**[1](#_bookmark0)**

LABORATORY MANUAL

**PROGRAMMING SKILL DEVELOPMENT LAB**

SE-IT SEMESTER-II

TEACHING SCHEME EXAMINATION SCHEME

Theory: 02 Hrs/Week Practical: 25 Marks

TW: 25 Marks

DEPARTMENT OF INFORMATION TECHNOLOGY

2020-2021

1

|  |  |
| --- | --- |
| **Sr. No.** | **Name of Experiment** |
| **Group A** | |
| **1** | Study of Embedded C programming language (Overview, syntax, One simple program like addition of two numbers). |
| 2 | Write an Embedded C program to add array of n numbers. |
| 3 | Write an Embedded C program to transfer elements from one location to another for following:   1. Internal to internal memory transfer 2. Internal to external memory transfer |
| 4 | Write an Embedded C menu driven program for:   1. Multiply 8-bit number by 8-bit number 2. Divide 8-bit number by 8-bit number |
| 5 | Write an Embedded C program for sorting the numbers in ascending and descending order. |
| **Group B** | |
| 6 | Write an Embedded C program to interface PIC 18FXXX with LED & blinking it using specified delay. |
| 7 | Write an Embedded C program for Timer programming ISR based buzzer on/off. |
| 8 | Write an Embedded C program for External interrupt input switch press, output at relay. |
| 9 | Write an Embedded C program for LCD interfacing with PIC 18FXXX. |
| **Group C** | |
| 10 | Write an Embedded C program for Generating PWM signal for servo motor/DC motor. |
| 11 | Write an Embedded C program for PC-to-PC serial communication using UART. |
| 12 | Write an Embedded C program for Temperature sensor interfacing using ADC & display on LCD. |
| **Group D:** | |
| 13 | Study of Arduino board and understand the OS installation process on Raspberry- pi. |
| 14 | Write simple program using Open-source prototype platform like  Raspberry-Pi/Beagle board/Arduino for digital read/write using LED and switch Analog read/write using sensor and actuators. |

# GROUP A: ASSIGNMENTS

## Experiment No: 01

**Title:** Study of Embedded C programming language (Overview, syntax, One simple program like addition of two numbers).

**Objective:** To learn embedded C programming and PIC18FXXXmicrocontrollers.

**Outcome:** On completion of this Assignment student will be able to

* Understand the concepts of embedded C programming
* Develop and Execute embedded C program to perform array addition

**Pre-requisites:** Computer Organization and Architecture

**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXXmicrocontroller kit

## Theory:

Embedded C programming plays a key role to make the microcontroller run & perform the preferred actions. At present, we normally utilize several electronic devices like mobile phones, washing machines, security systems, refrigerators, digital cameras, etc. The controlling of these embedded devices can be done with the help of an embedded C program. For example, in a digital camera, if we press a camera button to capture a photo then the microcontroller will execute the required function to click the image as well as to store it.

Embedded C programming builds with a set of functions where every function is a set of statements that are utilized to execute some particular tasks. Both the embedded C and C languages are the same and implemented through some fundamental elements like a variable, character set, keywords, data types, declaration of variables, expressions, statements. All these elements play a key role while writing an embedded C program.

In embedded system programming C code is preferred over other language. Due to the following reasons:

* Easy to understand
* High Reliability
* Portability
* Scalability

***Steps to Build an Embedded C Program***

There are different steps involved in designing an embedded c program like the following.

* + Multiline Comments Denoted using /\*……\*/
  + Single Line Comments Denoted using //
  + Preprocessor Directives #include<…> or #define
  + Global Variables Accessible anywhere in the program
  + Function Declarations Declaring Function
  + Main Function Main Function, execution begins here

{

Local Variables Variables confined to main function

Function Calls Calling other Functions

Infinite Loop Like while(1) or for(;;)

Statements . . . . .

….

….

}

o Function Definitions Defining the Functions

{

Local Variables Local Variables confined to this Function

Statements . . . . .

….

….

}

## Main Factors of Embedded C Program

The main factors to be considered while choosing the programming language for developing an embedded system include the following.

## Program Size

Every programming language occupies some memory where embedded processor like microcontroller includes an extremely less amount of random-access memory.

## Speed of the Program

The programming language should be very fast, so should run as quickly as possible. The speed of embedded hardware should not be reduced because of the slow-running software.

## Portability

For the different embedded processors, the compilation of similar programs can be done.

* Simple Implementation
* Simple Maintenance
* Readability

## Advantages

* It is very simple to understand.
* It executes simply a single task at once
* The cost of the hardware used in the embedded c is typically so much low.
* The applications of embedded are extremely appropriate in industries.
* It takes less time to develop an application program.
* It reduces the complexity of the program.
* It is easy to verify and understand.
* It is portable from one controller to another.

## Disadvantages

* At a time, it executes only one task but can’t execute the multi-tasks
* If we change the program then need to change the hardware as well
* It supports only the hardware system.
* It has a scalability issue
* It has a restriction like limited memory otherwise compatibility of the computer.

## Applications of Embedded C Program

* Embedded C programming is used in industries for different purposes
* The programming language used in the applications is speed checker on the highway, controlling of traffic lights, controlling of street lights, tracking the vehicle, artificial intelligence, home automation, and auto intensity control.

## What is MPLAB?

MPLAB is a software program that runs on your PC to provide a development environment for your embedded system design. In other words, it is a program package that makes writing and developing a program easier. It could best be described as developing environment for a standard program language that is intended for programming microcontrollers.

Get started to **MPLAB X Programming IDE. Step1:** Creating a new project

* + Go to the File Tab.
  + Click on New Project.
  + Step1: Choose Project:
  + Select: Microchip Embedded -> Standalone Project. Click Next.

**Step2**: Select Device:

* + Select: Family -> Advanced 8 Bit MCU (PIC18).
  + Select: Device: PIC18F4550. Click Next.

**Step3:** Select Tool: Simulator. Click Next. **Step4**: Select Compiler ->XC8. Click Next. **Step5**: Select Project Name and Folder.

* + Give Project Name.
  + Select project Location using Browse Button.
  + Uncheck Set as main project option.
  + Click Finish.

**Step6:** Creating a new Source file and Header File.

* + Go to the Project location in the Project window.
  + Click the + sign to open the project space.
  + Right Click on the Source Files folder (for a C file) and Header files (for a .h file).
  + New - > C Source file / or C Header File.

**Step7:** Opening an existing project.

* + Go to the File Tab.
  + Select Open Project.

Browse to the location and select the project name.X file (project file). Click on Open Project

**Input:** Two numbers

**Output:** Addition of given numbers

**Conclusion:** Based on the understanding of the basic concepts of Embedded C

Programming along with its environment setup used in a simple program for addition of two numbers.

**Experiment No: 02**

**Title:** Write an Embedded C program to add array of n numbers.

**Objective:** To calculate the sum of the array elements.

**Outcome:** On completion of this Assignment student will be able to

* Understand the concepts of embedded C programming
* Develop and Execute embedded C program to perform array addition

**Pre-requisites:** Fundamental knowledge of Microcontroller, Logic and knowing the basics of C language

**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXXmicrocontroller kit

## Theory:

There is actually not much difference between C and Embedded C apart from few extensions and the operating environment. Both C and Embedded C are ISO Standards that have almost same syntax, datatypes, functions, etc.

Embedded C is basically an extension to the Standard C Programming Language with additional features like Addressing I/O, multiple memory addressing and fixed-point arithmetic, etc. C Programming Language is generally used for developing desktop applications, whereas Embedded C is used in the development of Microcontroller based applications.

C arrays are declared in the following form type name [number of elements]; For example, if we want an array of five integers , we write in C:

int numbers[5];

For a five character array char letters[5];

type name [number of elements]={comma-separated values}

For example, if we want to initialize an array with five integers, with 1, 3, 5, 0, 9, as the initial values: int number[5]={1,3,5,0,9};

Let’s see the logic to calculate the sum of the array elements. Suppose **arr** is an integer array of size N (arr[N] ), the task is to write the C Program to sum the elements of an array.

## Logic to calculate the sum of the array elements:

1. Create an intermediate variable ‘sum’.
2. Initialize the variable ‘sum’ with 0.
3. To find the sum of all elements, iterate through each element, and add the current element to the sum.

//Logic within the loop sum = sum + arr[i];

**Input:** Array of size N (arr[N] )

**Output:** Sum of the elements of an array

**Conclusion:** We have implemented sum of elements of an array using embedded C programming.

**Experiment No: 03**

**Title:** Write an Embedded C program to transfer elements from one location to another for following:

1. Internal to internal memory transfer
2. Internal to external memory transfer

**Objective:** To learn memory transfer using embedded C programming and PIC18FXXXmicrocontrollers.

**Outcome:** On completion of this Assignment student will be able to

– Understand the concepts of memory transfer using embedded C programming

**Pre-requisites:** Computer Organization and Architecture

**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXXmicrocontroller kit

## Theory:

Memory types in microcontrollers Architecture

The microcontrollers units (MCUs) consist of three types of memory.

* 1. Program Memory
  2. Data Memory
  3. Data EEPOM

Program Memory type

This is common which have all the microcontroller and its purposes is to store the instructions.it consist of further four different types of memory.

1. ROM (Read only memory)
2. EPROM (Erasable programmable read only memory)
3. OTP (On time programmable)
4. FLASH EEPROM (Electrical erasable programmable read only memory)

ROM

In microcontrollers first type memory is ROM and during the manufacturing process once the program codes are set in ROM that can’t be changed after the manufacturing process, therefore it is called read only memory mean just read the code but can’t be changed. Due to this reason the microcontrollers which have the ROM memory are considers best for that applications where there is no need of program change only need of program read. These microcontrollers are less expensive as compared to the microcontrollers which have the OTP or FLAS programmable memory and these are ordered in large quantities.

EPROM

The second type is erasable programmable read only and this is used in two different type of packages. When EPROM is used in ceramic package with quartz window then microcontroller can be erased the program many times by using ultraviolet eraser and erase time depends upon the intensity of light. Normally the erase time is in between 5 and 30 minutes. In this the microcontroller can also reprogrammed the program. It is very expensive due to the high cost of the windowed ceramic package.

OTP

The one-time programmable memory used the same type of die as the EPROP windowed packaged devices. Its packaging makes it unique. These microcontrollers are in an opaque plastic packing and its program can’t be erase through ultraviolet light. The OTP devices are first transfer to customer side then these are programmed therefore these devices are called one time programmable.

Flash EPROM

This the type which provides the alternate flexibility because its program can be erased electrically and also reprogram in few seconds. It’s no need of any ultraviolet light to erase the program. Once the program is erased the program can reprogram with new code. The devices which have the flash memory can also be self-program by using some special sequence of instructions. These devices also contain a small amount non-volatile data EPROM and that can be written thousands of time. In these devices “F” is denoted by part number

**Input**: Elements in a array.

**Output:** Array after transferred elements

**Conclusion:** We have implemented transfer of elements from one location to another.

**Experiment No: 04**

**Title:** Write an Embedded C menu driven program for:

1. Multiply 8-bit number by 8-bit number
2. Divide 8-bit number by 8-bit number

**Objective:** To learn arithmetic operation i.e., multiplication and division in embedded C programming and PIC18FXXXmicrocontrollers.

**Outcome:** On completion of this Assignment student will be able to

* + Understand the concepts of embedded C programming
  + Develop and Execute embedded C program to perform multiplication and division

**Pre-requisites:** Fundamental knowledge of Microcontroller, Logic and knowing the basics of C language

**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXXmicrocontroller kit

## Theory:

Embedded C Programming Language, which is widely used in the development of Embedded Systems, is an extension of C Program Language. The Embedded C Programming Language uses the same syntax and semantics of the C Programming Language like main function, declaration of datatypes, defining variables, loops, functions, statements, etc.

The extension in Embedded C from standard C Programming Language include I/O Hardware Addressing, fixed point arithmetic operations, accessing address spaces, etc.

## C examples – with standard arithmetic operators

int i, j, k; // 32-bit signed integers uint8\_t m,n,p; // 8-bit unsigned numbers i = j + k; // add 32-bit integers

m = n - 5; // subtract 8-bit numbers j = i \* k; // multiply 32-bit integers m = n / p; // quotient of 8-bit divide

m = n % p; // remainder of 8-bit divide

i = (j + k) \* (i – 2); //arithmetic expression

\*, /, % are higher in precedence than +, - (higher precedence applied 1st) Example: j \* k + m / n = (j \* k) + (m / n)

**Input:** 8-bit number stored in internal memory locations.

**Output:** Result of 8- bit by 8-bit multiplication and division stored at internal memory locations

**Conclusion:** Arithmetic operations like multiplication and division of 8-bit number with 8-bit number are studied here

**Experiment No: 05**

**Title:** Write an Embedded C program for sorting the numbers in ascending and descending order.

**Objective:** To learn sorting using embedded C programming and PIC18FXXXmicrocontrollers.

**Outcome:** On completion of this Assignment student will be able to

* + Understand the concepts of sorting using embedded C programming

**Pre-requisites:** Computer Organization and Architecture

**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit

**Theory:**

**ALGORITHM**:

Step I: Initialize the number of elements counter. Step II: Initialize the number of comparisons counter.

Step III: Compare the elements. If first element < second element goto Step VIII Else go to step V.

Step IV: Swap the elements.

Step V: Decrement the comparison counter.

Step VI: Is count = 0? if yes go to step VIII else go to step IV. Step VII: Insert the number in proper position.

Step VIII: Increment the number of elements counter.

Step IX: Is count = N? If yes, go to step XI else go to step II Step X: Store the result.

Step XI: Stop

## Input: Numbers in a array

**Output: Sorted array of** numbers in ascending or descending order.

**Conclusion:** We have implemented sorted array of numbers in ascending or descending order.

# GROUP B: ASSIGNMENTS

## Experiment No: 06

**Title:** Write an Embedded C program to interface PIC 18FXXX with LED & blinking it using specified delay.

**Objective:** Connect LED to the microcontroller and it should start blinking.

**Outcome:** On completion of this Assignment student will be able to

* + Understand the concepts writing PIC program using embedded C programming and write the .exe code on the kit or use the Proteus software to blink the LED (real or virtual)

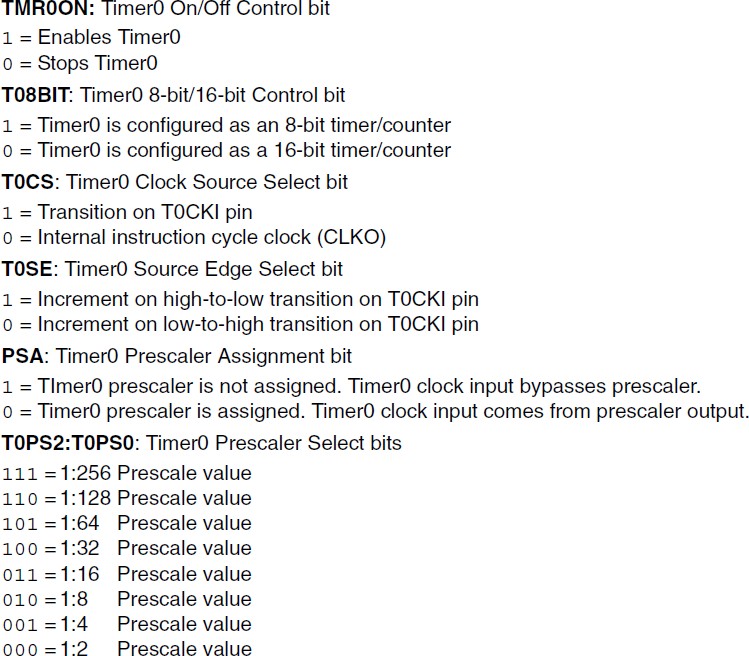
**Pre-requisites:** Embedded C programming, MPLAB, Proteus

**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit, Proteus, LED

**Theory:** Timer is used to control the output on a bit of a port.

There are 4 timers in PIC18. Each have corresponding timer registers TMRxH and TMRxL where x is the timer number that ranges from 0 to 3

There is a control register correspond to every timer TxCON. In this program Timer 0 is used. Following is the T0CON register.



The program is written such that a LED is connected to a pin of any port (PORTB) and that pin is toggled after a cycle controlled by the timer. This is repeated for infinite time. Thus when the port bit is 0, LED is off and when it is 1, LED is on. So the LED blinks after every cycle.

## ALGORITHM:

Step1: Reset TRISB so that PORTB is in output mode Step2: Reset PORTB

Step3: Call the delay Step4: Set PORTB Step5: goto Step3

**Input:** make all the required connections

**Output:** LED starts blinking with an equal interval

**Conclusion:** The student is able to understand the working of the timer, LED, and ports.

**Experiment No: 07**

**Title:** Write an Embedded C program for Timer programming ISR based buzzer on/off.

**Objective:** Connect buzzer to the microcontroller and it should be on or off as per the user input.

**Outcome:** On completion of this Assignment student will be able to

* + Understand the concepts writing PIC program using embedded C programming and write the .exe code on the kit or use the Proteus software to switch the buzzer on or off (real or virtual)

**Pre-requisites:** Embedded C programming, MPLAB, Proteus

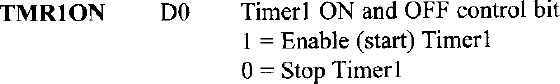
**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit, Proteus, Buzzer

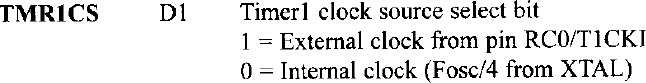
**Theory:** Timer is used to generate the delay for which the buzzer should be on. In this program Timer 1 is used. Following is the T1CON register.

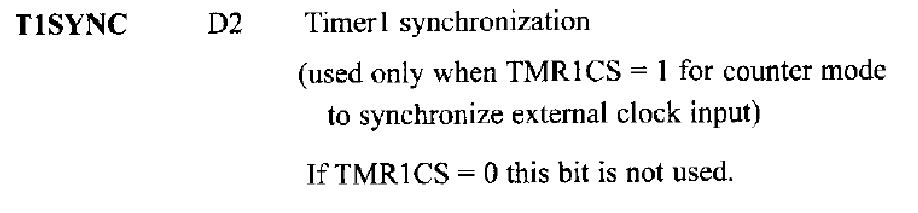


D7 D6 D5 D4 D3 D2 D1 D0

not used

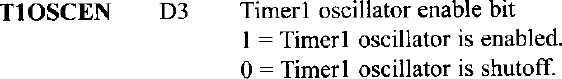


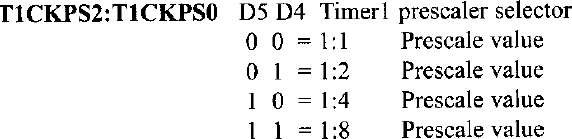


1 = Do not synchronize external clock input

0 = Synchronize external clock input

1 = Do not synchronize external clock input 0 = Synchronize external clock input





The program is written such that a buzzer is connected to a pin of any port (PORTB) and that pin is set when the switch is pressed. The duration for which the pin should be set is controlled by the timer. This is repeated for infinite time.

## ALGORITHM:

Step1: Reset TRISB so that PORTB is in output mode Step2: Reset PORTB

Step3: check the switch is pressed Step4: if yes then

Step5: Set PORTB Step6: Call the delay

Step7:end if Step8: goto Step3

**Input:** make all the required connections

**Output:** The buzzer is on when the switch is pressed.

**Conclusion:** The student is able to understand the working of the timer, buzzer, and ports.

**Experiment No: 08**

**Title:** Write an Embedded C program for External interrupt input switch press, output at relay.

**Objective:** To understand the working of interrupts in PIC18

**Outcome:** When PIC18 is interrupted, using RB0, the data at PORTD will be toggled

**Pre-requisites:** Embedded C programming, MPLAB, Proteus

**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit

## Theory:

Sometimes External devices are connected with [**microcontroller**](https://microcontrollerslab.com/8051-microcontroller/). If that external device has to send some information to microcontroller, then microcontroller needs to know about this situation to get that information. An example of such an external device is the [**digital thermometer**](https://microcontrollerslab.com/digital-thermometer-pic-mcp9700/). It measures the [**temperature**](https://microcontrollerslab.com/temperature-sensor-using-pic16f877a-microcontroller/) and at the end of measurements transmits results to the microcontroller. Now the purpose of this article to explain the fact that how does the microcontroller knows to get the required information from an external device.

Types of interrupts

There are two methods of communication between the microcontroller and the external device:

* By using Polling
* By using Interrupts

INTERRUPTS

Interrupt is the signal which is sent to the microcontroller to mark the event that requires immediate attention. This signal **requests** the microcontroller to stop to perform the current program temporarily **time** to execute a special code. It means when external device finishes the task imposed on it, the microcontroller will be notified that it can access and receive the information and use it.

INTERRUPT SOURCES in microcontrollers

The request to the microcontroller to stop to perform the current program temporarily can come from various sources:

* Through external hardware devices like pressing specific key on the keyboard, which sends Interrupt to the microcontroller to read the information of the pressed key.
* During execution of the program, the microcontroller can also send interrupts to itself to report an error in the code. For example, division by 0 will causes an interrupt.
* In the multi-processor system, the microcontrollers can send interrupts to each other to communicate. For example, to divide the work between them they will send signals between them.

INTERRUPT TYPES in pic microcontrollers

There are 2 types of interrupts for PIC microcontroller that can cause break.

**Software Interrupt:** It comes from a program that is executed by microcontroller or we can say that it is generated by internal peripherals of the microcontroller and requests the processor to hold the running of program and go to make an interrupt.

**Hardware Interrupt:** These interrupts are sent by external hardware devices at certain pins of [microcontroller](http://www.engineersgarage.com/microcontroller).

Following interrupts sources are present in PIC18F452

* + [**Timer over interrupt**](https://microcontrollerslab.com/timers-pic-microcontroller-delay/)
  + Pins RB0, RB1, RB2 for external hardware interrupts (INT0, INT1, INT2)
  + PORTB Change interrupts (any one of the upper four Port B pins. RB4-RB7)
  + ADC ([**analog-to-digital converter**](https://microcontrollerslab.com/analog-to-digital-adc-converter-working/)) Interrupt
  + CCP (compare capture pulse-width-modulation) Interrupt
  + [**Serial communication’s USART**](https://microcontrollerslab.com/serial-communication-using-pic16f877a-microcontroller/) interrupts (receive and transmit)
  + Reset, Brown-Out Reset, Watch-dog Reset, Power On Reset
  + Parallel Port Read/Write Interrupt
  + Master Synchronous Serial Port Interrupt
  + Data [**EEPROM**](https://microcontrollerslab.com/eeprom-working-interfacing-with-microcontroller/) Write Complete Interrupt

REGISTER CONFIGURATION for external interrupt

These are the registers for interrupt operation and minimum 1 register can be used to control the interrupt operation in PIC18F452 which are:

* + RCON (Reset Control Register)
  + INTCON, INTCON2, INTCON3 (Interrupt Control Registers)
  + PIR1, PIR2 (Peripheral Interrupt Request Registers)
  + PIE1, PIE2 (Peripheral Interrupt Enable Registers)

## RCON Register:

* + Reset control register
  + IPEN bit to enable interrupt priority scheme, 1= enable priority level on interrupts
  + Other bits used to indicate the cause of reset RI (Reset Instruction flag), TO (Watchdog Time Out flag), PD (Power on Detection flag), POR (Power on Reset status) and BOR (Brown Out Reset status bit)

## INTCON Register:

* + 3 Interrupt control registers INTCON, INTCON2, INTCON3
  + Readable and writable register which contains various enable and flag bits
  + Interrupt flag bits get set when an interrupt condition occurs
  + Contain enable, priority and flag bits for external interrupt, port B pin change and TMR0 overflow interrupt

## PIE Register:

* + Peripheral Interrupt Enable register
  + May be multiple register (PIE1, PIE2), depending on the number of peripheral interrupt sources
  + Contain the individual bits to enable/disable Peripheral interrupts for use

## PIR Register:

* + Peripheral Interrupt Flag register
* May be multiple register (PIR1, PIR2), depending on the number of peripheral interrupt sources
* Contain bits to identify which interrupt occurs (flags)
* Corresponding bits are set when the interrupt occurred EXTERNAL INTERRUPT registers setting

INTCON registers are just used to configure the external PIC interrupts.

## INTCON REGISTER:



**GIE: Global Interrupt Enable**

This bit is set high to enable all interrupts of PIC18F452. 1 = Enable all interrupts

0 = Disable all interrupts

## PEIE: Peripheral Interrupt Enable

This bit is set high to enable all the peripheral interrupts (Internal interrupts) of the microcontroller.

1 = Enable all peripheral interrupts 0 = Disable all peripheral interrupts

## T0IE: TMR0 Overflow Interrupt Enable

This bit is set high to enable the External Interrupt 0. 1 = Enable TMR0 overflow interrupt

0 = Disable TMR0 overflow interrupt

## INTE: INT External Interrupt Enable

This bit is set high to enable the external interrupts. 1 = Enables the INT external interrupt

0 = Disables the INT external interrupt

## RBIE: RB Interrupt Enable

This bit is set high to enable the RB Port Change interrupt pin. 1 = Enables the RB port change interrupt

0 = Disables the RB port change interrupt

## T0IF: TMR0 Overflow Interrupt Flag

1 = TMR0 register has overflowed (it must be cleared in software) 0 = TMR0 register has not overflowed

## INTF: INT External Interrupt Flag

1 = The INT external interrupt occurred (it must be cleared in software) 0 = The INT external interrupt did not occur

## RBIF: RB Port Change Interrupt Flag

1 = At least one of the RB7:RB4 pins changed the state (must be cleared in software) 0 = None of RB7:RB4 pins have changed the state

## ALGORITHM:

In this program, we configure the External Interrupt 0 (INT0). PORT D is used as output port and it is monitored through a set of 8 LEDs. Reset is given through pin 1. Push button is connected to RB0 for external interrupt source.

Step1: Enable the External Interrupt 0 by setting INT0IE bit high (INTCON=0x10).

Step2: Set the interrupt on falling edge by setting the INTEDG0 in INTCON2 to zero (INTCON2=0)

Step3: Set the Global Interrupt Enable in INTCON to high (INTCON.GIE=1) Step4: Initialize PORTD with certain value (LATD=0xAA)

Step5: Write the ISR (Interrupt Service Routine) for the interrupt Step6: Clear the INT0IF bit of INTCON (INTCON.INT0IF=0) Step7: Invert or toggle the value at PORTD (LATD=~LATD) Step8: go to step3.

**Input:** press the push button

**Output:** the LED’s connected to the PORTD are toggled when the button is pressed.

**Conclusion:** The student is able to understand the working of the interrupts and able to write the ISR (Interrupt Service Routine).

**Experiment No: 09**

**Title:** Write an Embedded C program for LCD interfacing with PIC 18FXXX.

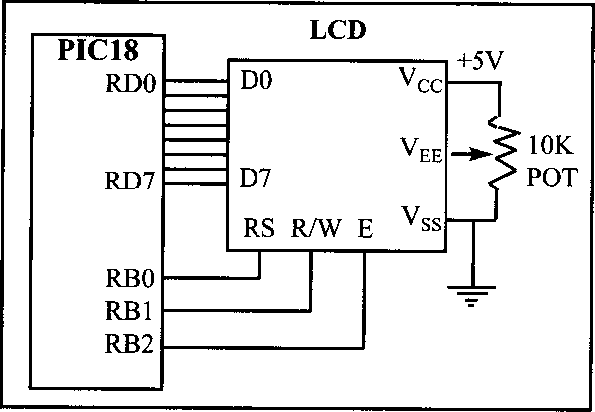
**Objective:** To study LCD connection with PIC18 and communication.

**Outcome:** On completion of this Assignment student will be able to understand the communication from PIC18 with LCD

**Pre-requisites:** Embedded C programming, MPLAB, Proteus

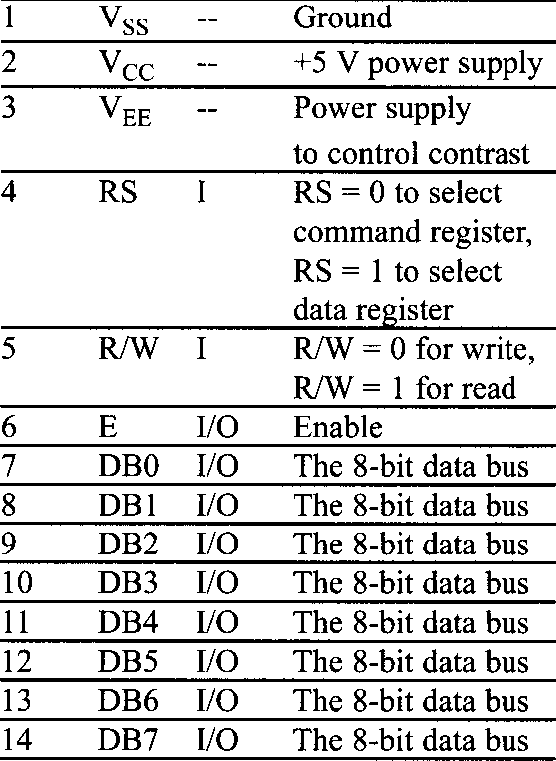
**Lab facility:** MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit, LCD, Proteus

## Theory:

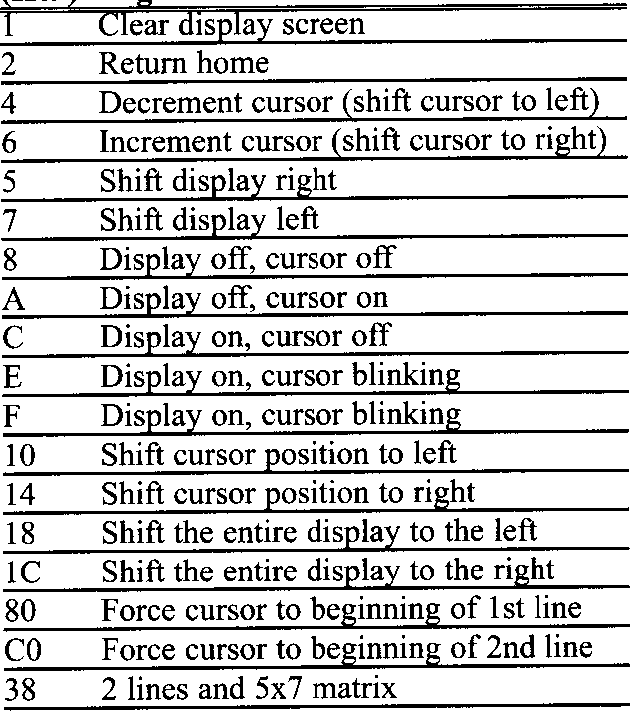


The above figure shows the connection of LCD with PIC18.

The resistor R1 is used for giving the contrast to the LCD. The crystal oscillator of 12 MHz is connected to the OSC1 and OSC2 pins of Pic microcontroller PIC18F4550 for system clock. The capacitor C2 and C3 will act filters to the crystal oscillator. You can use different ports or pins for interfacing the LCD before going to different ports please check the data sheet whether the pins for general purpose or they are special function pins.



The above figure shows pin description of LCD.



The above figure shows the hex codes and their corresponding instruction to LCD register. The characters are sent ot the LCD without checking the busy flag. We need to wait 5-10 ms between issuing each character to the LCD. In programming an LCD we need a long delay for the power-up process.

## Initializing the LCD function:

lcdcmd(0x38);//Configure the LCD in 8-bit mode,2 line and 5×7 font lcdcmd(0x0C);// Display On and Cursor Off

lcdcmd(0x01);// Clear display screen lcdcmd(0x06);// Increment cursor

lcdcmd(0x80);// Set cursor position to 1st line,1st column

## Sending command to the LC:

* rs=0; Register select pin is low.
* rw=0; Read/write Pin is also for writing the command to the LCD.
* en=1;enable pin is high.

## Sending data to the LCD:

* rs=1; Register select pin is high.
* rw=0; Read/write Pin is also for writing the command to the LCD.
* en=1; enable pin is high.

## ALGORITHM:

Step1: Reset TRISD Step2: Reset TRISE

Step3: store the messages in two character arrays Step4: store 0x0F in ADCON1 register

Step5: initialize LCD Step6: call the delay Step7: write the first string Step8: call the delay

Step9: move the cursor to the second line in LCD Step10: write the second string

**Input:** Make all the connections and start

**Output:** two input strings are displayed on the LCD.

**Conclusion:** The student is able to understand the working of the PIC18 with LCD and able to write the codes for the initialization of LCD, writing to LCD etc.