****

Department of Electronics & Telecommunication Engineering

**AY: 2023-24**

**SEMESTER: I**

**LAB MANUAL**

**SUB:** MICROCONTROLLERS

**SUBJECT TEACHER:** Dr. PRACHI P. VAST

**INDEX**

|  |  |
| --- | --- |
| **Sr. No** | **Title of Experiment** |
| 1 | Parallel port interacting of LEDS—Different programs (flashing, Counter, BCD, HEX, Display of Characteristic) |
| 2 | Interfacing of Multiplexed 7-segment display (counting application) |
| 3 | Waveform Generation using DAC |
| 4 | Write a program for interfacing button, LED, relay & buzzer as follows. |
| 5 | Interfacing of LCD to PIC 18FXXXX. |
| 6 | Generate square wave using timer with interrupt. |
| 7 | Interfacing serial port with PC both side communication. |
| 8 | Generation of PWM signal for DC Motor control. |

***Experiment No :1***

***Title:*** Parallel port interacting of LEDS—Different programs (flashing, Counter, BCD, HEX, Display of Characteristic)

Roll No: \_ \_ \_ Batch: \_ \_ \_

Date of Performance:\_ \_ /\_ \_/\_ \_ \_ \_

Date of Assessment:\_ \_ /\_ \_/\_ \_ \_ \_

**TITLE:**Interfacing of output (LED) devices to the ports.

**AIM**: Write an embedded C program of 8051 microcontroller for interfacing LEDs as output device to Port2 as follows,

1. Flashing LEDs.

2.As a Counter.

3.To display BCD data.

4. To display HEX data.

5. To display Character.

**HARDWARE**:STK 8051 kit, PC, RS-232 Cable with USB

**SOFTWARE**: KeilIDE, Flash Magic to download the program.

**THEORY :**

The general View of LED is shown in figure 1.

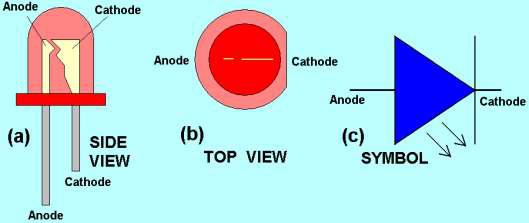


Fig. 1: LED Symbol and view

Current flows from anode to cathode. A 330Ω register is connected between port lines and LED. Resistance of an LED is almost zero. Hence current flowing through LED is I=V/R which is approximately in mA.

In this experiment, the LEDs are connected to port of the microcontroller.

The LEDs will glow alternatively with a time delay.

**PROCEDURE:**

• Compile the written code if errors are present then debug the code, modify it, save it and recompile the code using the build all option. After build succeeded and Hex file generated.

• Download and run this program.

**CONNECTION :**

****

**Fig.2: LED connections at port2 (P2.0-P2.7)**

**PROTEUS INTERFACING:**

## 

**PROGRAM :**

1. **8051 PROGRAM for Flashing LEDs.**
2. **8051 PROGRAM LED as a Counter.**
3. **8051 PROGRAM to display BCD data.**
4. **8051 PROGRAM TO display HEX data.**
5. **8051 PROGRAM TO Display Character.**

***Experiment No :2***

***Title:*** Interfacing of Multiplexed 7-segment display (counting application)

Roll No: \_ \_ \_ Batch: \_ \_ \_

Date of Performance:\_ \_ /\_ \_/\_ \_ \_ \_

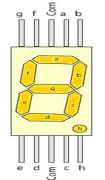
Date of Assessment:\_ \_ /\_ \_/\_ \_ \_ \_

**TITLE :**Interfacing of Multiplexed 7-segment display (counting application)

**HARDWARE** :STK 8051 kit, PC, RS-232 Cable with USB

**SOFTWARE**: KeilIDE, Flash Magic to download the program.

**THEORY:**A seven segment display is the most basic electronic display device that can display the digits from 0-F (hexadecimal numbers). The seven segment pins (a,b,c,d,e,f,g) plus the decimal point of a common anode display are connected to port pins of 8051 via current limiting resistors (220 Omega). The program is developed using software to display hexadecimal numbers 0-F on the display.



**Fig.1 Seven Segment Display**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Number** | **h** | **g** | **f** | **e** | **d** | **c** | **b** | **a** |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 4 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 8 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

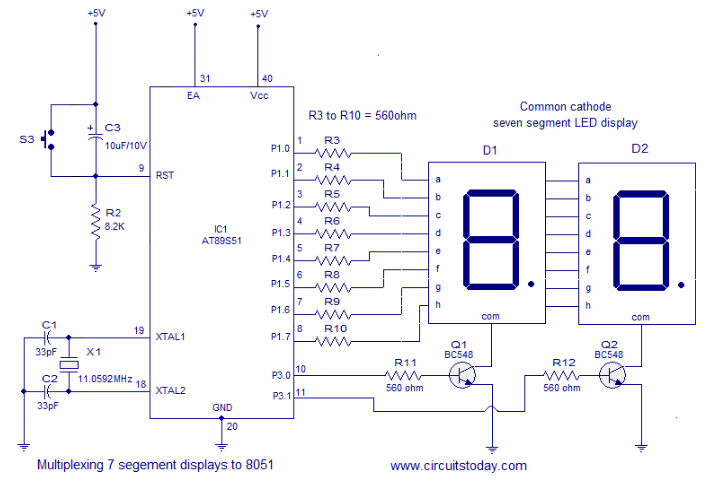
**Table 1: Segment Control table for displaying characters 0 to 9**

**PROCEDURE:**

• Compile the written code if errors are present then debug the code, modify it, save it and recompile the code using the build all option. After build succeeded and Hex file generated.

• Download and run this program.

**INTERFACING DIAGRAM:**



**PROGRAM :**

***Experiment No : 3***

***Title:*** Waveform Generation using DAC

Roll No: \_ \_ \_ Batch: \_ \_ \_

Date of Performance:\_ \_ /\_ \_/\_ \_ \_ \_

Date of Assessment:\_ \_ /\_ \_/\_ \_ \_ \_

**TITLE:**Waveform generation using DAC

**AIM:**Write a program to generate the sinusoidal wave

**EQUIPMENTS REQUIRED:**P89V51RD2 microcontroller board, DAC Board ,CRO , RS232 cable, FRC cables, Power adaptor etc

**S/W PACKAGES AND H/W USED:** Keil version 4 software, Flash Magic software,.

**THEORY:**

The digital to analog converter is a device widely used to convert digital pulses to analog signals. The two methods of creating DAC are binary weighted and R-2R ladder. DAC 0808 uses the R-2R method since it can achieve a high degree of precision. The first criterion for judging a DAC is its resolution, which is the functionof the number of binary inputs. The common ones are 8, 10 and 12 bits. The number of data bit inputs decides the resolution of the DAC since the number of analog output levels is equal to 2n, where n is the number of data inputs. DAC 0808 provides 256 discrete voltage or current levels of output. In DAC 0808, the digital inputs are converted into current I out and by connecting a resistor to I out pin, we convert the result to voltage. The total current provided by I OUT pin is a function of binary numbers at the D0-D7 pins inputs to DAC 0808 and reference current (Iref) is as follows:

Iout = Iref (D7/2+D6/4+D5/8+D4/16+D3/32...+D0/256)

Where D0 is the LSB, D7 is the MSB for the inputs and Iref is the input current that must be applied.

**Generating a sine wave**

To generate a sine wave, we first need a table whose values represent the magnitude of the sine of angles between 0 and 360 degrees. The values for the sine function vary from -1.0 to +1.0 for 0- to 360-degree angles. Therefore, the table values are integer numbers representing the voltage magnitude for the sine of theta. This method ensures that only integer numbers are output to the DAC by the 8051 microcontroller. Table 13-7 shows the angles, the sine values, the voltage magnitudes, and the integer values representing the voltage magnitude for each angle (with 30-degree increments). To generate Table 13-7, we assumed the full-scale voltage of 10 V for DAC output (as designed in Figure 13-18). Full-scale output of the DAC is achieved when all the data inputs of the DAC are high. Therefore, to achieve the full-scale 10 V output, we use the following equation.



Vout of DAC for various angles is calculated and shown in Table 13-7. See Example 13-5 for verification of the calculations.

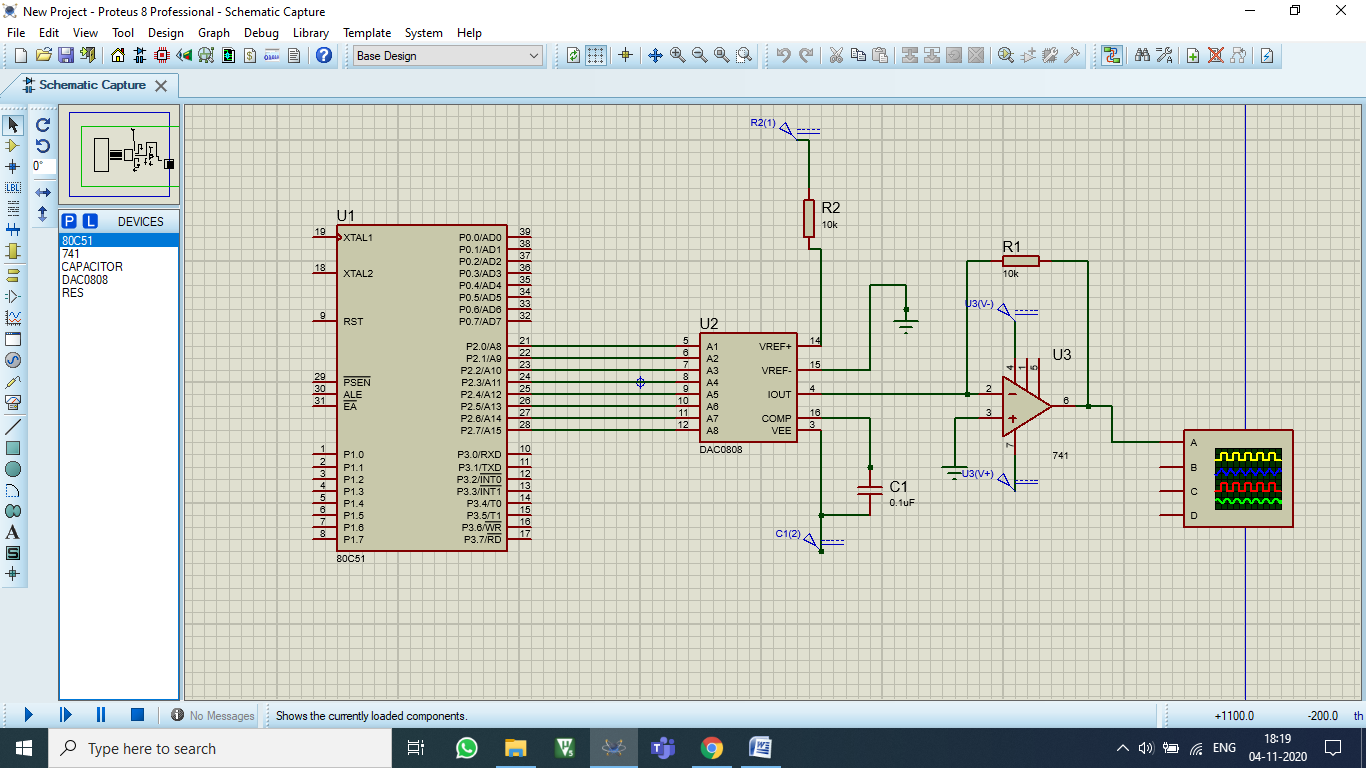
**Experimental Procedure:**

1. Click on Keil µVision4 icon for getting started.
2. Click on Project tab>Make new project> Select target device.
3. Click on File>New file.
4. Prepare a test code in assembly language as shown in editor window. Save it with .asm extension.
5. Add this created file to project. One may add one or more than one file in a single project.
6. Click Target1 (at left side pane)>Source Group> Right click to add code file.
7. Open Project tab> Options for target target1> Output tab>check ‘create hex file’ option.
8. Open Project tab> Build target. This will generate compiled .hex file from the

.asm or .C file, in the project created.

1. Now burn hex code onto microcontroller with Flash Magic software. Then observe output on peripherals.

**PROTEUS INTERFACING:**



**PROGRAM :**

1. **DAC SQUARE WAVE**
2. **DAC STAIRE CASE**

***Experiment No : 4***

**Title:**INTERFACING SWITCH RELAY

LED BUZZER

Roll No: \_ \_ \_ Batch: \_ \_ \_

Date of Performance:\_ \_ /\_ \_/\_ \_ \_ \_

Date of Assessment:\_ \_ /\_ \_/\_ \_ \_ \_

**TITLE** :Interfacing of input ( Switch, Relay) and output(LED, Buzzer) devices

to the ports.

**AIM:** Write a program for interfacing button, LED, relay & buzzer to PIC18FXX as follows:

1. When button 1 is pressed, relay and buzzer is turned ON and LED‟s start chasing from left to right
2. When button 2 is pressed, relay and buzzer is turned OFF and LED start chasing from right to left

**HARDWARE** :STK PIC 18F4520 kit, PC, PIC kit 2 programmer.

**SOFTWARE**: MPLAB IDE

**CONNECTIONS:**

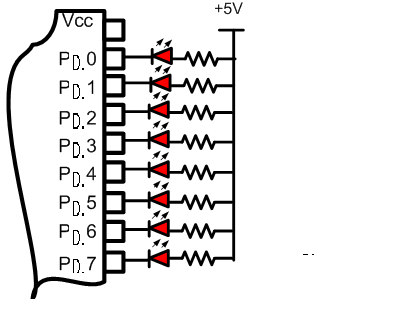
• Keep S6.1 switch in ON position.

• Keep S13.1 switch in ON position.

• Keep S12 switch in ON position.

• 8 LEDs (D9 to D16) present on STK-PIC18F4520 V1.1 are connected to PORT D.

All the LEDs are connected by common anode method. That means the positive leg ofeach LED is connected to VCC and negative leg to the port pins of the microcontroller. Logic 0 on the port pin will make LED ON and logic 1 will make it OFF.

****

**Fig.1: Interfacing with LED**

**PROCEDURE:**

• Compile the written code if errors are present then debug the code, modify it, save it and recompile the code using the build all option. After build succeeded and Hex file generated.

• Download and run this program.

**INTERFACING DIAGRAM:**

**PROGRAM :**

***Experiment No :5***

***Title:*** Interfacing of LCD to PIC 18FXXXX

Roll No: \_ \_ \_ Batch: \_ \_ \_

Date of Performance:\_ \_ /\_ \_/\_ \_ \_ \_

Date of Assessment:\_ \_ /\_ \_/\_ \_ \_ \_

**TITLE :**Interfacing of LCD (16x2) with PIC18F458.

**AIM:** Write an assembly language program to interface a LCD to 8051. Write down a program to display “HELLO” on first line and “ENTC” on second line of LCD.

**HARDWARE** :STK PIC kit, PC, PIC kit 2 programmer.

**SOFTWARE**: MIDE IDE, Flash Magic to Download.

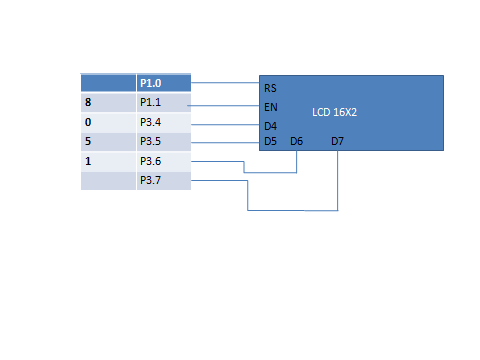
**LCD 4-Bit Operation:**

 LCD in 4-Bit means we are using 4 Lines of data bus instead of using 8 Line data bus. In this Method, we are Splitting Bytes of data in Nibbles. If you successfully interface Microcontroller with LCD with 4 Pins. Then we can save 4 Lines of Microcontroller, which can be used for other purpose. we are using 16 x 2 LCD.

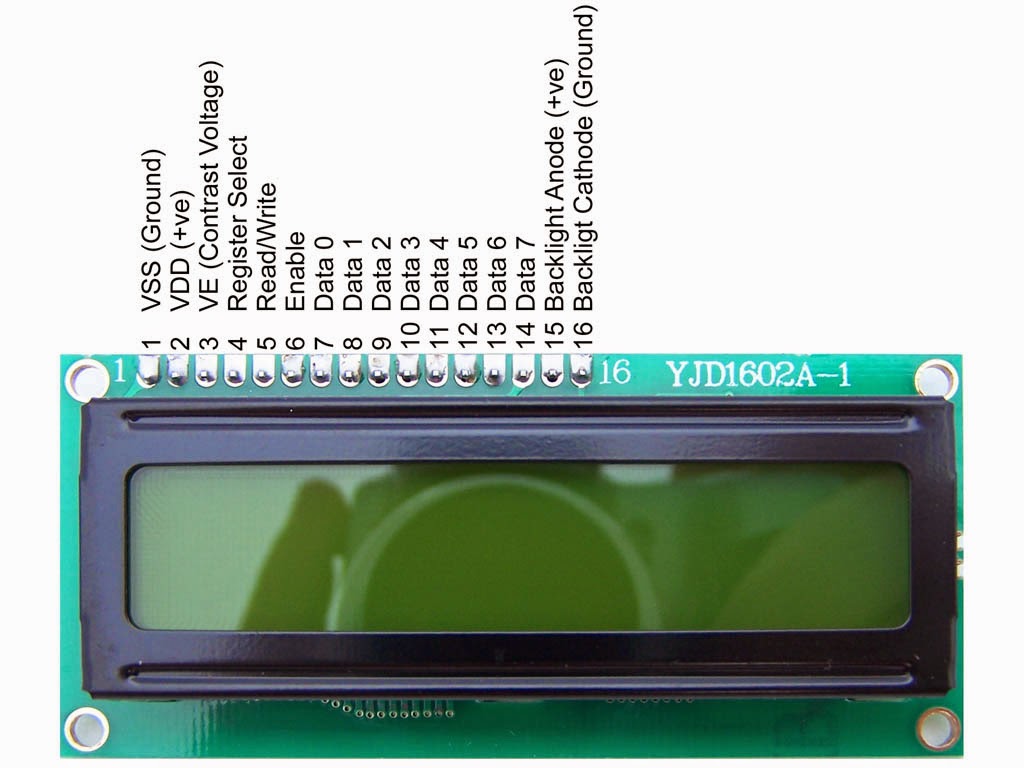
**CONNECTIONS:**

**8051 PORT 3 AND PORT 1 PINS ARE USED TO INTERFACE LCD PINS.**

|  |  |
| --- | --- |
| **8051 PORT PINS** | **LCD PINS** |
| **PORT P1.0** | **RS** |
| **PORT P1.1** | **EN** |
| **PORT P3.4** | **D4** |
| **PORT P3.5** | **D5** |
| **PORT P3.6** | **D6** |
| **PORT P3.7** | **D7** |

****

**16×2 LCD:**

****

LCD displays are available typically as 16x2, 20x1, 20x2 etc, along with LCD controller.16x2 means that 16 characters in each of the 2 lines can be stored.

This type of LCD module is very common and used widely in many types of display applications. It consists of 16 rows and 2 columns of 5×7 dot marices. The LCD display was a 16 pin package with back light, Contrast adjustment and 5×7 dot resolution. It consists of two built in registers known as data and Command register each has a specific function to perform with the display. The Data register is for writing the data to be displayed and Command register is to place the commands. The pin description of 16×2 LCD was given below.

The datas and the Commands both are given to the LCD through the data pins D0 to D7 but the logic state in the RS pin decides the data or command was given to the LCD.As stated in the above table for sending commands the RS pins should be in Logic 0 and for the datas the RS Pin should hold the Logic 1 or high state.

Alphanumeric Liquid Crystal Display (LCD’s) allow a better user interface with text messages to enter the instructions & get the response in the form of the text & know in a better manner what the machine is doing, including its diagnostic information. This also helps in fault findings and debugging. The main advantage of LCD displays is low power consumption and high speed with which the displayed information is updated.

**ADVANTAGES OF LCD OVER LEDS**

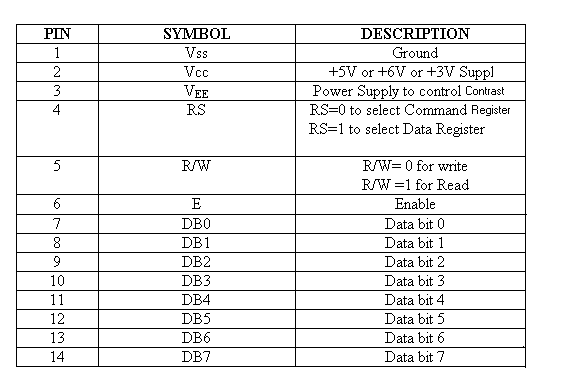
1.The decline prices of LCDs

2. The ability to display numbers, characters and graphics. This is in contrast to LEDs, which are limited to numbers and few characters

3. In corporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, the LEDs must be refreshed by the CPU to keep displaying the data.

4. Ease of programming for characters and graphics.

**Pin Description for LCD**

****

NOTE:-Pin number’s 15 and 16 are used for backlight control for saving power it should be grounded.

**RS (Register Select):**

When RS = 0 Command Code Register get selected for user to send command such as clear display, curser at home. When RS =1 Data Register get selected, allowing the user to Send data to be displayed on the LCD.

**R/W (Read/ Write):**

When R/W = 1 , microcontroller is reading information from LCD and when R/W = 0 microcontroller write information to LCD.

**E (Enable):**

This pin is used to latch the information presented to its data pins. When data is supplied to data pins, a high to low pulse must apply to this pin. The pulse must be Minimum of 450 ns wide.

**D0-D7 (Data Pins):**

The 8- bit data pins are used to send the information to the LCD or read the content of LCD’s internal register. To display letters and numbers, we send ASCII codes for the letters A-Z, a-z and numbers 0-9, while making RS =1.

**Busy Flag (D7):-**

The busy flag is D7 and can we read when RW=1 and RS=0, as follows; if RW=1, RS=0 when D7=1(busy flag=1), the LCD is busy taking care of internal operations and will not accept any new information. When D7=0, the LCD is ready to receive new information.

**LCD COMMANDS:**

|  |  |
| --- | --- |
| **CODE IN HEX** | **COMMAND TO INSTRUCTION REGISTER** |
| **01** | **Clear Display Screen** |
| **02** | **Cursor Return Home** |
| **04** | **Decrement cursor(shift cursor to left)** |
| **05** | **Shift display right** |
| **06** | **Increment cursor (shift cursor to right)** |
| **07** | **Shift display left** |
| **08** | **Display OFF, Cursor OFf** |
| **0A** | **Display OFF, Cursor ON** |
| **0C** | **Display ON, Cursor OFF** |
| **0E** | **Display ON, Cursor ON** |
| **0F** | **Display ON, Cursor Blinking** |
| **10** | **Shift cursor position to left by one character** |
| **14** | **Shift cursor position to right by one character** |
| **18** | **Shift the entire display to left** |
| **1C** | **Shift the entire display to right** |
| **80** | **Force cursor to the beginning of first line** |
| **C0** | **Force cursor to the beginning of second line** |
| **38** | **8-bit, 2 lines and 5x7 matrix dots** |
| **30** | **8-bit, 1 lines and 5x7 matrix dots** |
| **20** | **4-bit, 1 lines and 5x7 matrix dots** |
| **28** | **4-bit, 2 lines and 5x7 matrix dots** |
| **90** | **Force cursor to beginning of 3rd line** |
| **D0** | **Force cursor to beginning of 4th line** |
| **81** | **Cursor line 1 position** |
| **83** | **Cursor line 1 position 3** |
| **3C** | **Activate second line** |
| **C1** | **Jump to second line, position1** |
| **C2** | **Jump to second line, position2** |

**Command Routine: (As per the above interfacing)**

1. Microcontroller will send (or write) command on port 3 (same command is available on D4-D7 lines of LCD).
2. Microcontroller makes P1.0 low, which means RS=0; after that LCD will select command register.
3. It is write operation so microcontroller makes pin low, meaning R/W=0.
4. The command which is available on port P3 (or D4-D7) is latched into command register by making high to low transition on P1.1 (Enable pin).
5. After receiving the command in command register, LCD will perform operation as per the given command.
6. LCD will take some time to perform operation so after some time delay microcontroller will send next command ( Repeat step 1 to 6 for new command)

**Data Routine: (As per the above interfacing)**

1. Microcontroller will send (or write) data on port 3 (same data is available on D4-D7 lines of LCD).
2. Microcontroller makes P1.0 high, which means RS=1; after that LCD will select data register.
3. It is write operation so microcontroller makes pin low, meaning R/W=0.
4. The data which is available on port P3 (or D4-D7) is latched into data register by making high to low transition on P1.1 (Enable pin).
5. After receiving the data in data register, LCD will display the data.
6. LCD will take some time to display data so after some time delay microcontroller will send next data ( Repeat step 1 to 6 for new data)

**PROGRAM :**

***Experiment No :6***

***Title:*** Generate square wave using timer with interrupt

Roll No: \_ \_ \_ Batch: \_ \_ \_

Date of Performance:\_ \_ /\_ \_/\_ \_ \_ \_

Date of Assessment:\_ \_ /\_ \_/\_ \_ \_ \_

**AIM:** Generate square wave using timer with interrupt.

**SOFTWARE USED:**MPLABX IDE, XC8 Compiler

**THEORY:**

The Timer0 module incorporates the following features:

* Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
* Readable and writable registers
* Dedicated 8-bit, software programmable prescaler
* Selectable clock source (internal or external)
* Edge select for external clock
* Interrupt on overflow

T0CON: Timer0 Control Register

bit 7: TMR0ON: Timer0 On/Off Control bit 1 = Enables Timer0

0 = Stops Timer0

bit 6: T08BIT: Timer0 8-Bit/16-Bit Control bit

1 = Timer0 is configured as an 8-bit timer/counter

0 = Timer0 is configured as a 16-bit timer/counter

bit 5: T0CS: Timer0 Clock Source Select bit 1 = Transition on T0CKI pin

0 = Internal instruction cycle clock (CLKO)

bit 4: T0SE: Timer0 Source Edge Select bit

1. = Increment on high-to-low transition on T0CKI pin
2. = Increment on low-to-high transition on T0CKI pin

bit 3: PSA: Timer0 Prescaler Assignment bit 1 = TImer0 prescaler is NOT assigned. 0 = Timer0 prescaler is assigned.

bit 2-0: T0PS2:T0PS0: Timer0 Prescaler Select bits 111 = 1:256 Prescale value

110 = 1:128 Prescalevalue

101 = 1:64 Prescalevalue

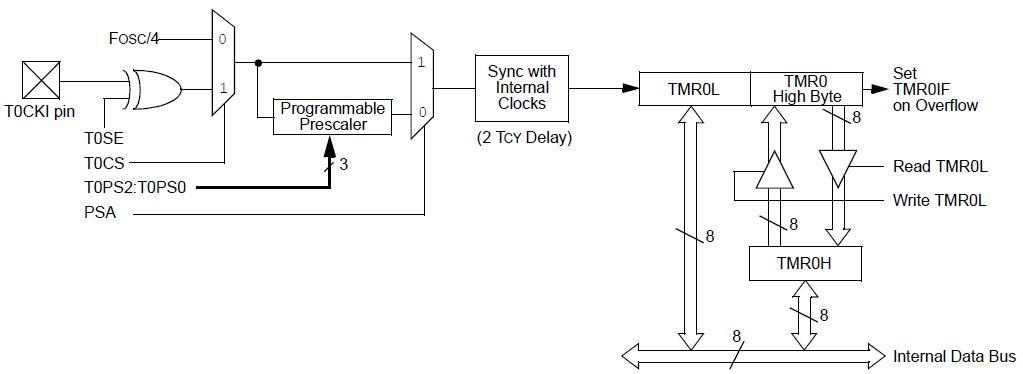
100 = 1:32 Prescalevalue

011 = 1:16 Prescalevalue

010 = 1:8 Prescalevalue

001 = 1:4 Prescalevalue

000 = 1:2 Prescalevalue



**Fig. Timer0 block diagram**

The PIC18F4550 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low-priority level.

There are ten registers which are used to control interrupt operation. These registers are:

* RCON
* INTCON, INTCON2, INTCON3
* PIR1, PIR2
* PIE1, PIE2
* IPR1, IPR2

Each interrupt source has three bits to control its operation. The functions of these bits are:

* Flag bit to indicate that an interrupt event occurred
* Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
* Priority bit to select high priority or low priority

When an interrupt is responded to, the global interrupt enable (GIE) bit is cleared to disable further interrupts. High-priority interrupt sources can interrupt a low priority interrupt. Low-priority interrupts are not processed while high-priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address. Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The “return from interrupt” instruction, RETFIE, exits the interrupt routine and sets the GIE bit which re-enables interrupts.

INTCON: Interrupt Control Register

bit 7: GIE/GIEH: Global Interrupt Enable bit

1 = Enable global/high priority interrupt.

0 = Disable global/high priority interrupt.

bit 6: PEIE/GIEL: Peripheral Interrupt Enable bit 1 = Enables all peripheral/low interrupts.

0 = Disables all peripheral/low interrupts.

bit 5: TMR0IE: TMR0 Overflow Interrupt Enable bit 1 = Enables the TMR0 overflow interrupt.

0 = Disables the TMR0 overflow interrupt. bit 4: INT0IE: INT0 External Interrupt Enable bit

1 = Enables the INT0 external interrupt.

0 = Disables the INT0 external interrupt. bit 3: RBIE: RB Port Change Interrupt Enable bit

1 = Enables the RB port change interrupt.

0 = Disables the RB port change interrupt. bit 2: TMR0IF: TMR0 Overflow Interrupt Flag bit

1 = TMR0 register has overflowed. (must be cleared in software)

0 = TMR0 register did not overflow.

bit 1: INT0IF: INT0 External Interrupt Flag bit

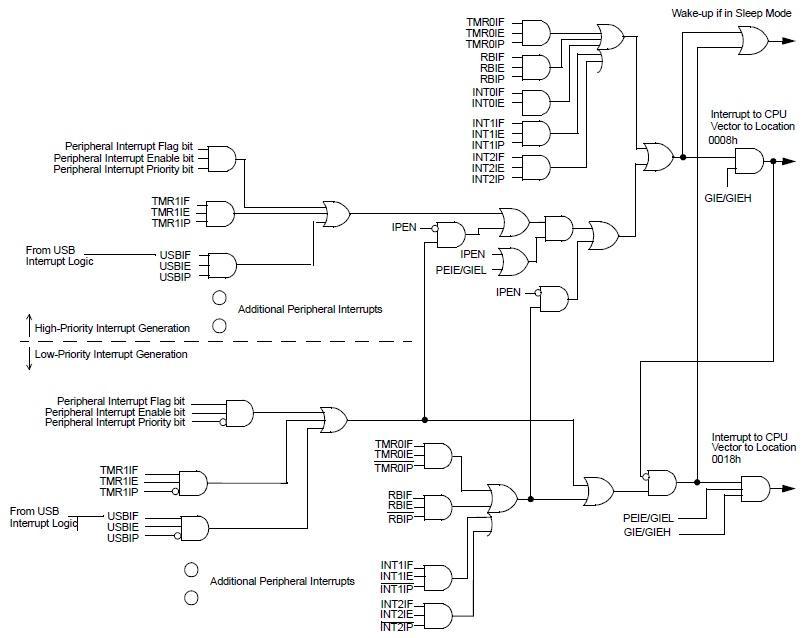
= The INT0 external interrupt occurred. (must be cleared in software)

0 = The INT0 external interrupt did not occur.

bit 0: RBIF: RB Port Change Interrupt Flag bit

1 = At least one of the RB7:RB4 pins changed state.

0 = None of the RB7:RB4 pins have changed state.



**Fig. Interrupt Logic**

To generate a square wave using timer interrupt, follow these steps:

1. Configure port pin as output.
2. Configure Timer0 in 16 bit mode and 1:256 prescale value.
3. Set Timer0 Interrupt Enable bit and clear the Timer0 Interrupt Flag.
4. Enable Global Interrupt register (GIE).
5. Load the Timer0 16 bit register in TMR0L and TMR0H.
6. Enable Timer0.

Interrupt Service Routine: When the Timer0 register overflows from 0xFFFF to 0x0000, the Timer0 flag (TMR0IF) is set and the Interrupt Service Routine is invoked.

1. Check if the Timer0 Interrupt flag is set.
2. If the flag is set, clear the flag and stop the timer.
3. Toggle the port pin.
4. Reload the Timer0 values and start the timer.
5. Return to the main function.

**PROGRAM:**

***Experiment No :7***

***Title:*** Interfacing serial port with PC both side communication.

Roll No: \_ \_ \_ Batch: \_ \_ \_

Date of Performance:\_ \_ /\_ \_/\_ \_ \_ \_

Date of Assessment:\_ \_ /\_ \_/\_ \_ \_ \_

**TITLE :**SERIAL COMMUNICATION

**AIM:** Interfacing serial port with PC both side communication.

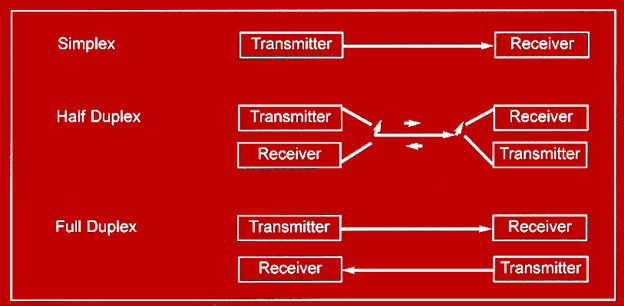
**HARDWARE** :STK PIC kit, PC, PIC kit 2 programmer.

**SOFTWARE**: MIDE IDE, FLASH MAGIC TO DOWNLOAD

**BASIC OF SERIAL DATA COMMUNICATION:**

**Communication Links**

Serial communication is classified into three types of communication links, Simplex, Half Duplex and Full duplex.



**Simplex:** In simplex transmission, the line is dedicated for transmission. The Transmitter sends and receives the data.

**Half Duplex:** In half duplex, the communication link can be used for either transmission or reception. Data is transmitted is only one direction at a time.

**Full Duplex:** If the data is transmitted in both ways at the same time, it is full Duplex, i.e. transmission and reception can proceed simultaneously. This communication link requires two wires data, one for data transmission and one for reception.

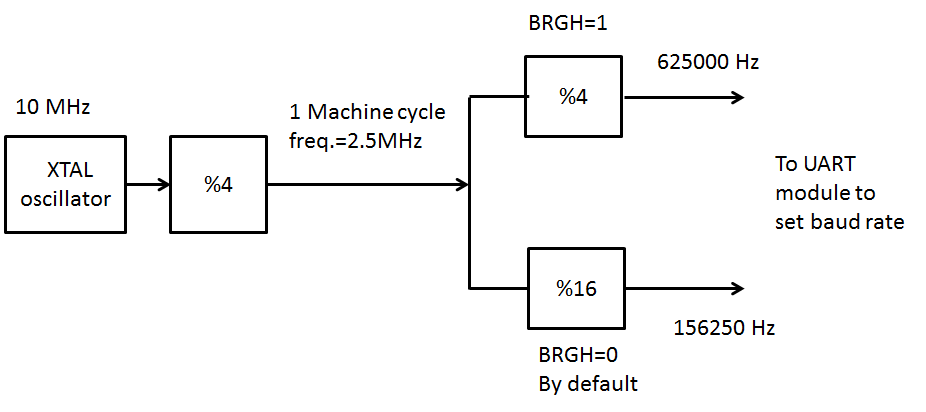
**Types of Serial Data communication**: Serial data communication uses two types of communication i.e. synchronous serial data communication and asynchronous serial data communication.

**Synchronous Serial Data Communication**: In synchronous serial data communication, transmitter and receiver are synchronized. It uses a common clock signal to synchronize the receiver and the transmitter. During the transmission of data; first the sync character and then the data is transmitted. This format is generally used for high speed transmission.

**Asynchronous Serial Data Communication**: In asynchronous serial data communication different clock sources are used for transmitter and receiver. In this mode, data is transmitted with start and stop bits. Transmission begins with start bit, followed by data and then stop bit. Some time one more bit i.e. parity bit is included just prior to stop bit.

**Comparison between synchronous and asynchronous data transfer**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Asynchronous data transfer** | **Synchronous data transfer** |
| 1 | It is used to transfer one character at a time. ( character oriented) | It is used to transfer a block of characters at a time. (block oriented) |
| 2 | Used for data transfer rates<= 20k BPS ( Slow data rate) | Used for high data rate (>=20kBPS) |
| 3 | Start bit and stop bit for each character is present which forms a frame. | No start and stop bit is used |
| 4. | Sync characters are not transmitted along with characters. | Sync. Characters are transmitted along with the group of characters. |
| 5. | Two separate clock inputs can be used for transmitter & receiver. | One clock inputs is used for transmitter & receiver. |
| 6 | Speed is slow ( more overhead) | speed is high (less overhead) |

****

**Fig. PIC 18F UART BUADRATE GENERATION**

PIC18 divides the crystal frequency Fosc by 4 to get the instruction cycle time frequency.

The PIC18s UART circuitry divides the instruction cycle frequency by 16 once more before it is used by an internal timer to set the baud rate.

The operation of the Enhanced USART module is performed using following registers:

1. Serial port Baud Rate Generator(SPBRG)
2. Transfer Buffer Register(TXREG) and Receive Buffer Register (RCREG)
3. Transmit Status and Control (TXSTA)
4. Receive Status and Control (RCSTA)
5. Peripheral Interrupt Register(PIR1)
6. Baud Rate Control (BAUDCON)

**PROGRAM CODE:**

***Experiment No :8***

***Title:***Generation of PWM signal for DC Motor control

Roll No: \_ \_ \_ Batch: \_ \_ \_

Date of Performance:\_ \_ /\_ \_/\_ \_ \_ \_

Date of Assessment:\_ \_ /\_ \_/\_ \_ \_ \_

**TITLE :**Interfacing of DC MOTOR using PWM

**AIM**: Write an embedded ‘C’ program for generation of PWM signal for DC Motor

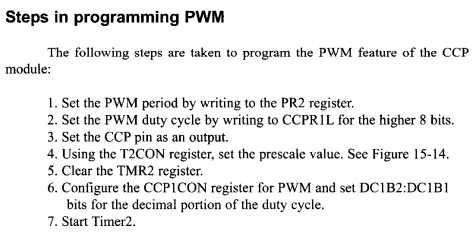
Control.

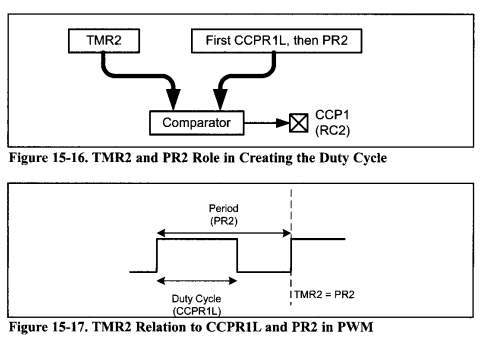
**HARDWARE** :STK PIC kit, PC, PIC kit 2 programmer.

**SOFTWARE**: MPLAB IDE

**THEORY:**

* The PWM feature allows us to create pulses with variable widths.
* Although we can program timers to create PWM, the CCP module makes the programming of PWM much easier and less tedious.
* PWM is widely used in industrial controls such as DC motor control.
* In creating Pulses with variable widths for the PWM, two factors are important, The period of the pulse and its duty cycle.

****

****

**PROCEDURE:**

* Compile the written code if errors are present then debug the code, modify it, save it and recompile the code using the build all option. After build succeeded Hex file is generated.
* Download and run this program.

**PROGRAM :**