device is of high enough frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electron (and its hole partner) conduct electricity, thereby lowering resistance.

A photoelectric device can be either intrinsic or extrinsic. An intrinsic semiconductor has its own charge carriers and is not an efficient semiconductor, for example, silicon. In intrinsic devices the only available electrons are in the valence band, and hence the photon must have enough energy to excite the electron across the entire bandgap. Extrinsic devices have impurities, also called dopants, added whose ground state energy is closer to the conduction band; since the electrons do not have as far to jump, lower energy photons (that is, longer wavelengths and lower frequencies) are sufficient to trigger the device. If a sample of silicon has some of its atoms replaced by phosphorus atoms (impurities), there will be extra electrons available for conduction. This is an example of an extrinsic semiconductor. Photoresistors are basically photocells.

Figure 1 : LDR

DC MOTOR DRIVER

MD293V1 is a simple and reliable DC motor driver. This motor driver can handle current up to 1 ampere. The design is dedicated for those who don't have much time in developing a small dc motor driver for projects application. It also helps developers to minimize the soldering error and maximize the time used for other applications.

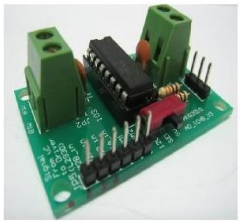


Figure 2 : DC Motor Driver

ADCON1=6;

Address constant

In IBM/360 (and through to present day z/Architecture), an **address constant** or **"adcon"** is an Assembly language data type whose value refers directly to (or "points to") another value stored elsewhere in the computer memory using its address. An address constant can be one, two, three or four bytes long (for IBM/360 architecture). It is defined using an assembler language **"DC"**statement using a type of **A** (or **V** if the adcon refers to an address outside of the current program module). If the adcon is less than three bytes it is usually used to hold a 16bit integer such as a length, a relative address or some index value. If the adcon is a 'V' type, it addresses an external program entry point, resolved by the link-editor when the external module is included with the module making the reference.

PROCEDURE A : SOFTWARE & HARDWARE INSTALLATION

1) Prepare MPLab software and PICKits to run the source code.
2) Hardware connection:

Super Bright LED – A2 White and 0V Black

LDR – A0 White, 5V Red and 0V Black

Motor :

Left motor connect to M1A/M1B

Right motor connect to M2A/M2B

Robot Base :

M1A – C0

M1B – C1

M2A – C2

M2B – C3

PWM1 – B0

PWM2 – B1

Motor Driver Supply – 0V/5V

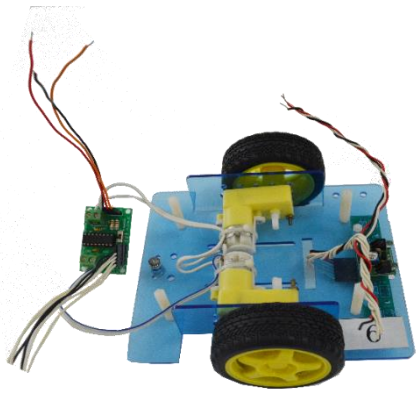


Figure 3 : Robot Base with Motor Driver

PROCEDURE B : DC MOTOR TABLE

1) Fill the table as below :

	MOTOR LEFT		MOTOR RIGHT	
	C0	C1	C2	C3
FORWARD				
REVERSE				

2) Create a source code with condition:
   a. LDR ‘ON’ - DC motor in forward direction
   b. Otherwise reverse direction
   c. Super Bright LED always ‘ON’
   d. #define XTAL_FREQ 20Mhz
   e. ADCON1=6;

EXERCISES

a. Built a source code to function as a sprinkle.

b. Draw a flow chart for Lab5.

Sprinkle – DC motor moving in one direction

PRACTICAL WORK : 6

TITLE : HARDWARE LCD INTERFACING

OBJECTIVES

a) To design and connect interfacing for LCD to be controlled by PIC18.
b) To determine and with limits, control the amount of delay to execute a program or subroutine.
c) To connect interfacing for the LCD module.

EQUIPMENTS

1. C18 Compiler
2. MPLAB IDE v8.85
3. Computer
4. PIC18F458 Training Kit
5. LCD Module

THEORY

MPLAB

MPLAB Integrated Development Environment (IDE) is a comprehensive editor, project manager and design desktop for application development of embedded designs using Microchip PIC micro and dsPIC microcontrollers. It is a free product of Microchip Inc. and is an effort to make source code development as smooth and comprehensive as possible. It is called an Integrated Development Environment, or IDE, because it provides a single integrated “environment” to develop code for embedded microcontrollers. MPLAB IDE provides a good platform for other compiler language tools to be integrated. MPLAB C17, MPLAB C18 and MPLAB C30 from microchip provide fully integrated, optimized code. Along with compilers from HI-TECH, IAR, micro Engineering Labs, CSC and Byte Craft, they are invoked by MPLAB IDE project managers to compile code. The initial use of MPLAB IDE is covered here. This section shows the method to install MPLAB IDE. It is followed by a simple tutorial to create project. Those who are unfamiliar with MPLAB IDE will get a basic understanding of using the system to develop an application. No previous knowledge is assumed, and comprehensive technical details of MPLAB IDE and its components are omitted in order to present the basic framework for using MPLAB IDE.

Programming and Compiler

C18 is a very widely used compiler for PIC18. It offers programming of almost all the PIC Microcontrollers, along with its appealing and friendly interface.

Setting input and output port

TRIS register is a vital part in order to control the PORT in PIC18. The value in the register will determine the port becoming an input or output port. For example, the value TRISB = FFH or TRISB = B '11111111' will set PORTB as input port and TRISB = 00H or TRISB = B '00000000' will turn PORTB as output port. Value 1 or 0 will determine the state of port. Sometimes, PORT can also be used as input and output ports. For example TRISB = F0H or TRISB = B '11110000' will make the pins RB0 – RB3 as output port and the rest pins RB4 – RB7 as input port. This technique is applicable for all PORTA, PORTC, PORTD and PORTE.

LCD Display

The 2x16 character LCD offers character display for embedded systems. It can be used to display numerical information, text messages and also special symbols. We can control a LCD using either 8 pins (8-bit interface) or 4 pins (4-bit interface), depending on the I/O pins that we have.

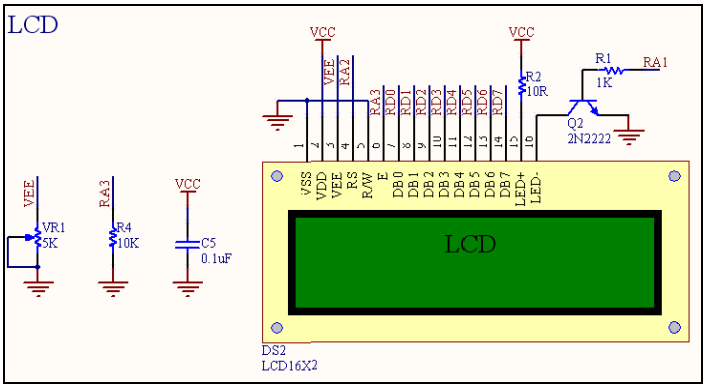


Figure 1 : Connection of a 2x16 character LCD

PROCEDURE A

1) Write the source code as below (**save as LCD.c)**;

```
#pragma config OSC = HS, OSCS = OFF
#pragma config PWRT= OFF, BOR=ON, BORV = 45
#pragma config WDT = OFF
#pragma config DEBUG = OFF, LVP =OFF, STVR =OFF
#include <p18f458.h>
#include "lcd16.h"

void main()
{

        TRISD=0;
        TRISC=0;
        EN=0;
        lcdinit();
        gotoXy(0,0);            // Lcd line
        prints("Hello World!!");

while(1)
{
}
}
```

2) By using MPLAB compile the source code with external header as figure 2 to get the hex file.

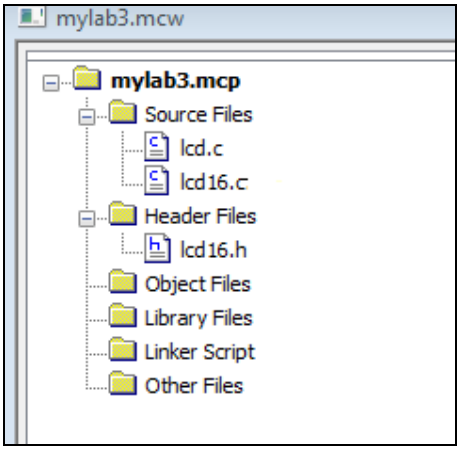


Figure 2 : Header File

3) Finally the hex file was created as

PROCEDURE B

1) Connect the circuit below by using the LCD module and Junior Pack.

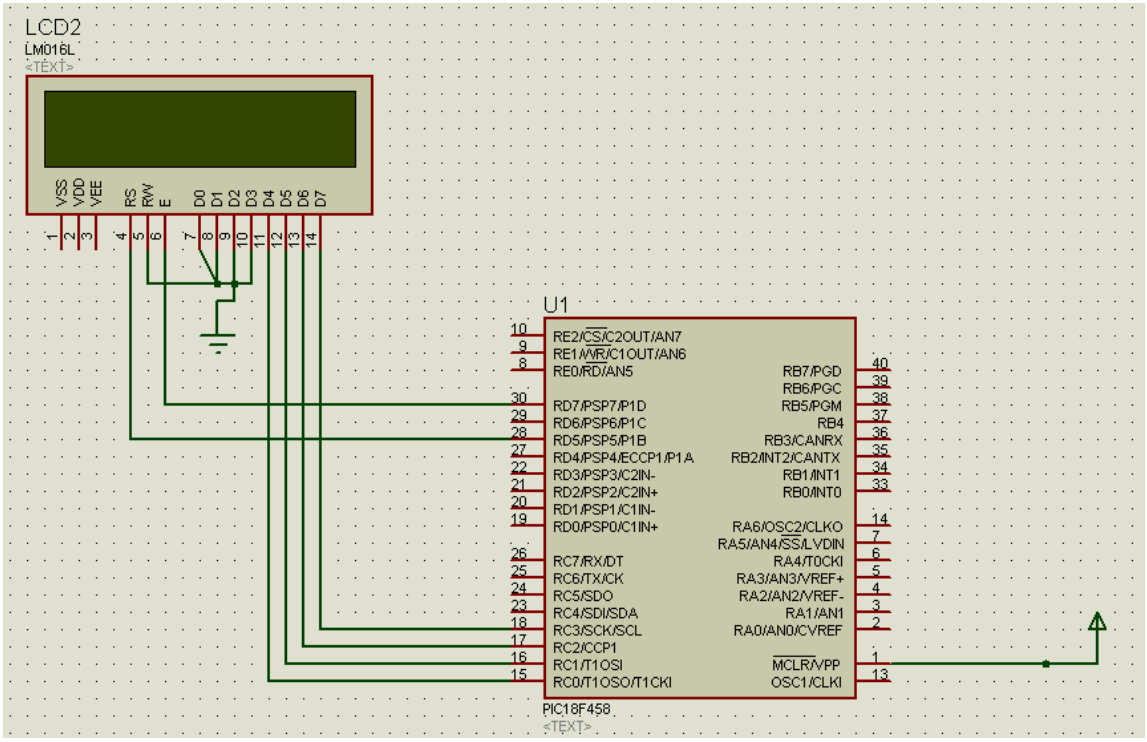


Figure 3 : PIC18F458 LCD Interfacing

2) Check connection of Vdd(5V) and Vss(0V) LCD Displays are connected to 5V and 0V target boards.
3) Connect a target board to the programmer.
4) Run the source code.
5) Observe the output at the LCD display.

EXERCISES

1) Write the source code to display output as below ;

*Embedded System ****DEC5052*****

2) Write a source code to display output as blinking sentences**;**

3) Write a source code to display output as running sentences.

PROJECT EXERCISE

PROJECT A : DELAY FUNCTION

1. Write the programme using C language based on the following specification :
    i. Use pic 18
    ii. Use delay.h
    iii. Use this configuration
        - FOSC=INTOSCIO_EC
        - WDT=OFF
        - PLLDIV=1
    iv. Set RB0 as an input and label it as SW
    v. Set PORTD as an output and label as LED

2. Use Timer0 and Timer1 interrupts to generate square waves on pin RB1 and RB7, respectively, while data is being transferred from PORTC to PORTD.

PROJECT B : SEND VALUE XX TO PORT X

1. Write a C18 program to send hex values for ASCII characters of 0,1,2,3,4,5,A,B,C, and D to Port B.

2. LEDs are connected to bits in Port B and Port C. Write a C18 program that shows the count from 0-FFH on LEDs.

PROJECT C : DOOR SENSOR

A door sensor is connected to RB1 pin, and a buzzer is connected to RC7. Write a C18 program to monitor the door sensor, and when it opens, sound the buzzer.

PROJECT D : TIMER INTERRUPT

Two interrupts: (1) PORTC counts up every time Timer0 overflows. It uses the 16-bit mode of Timer0 with the largest prescale possible; (2) a 1 Hz pulse is fed into Timer1 where Timer1 is used as a counter and counts up. Whenever the count reaches 200, it will toggle pin RB6. Write a C program to solve this problem.

PROJECT E : PORTB-CHANGE INTERRUPT (RB4-RB5)

SW1 and SW2 connected to pins RB4 and RB5 respectively. The activation of SW1 and SW2 will result in changing the state of LED1 and LED2 respectively. Write a C program to solve this problem.

PROJECT F : EXTERNAL HARDWARE INTERRUPT

1. To toggle the LED again, the INT0 pulse must be brought back LOW and then forced HIGH to create a rising edge to activate the interrupt.

2. Pin RB1(INT1) is connected to a pulse generator and the pin RB7 is connected to an LED. The program will toggle the LED on the falling edge of the pulse. In other words, the LED is turned on and off at the same rate as the pulses are applied to the INT1 pin.

3. Write C program to monitor the status of Switch (SW) and perform the following:
   (a) If SW=0,the DC motor moves with 50% duty cycle pulse.
   (b) If SW=1, the DC motor moves with 25% duty cycle pulse.

PROJECT G : KEYPAD

Write a C program to scan the keypad buttons at column 1 (button 1, 4, 7 and *). Pressed button will switch ON the LED base on the binary numbers of the button location. Ie. Button no. 1 = 0001, button no. 4 = 0100, button no. 7 = 0111, and button * =1111.

REFERENCES

1. Muhammad Ali Mazidi, Rolin D.Mckinlay dan Danny Causey (2008). *PIC Microcontroller and Embedded Systems Using Asembly and C For PIC18*. Pearson Prentice Hall.
2. M. M. Asad, R. Hassan, and F. Sherwan (2015). Design and Development of a PIC Microcontroller Based Embedded System Trainer Panel for Electrical Personnel Training. *Nature and Science 2014;12(8)*, 129-133. http://www.sciencepub.net/nature
3. Alba-Flores, R. (2007). Laboratory Enhancements for Improving Embedded Systems Education. *Proceeding of the 2007 American Society for Engineering Educational Annual Conference & Exposition*. American Society for Engineering Education.
4. http://www.astanadigital.com
5. http://www.cytron.com.my/
6. http://www.microchip.com