Sinhgad Technical Education Society's

RMD SINHGAD SCHOOL OF EGINEERING, WARJE 58

Sinhgad Institutes

DEPARTMENT: ELECTRONICS AND TELECOMMUNICATION

INDEX

SUBJECT: MICROCONTROLLERS

STES

RMD,SINHGAD SCHOOL OF EGINEERING, WARJE 58

EXPERIMENT NO: 01

Name of the Student:-_____

Roll No._____ Subject:-_____

Date of Practical Performed:-_____ Staff Signature with Date & Marks

TITLE : To study implementation & interfacing of LED.

AIM : Write an Embedded C program language program for LED flashing.
[ON/OFF, counter, Ring counter, alternate flashing] etc.

PROBLEM STATEMENTS:

Design and develop an LED interfacing board with 8051 and
1. Write an Embedded C program for flashing led on LED bank .

2. Write an Embedded C program to display ring counter on LED interface which will rotate first in left and then right direction also make provision of passing the same data in opposite direction to different ports.

OBJECTIVE:

1. To understand the concept of interfacing of LEDs with port of 8051 microcontroller.
2. Write programs using simple routine for flashing of LEDs
3. Write an Embedded C program to display ring counter on LED interface .

S/W PACKAGES AND H/W USED:

Keil micro vision, Explore 8051Development Board

THEORY:

Light Emitting Diodes are the semi conductor light sources. Commonly used LEDs will have a cut-off voltage of 1.7V and current of 10mA. When an LED is applied with its required voltage and current it glows with full intensity. The Light Emitting Diode is similar to the normal PN diode but it emits energy in the form of light. The colour of light depends on the band gap of the semiconductor.

Thus, LED is directly connected to the AT89C51 microcontroller. The negative terminal of the LED is connected to the ground through a resistor. Value of this resistor is calculated using the following formula.

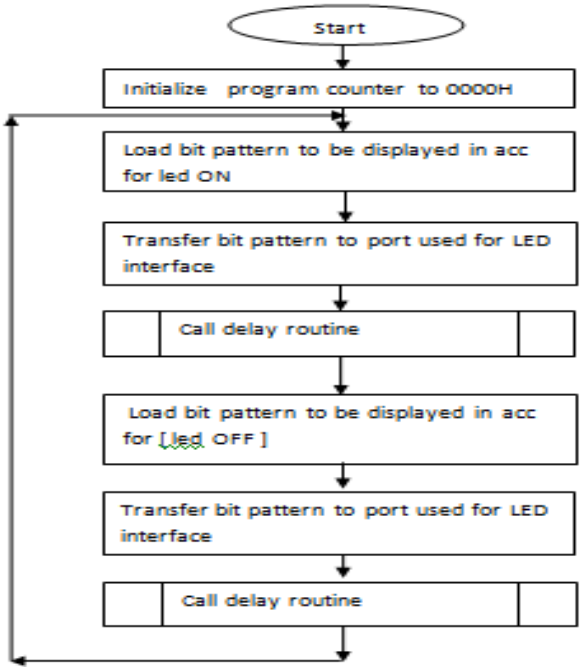
R= (V-1.7)/10mA, where V is the input voltage.

Generally, microcontrollers output a maximum voltage of 5V. Thus the value of resistor calculated for this is 330 Ohms. Thus this can be connected either to the cathode or anode of the LED.

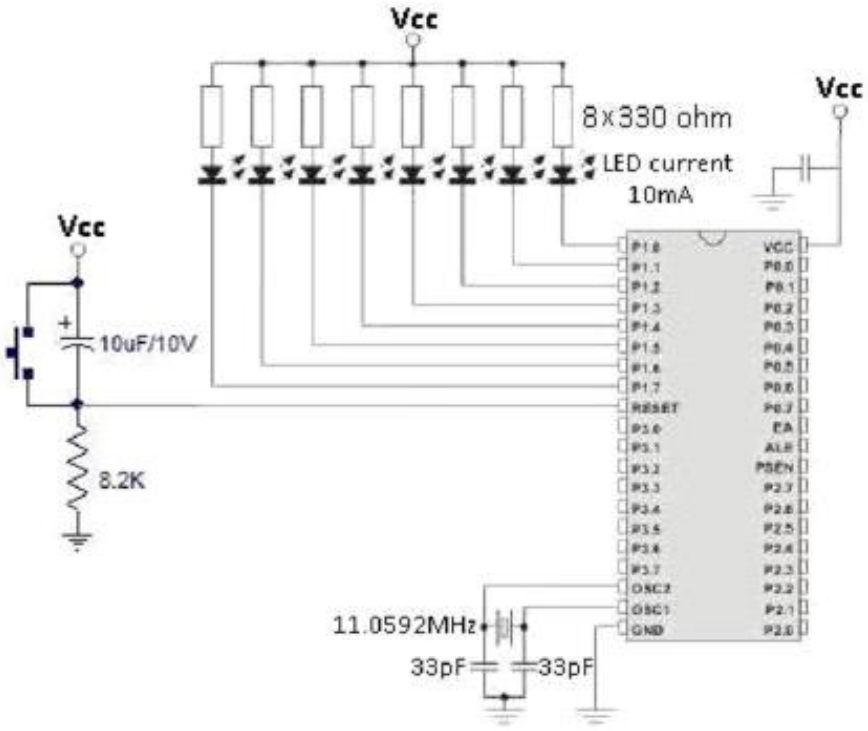
ALGORITHM

Step 1 : Initialize program counter to 0000h
Step 2 : Load bit pattern to glow LED into the accumulator
Step 3 : Move contents of accumulator to Port register
Step 4 : Wait for some time ie delay [may be generated using timers or register]
Step 5 : Load Same or different Data sequence of LED glowing into the accumulator
Step 6 : Move contents of accumulator to port register
Step 7 : Wait for some time i.e. Delay
Step 8 : Continue Go to step 1

FLOWCHART:



INTERFACING DIAGRAM:



Interfacing of led with 8051

CONCLUSION:

Write answers for following questions:

1. Explain structure of Port 1.
2. Explain the different pins of port 3.
3. Comment on interfacing of common Anode and common Cathode LED.
4. Specify minimum requirement of LED interfacing with 8051.
5. Explain functions TMOD and TCON register
6. Explain importance of RST pin
7. Explain why clock frequency of 8051 is set to 11.0592MHz
8. Explain significance of Interrupt register in programming with format
9. What is vectored interrupts give tabulation with priority
10. Enlist general steps in calculation of delay using timers

Viva Questions:

1. Which register bank is selected after reset?
2. What is significance of bit addressable instructions?
3. Why it is necessary to set stack pointer in subroutine programs.
4. Explain different flags of 8051 how they will be set and reset.
5. Differentiate between interfacing of CA and CC type of LED.

STES

RMD,SINHGAD SCHOOL OF EGINEERING, WARJE 58

EXPERIMENT NO: 02

Name of the Student:-_____

Roll No._____ Subject:-_____

Date of Practical Performed:-_____ Staff Signature with Date & Marks

TITLE: INTERFACING OF DAC WITH 8051

AIM: To generate different types of waveforms using DAC 0808

Problem Statements:

Design and develop DAC interface card and

1. Write an ALP for generation of Square wave .
2. Write ALP for generation of triangular wave one after another with 60% duty cycle
3. Write an ALP to generate staircase waveform.

OBJECTIVE:

1. To understand the concept waveform generation
2. Understand the interfacing diagram with microcontroller
3. Able to write programs foe generation of waveforms

THEORY: The different types of waveforms can be generated using DAC 0808 when interfaced with 8051 Microcontroller. The DAC0808 is an 8-bit monolithic digital-to-analog converter (DAC) featuring a full scale output current settling time of 150 ns while dissipating only 33 mW with ±5V supplies No reference current (IREF) trimming is required for most

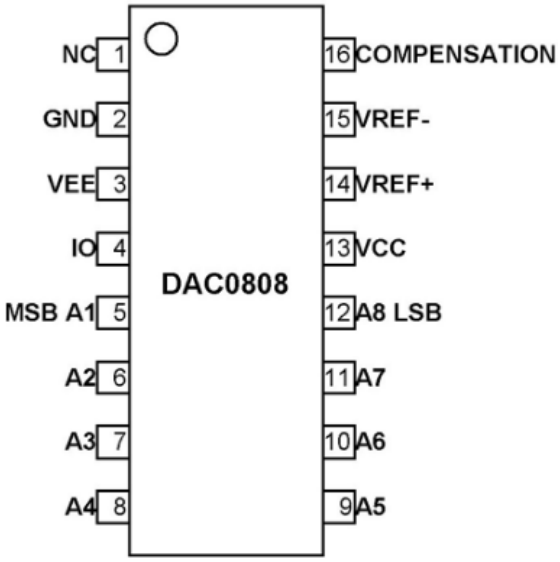
applications since the full scale output current is typically ±1 LSB of 255 IREF/256. Relative accuracies of better than ±0.19% assure 8-bit monotonicity and linearity while zero level output current of less than 4 μA provides 8-bit zero accuracy for IREF^{3}2 mA. The power supply current of the DAC0808 is independent of bit codes, and exhibits essentially constant device characteristics over the entire supply voltage range. The DAC0808 will interface directly with popular TTL, DTL or CMOS logic levels,

The DAC 0808 has following features

1. Relative accuracy: ±0.19% error maximum
2. Full scale current match: ±1 LSB typ
3. Fast settling time: 150 ns typ
4. Non inverting digital inputs are TTL and CMOS compatible
5. High speed multiplying input slew rate: 8 mA/μs
6. Power supply voltage range: ±4.5V to ±18V

7. Low power consumption: 33 mW @ ±5V

The pin out diagram for DAC 0808 is shown in fig 1.



The pins are labeled A1 through A8, but note that A1 is the Most Significant Bit, and A8 is the Least Significant Bit (the opposite of the normal convention). Ground the two least significant bits. The D/A convertor has an output current, instead of an output voltage. The output pin should stay at about 0 volts. The op-amp on the "Typical Application" on the datasheet converts the current to a voltage. How does it do this? The output current from pin 4 ranges between 0 (when the inputs are all 0) to Imax *255/256 when all the inputs are 1. The current, Imax , is determined by the current into pin 14 (which is at 0 volts). Note: Since we are using 8 bits, the maximum value is IMAX *255/256. The output of the D/A convertor takes some time to settle. You may need to take this in consideration when planning the timing of the A/D conversion.

Fig(1). Pin Configuration DAC0808

The D/A converter have an output current, instead of an output voltage. The output pin should stay at about 0 volts. It uses I to V converter at the output pin of DAC

$$I_o = K\left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{64} + \frac{A7}{128} + \frac{A8}{256}\right)$$

$$where, K \cong \frac{V_{REF}}{R14}$$

$$Vo = R_f . I_o$$

$$so, Vo = Rf . K\left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{64} + \frac{A7}{128} + \frac{A8}{256}\right)$$

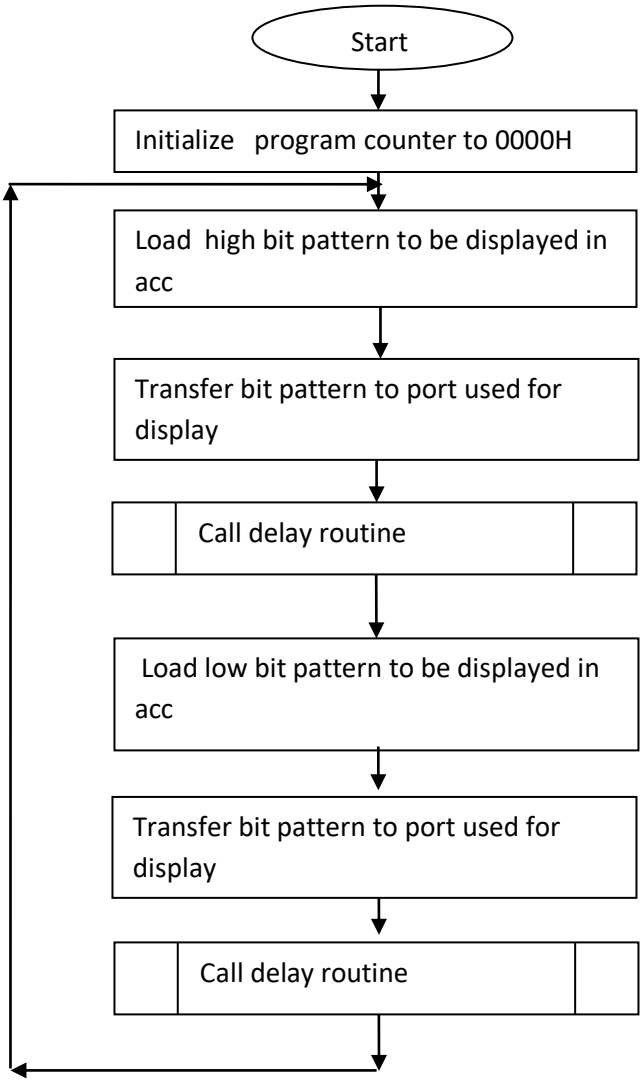
- **ALGORITHM :** Depending on type of waveform generated algorithm changes

FOR square Wave: it can be generates on single port pin or at all the port pins

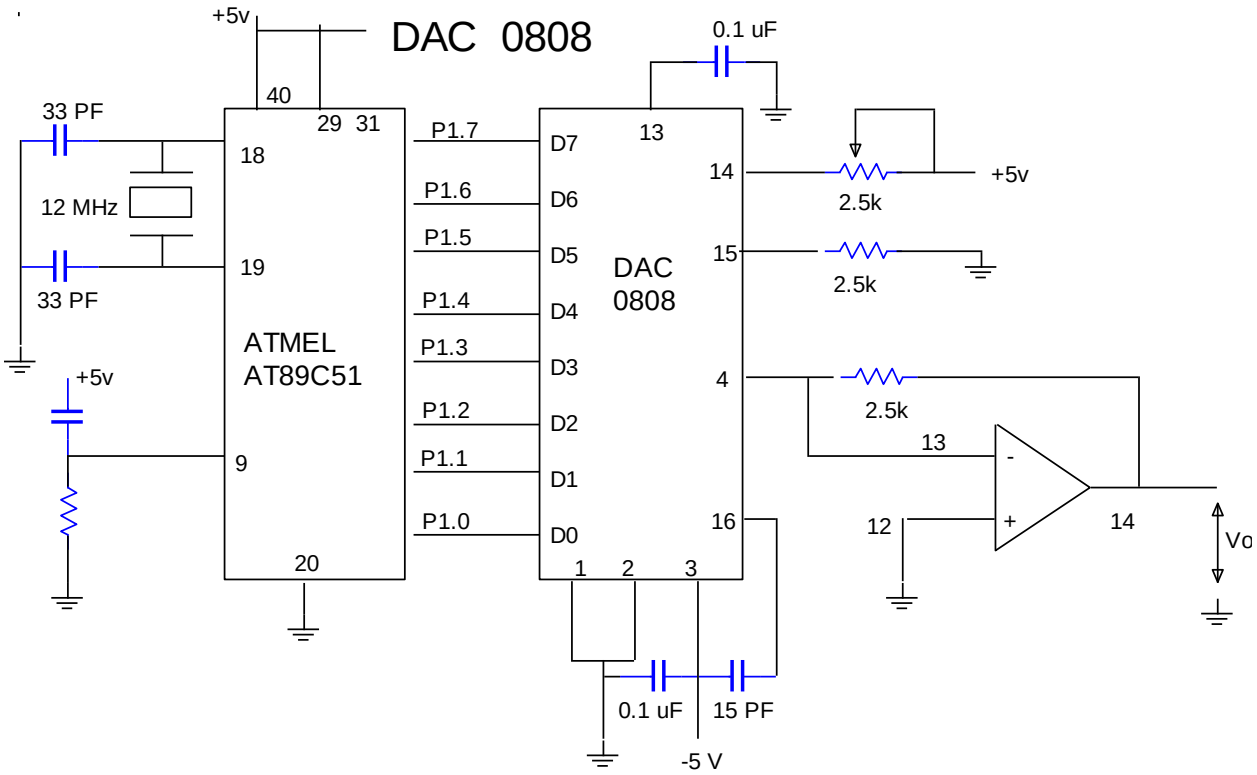
1. Load high[0ff] data into accumulator
2. Transfer it to port
3. Delay :Wait for some time depending on duty cycle
4. Output Low(00) to port
5. Delay :
6. Repeat Steps 1-5

** Wave forms can be generated with change in reference point. Either by software or by Hardware

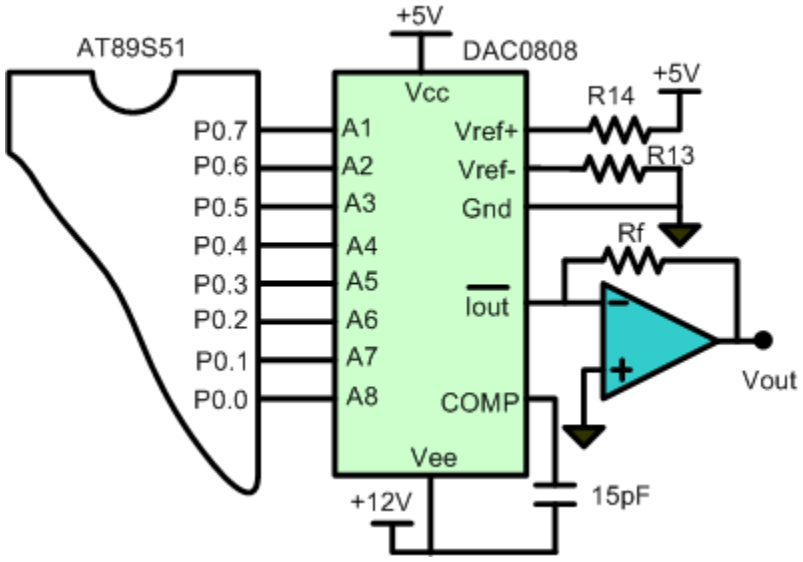
- **Flow chart**



Interfacing diagram: The complete interface of DAC with 8051 is shown in fig (2)



OR



Fig(2): DAC interface with 8051

Conclusion:

1. Write answers for following questions:

2. Draw an interfacing diagram to interface DAC .

3. write an embedded c code for generation of following waveforms

   a.Square wave b.Triangular wave

   c.Staircase ramp d. Ramp wave

4. Why DAC conversion is necessary.

STES

RMD,SINHGAD SCHOOL OF EGINEERING, WARJE 58

EXPERIMENT NO: 03

Name of the Student:-_____

Roll No._____ Subject:-_____

Date of Practical Performed:-_____ Staff Signature with Date

& Marks

Title: STEPPER MOTOR INTERFACING

AIM : Write an 8051 code for Stepper motor interfacing, to step Forward and Reverse direction (clockwise and anticlockwise)

OBJECTIVE:

1. To understand the concept of stepper motor interface
2. Able to write programs for rotating stepper motor at different angles

PROBLEM STATEMENT:

Using concept of stepper motor and interfacing with 8051 , to step forward and Reverse direction.

S/W & H/W USED : 1. Keil Simulator

2. Flash magic Software for downloading

3. 8051 Trainer kit.

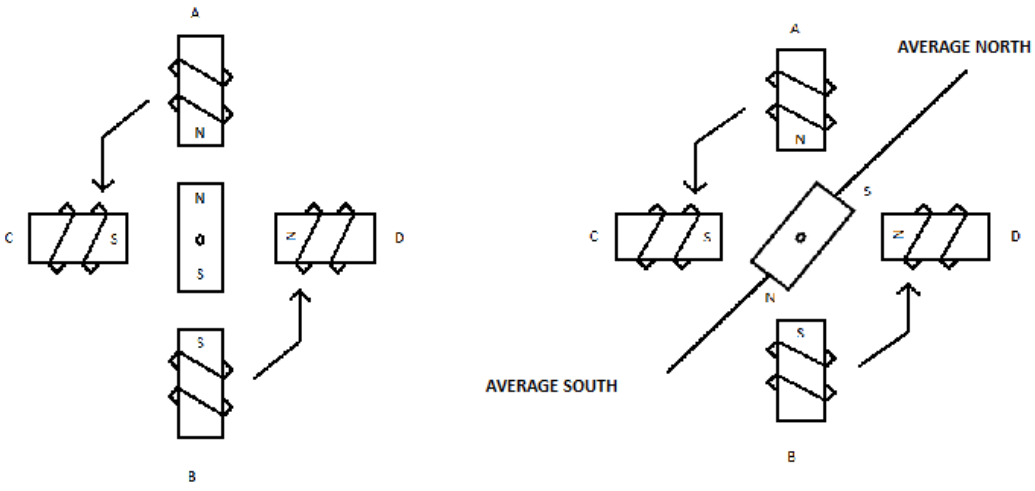
Theory

This section begins with an overview of the basic operation of stepper motors. Then we describe how to interface a stepper motor to the 8051. Finally we use assembly language programs to demonstrate control of the angle and direction of stepper motor rotation.

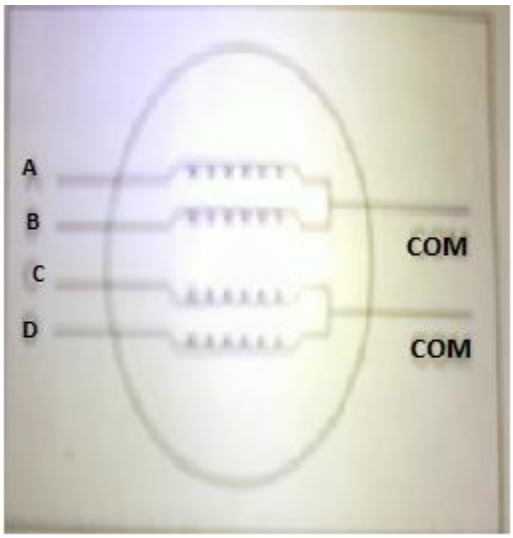
Stepper motors

A stepper motor is a widely used device that translates electrical pulses into mechanical movement. In applications such as disk drives, dot matrix printers, and robotics the stepper motor is used for position control. Stepper motors commonly have permanent magnet rotor surrounded by a stator. There are also steppers called variable reluctance stepper motors that do not have a PM rotor. The most common stepper motors have four stator windings that are paired with centre tapped

common as shown. This type of stepper motor is most commonly referred to as a four phases or unipolar stepper motor. The centre tap allows the change of current direction in each of the two coils when a winding is grounded, thereby resulting in a polarity change of the stator. Notice that while a conventional motor shaft runs freely, the stepper motor shaft moves in a fixed repeatable increment, which allows one to move to a precise position. This repeatable fixed movement is possible as a result of the basic magnetic theory where the poles of the same polarity repel and opposite poles attract. The direction of rotation is dictated by the stator poles. The stator poles are determined by the current sent through the wired coils. As the direction of the current is changed, the polarity is also changed causing the reverse rotation of the motor. The stepper motor discussed here has total of 6 leads : 4 leads representing four stator windings and 2 common for centre tapped leads. As the sequence of the power is applied to each stator winding the rotor will rotate. There are several widely used sequences where each a different degree of precision.



It must be noted that although we can start with any any of the sequence , once we start we must continue in proper order.



STATOR WINDING CONFIGURATION

NORMAL 4-STEP SEQUENCE

CLOCKWISE ⇩ ⇧ COUNTER CLOCKWISE

Step#	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

Step angle

How much movement is associated with a single step? This depends on the internal construction of the motor, in particular of the number of the teeth on the stator and the rotor. The step angle is minimum degree of rotation associated with a single step. In table notice the term steps per revolution. This is the total number of the steps needed to rotate one complete rotation or 360 degrees.

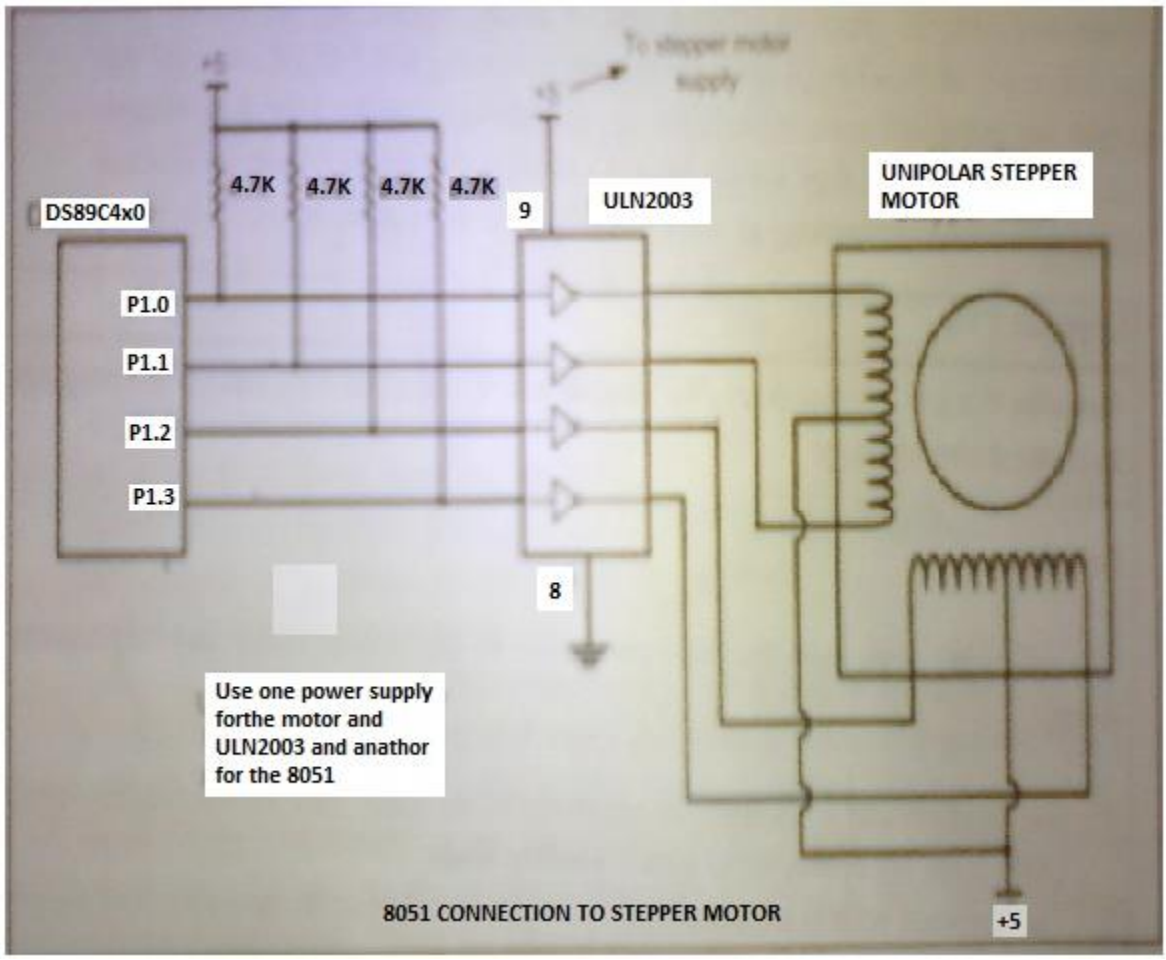
STEP ANGLE	STEPS PER REVOLUTION
0.72	500
1.8	200
2	180
2.5	144
5	72
7.5	48
15	24

It must be noted that perhaps contrary to one's initial impression, a stepper motor does not need more terminal leads for the stator to achieve smaller steps. All the stepper motors discussed in section have 4 leads for the stator winding and 2 COM wires for the common signal instead of two , they always have 4 leads for the stators. Next we discuss some associated terminology in order to understand the stepper motor further.

Steps per second and RPM relation

The relation between rpm, steps per revolution and steps per second is as follows:

Steps per second = (rpm x steps per revolution)/(60)



The four step sequence and number of teeth on the rotor

The switching sequence shown earlier in the table is called the 4 step switching sequence since after the same two windings will "ON". How much movement is associated with these four steps? After completing every 4 steps the rotor moves only when one tooth pitch. Therefore, in a stepper motor in 200 steps per revolution, the rotor has 50 teeth since 4 x 50 = 200 steps are needed to complete one revolution. This leads to the conclusion that the minimum step angle is always a function of the number of teeth on the rotor. In other words, the smaller the step angle, the more teeth the rotor passes.

One might wonder what happens if we want to move 45 degrees, since the steps are 2 degrees each. To allow for finer resolutions, all stepper motors allow what is called an 8-step switching sequence. The 8 step sequence is also called half stepping, since in the 8 step sequence each step is half of normal step angle. For example, a motor with a 2 degree step angle can be used as a 1 degree step angle if the sequence of table is applied.

HALF STEP 8 STEP SEQUENCE

	WINDING	WINDING	WINDING	WINDING
STEP#	A	B	C	D
1	1	0	0	1
2	1	0	0	0
3	1	1	0	0
4	0	1	0	0
5	0	1	1	0
6	0	0	1	0
7	0	0	1	1
8	0	0	0	1

ALGORITHM:

FLOWCHART:

Conclusion:

Questions:

1. Write a short note on interfacing of Stepper motor.
2. Write an embedded c code for rotate the motor clockwise and anticlockwise.
3. Explain TMOD and TCON register of Timer.
4. Explain SBUFF and SCON register of 8051.

STES

RMD,SINHGAD SCHOOL OF EGINEERING, WARJE 58

EXPERIMENT NO: 04

Name of the Student:-_____

Roll No._____ Subject:-_____

Date of Practical Performed:-_____ Staff Signature with Date & Marks

TITLE : Interfacing Push buttons, LEDs, Relay & Buzzer to PIC Microcontroller

PROBLEM STATEMENT:

Interface Push buttons, LEDs, Relay and Buzzer to PIC Microcontroller. Write a program in Embedded C to interact with peripherals as follows.

a. LED's start chasing from left to right and turn ON Relay, buzzer whenever pushbutton 1 is pressed.
b. LED's start chasing from right to left and turn OFF Relay, buzzer whenever pushbutton 2 is pressed.

OBJECTIVE :

a. To understand the PORT Structure of PIC Microcontroller.
b. To study the SFRs to control the PORT Pins.
c. To interface common peripherals like pushbuttons, LEDs, relay.
d. To understand the use of MPLAB IDE and C18 Compiler.
e. To write a simple program in Embedded C.

S/W PACKAGES AND H/W USED:

MPLAB IDE, C18 Compiler, ExplorePIC Development Board

THEORY

Depending on the device selected, there are up to five general purpose I/O ports available on PIC18F Microcontroller devices. 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.

1. Some common Features of the I/O Ports

- Up to 70 bi-directional I/O pins
  - Some multiplexed with peripheral functions
- High drive capability
  - 25mA source/sink capability
- Direct, single cycle bit manipulation
- 4kV ESD protection diodes
  - Based on human body model
- After reset:
  - Digital I/O default to Input (Hi-Z)
  - Analog capable pins default to analog

2. SFR Associated with I/O Port

Each port has three registers for its operation and figure 1.1 and figure 1.2 shows

the functioning of each registers:

TRIS register (Data Direction register): To select PORT pin as input or output. All port pins are input by default. Whenever a bit in the TRISx register is a 0, the corresponding bit in PORTx is an output. If the bit in TRISx is a 1, the corr esponding bit in PORTx is an input.

PORT register (reads the levels on the pins of the device)

LAT register (output latch): The data latch (LAT register) is u seful for read-modify-write operations on the value that the I/O pins are driving.

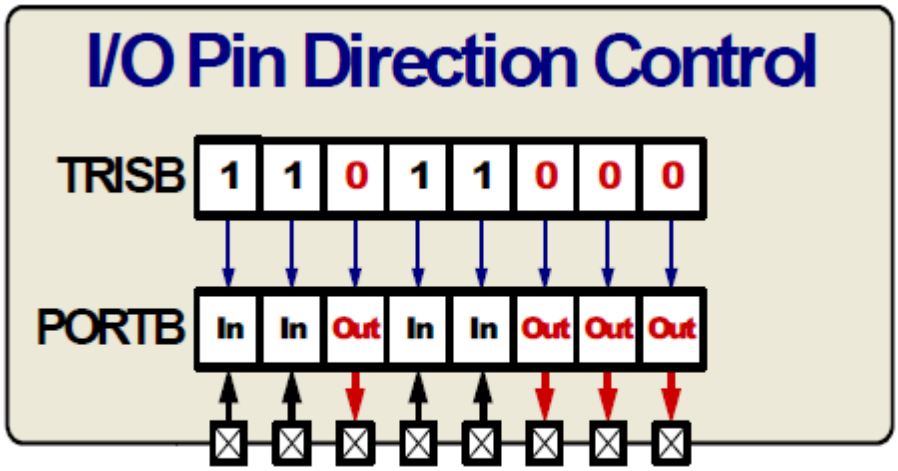


Figure 1.1 : TRISx Register

3. Generic I/O Port structure:

All the ports of PIC18 are bidirectional and identical. They all have the following four components in their structure as shown in figure 1.3.

1. DATA LATCH
2. OUTPUT DRIVER
3. INPUT BUFFER
4. TRIS LATCH

The PIC18 Ports have both the latch and buffer. Therefore, when reading the ports there are two possibilities:

1. Reading the input pin
2. Reading the latch

3.1 PORTA, TRISA and LATA Registers

• PORTA is an 8-bit wide, bidirectional port.

• The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin.

• The RA6 pin is multiplexed with the main oscillator pin; it is enabled as an oscillator or I/O pin by

3.2 PORTB, TRISB and LATB Registers

• PORTB is an 8-bit wide, bidirectional port.

• Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit RBPU (INTCON2 register). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

• Four of the PORTB pins (RB7:RB4) have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output

3.3 PORTC, TRISC and LATC Registers

• PORTC is an 8-bit wide, bidirectional port.

• PORTC is multiplexed with several peripheral functions. PORTC is primarily multiplexed with serial communication modules, including the EUSART, MSSP module and the USB module.

• PORTC pins have Schmitt Trigger input buffers.

3.4 PORTD, TRISD and LATD Registers

• PORTD is an 8-bit wide, bidirectional port.

• All pins on PORTD are implemented with Schmitt Trigger input buffers.

• Three of the PORTD pins are multiplexed with outputs P1B, P1C and P1D of the enhanced CCP module.

• PORTD can also be configured as an 8 Slave Port (PSP) or Streaming Parallel Port (SPP)). In this mode, the input buffers are TTL.

• On a Power-on Reset, these pins are configured as digital inputs.

3.5 PORTE, TRISE and LATE Registers

• For 40/44-pin devices, PORTE is a 4

• Three pins (RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7) are individually configurable as inputs or outputs.

• These pins have Schmitt Trigger input buffers. When selected as an analog input, these pins will read as '0's..

3. Interfacing Diagram

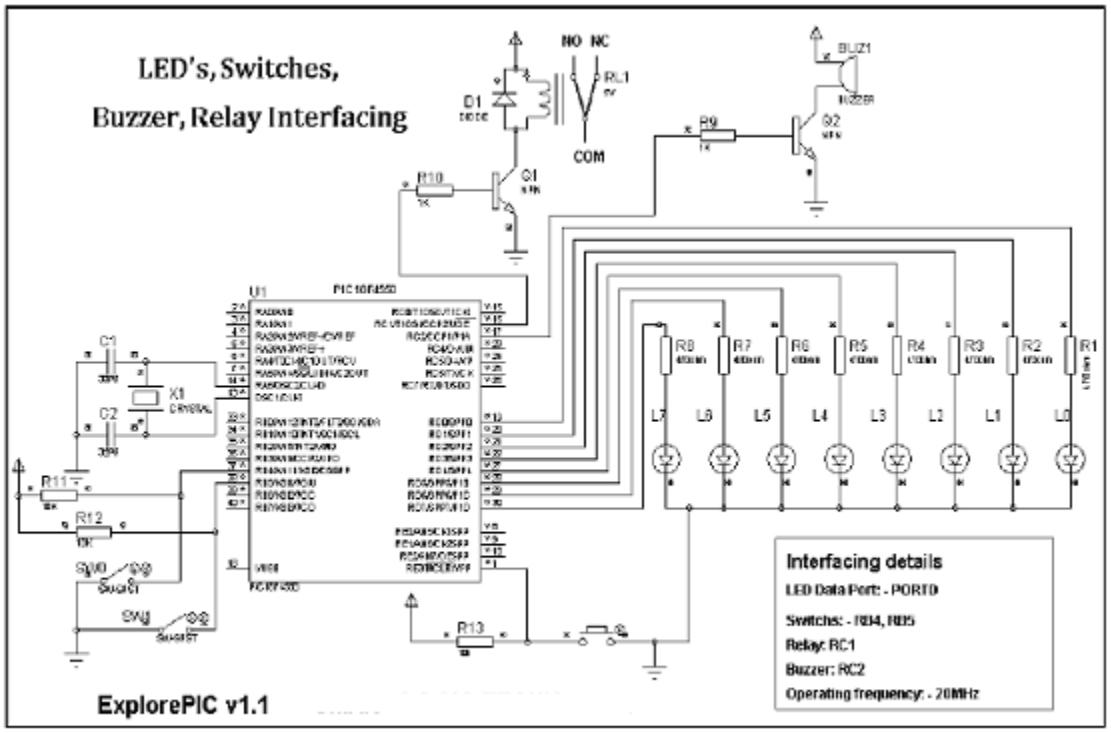


Figure : Interfacing LED's, Switches, Buzzer and Relay

4. Algorithm

1. As LED connected to PORTD configure these pins as output by writing 0x00 to the appropriated TRES register.

2. As buttons are connected to RB.4 and RB.5 so configure these pins as input by writing 1 to the appropriated bits of TRES register.

3. Check the status of RB.4, if it is zero then move 1 bit of PORTB from right to left.

4. Else check the status of RB.5, if it is zero then move 1 bit of PORTB from left to right.

5. Repeat step 3 to 4.

CONCLUSION:

Questions:

1. Explain the addressing modes of PIC18F.
2. Write an Embedded C program to blink LED connected to Port of PIC.
3. Draw an interfacing diagram to interface LED and write an embedded C program to blink alternate LED to port B using interrupt routine.
4. Explain with diagram interrupt structure of PIC 18f.

STES

RMD,SINHGAD SCHOOL OF EGINEERING, WARJE 58

EXPERIMENT NO: 03

Name of the Student:-_____

Roll No._____ Subject:-_____

Date of Practical Performed:-_____ Staff Signature with Date & Marks

EXPERIMENT NO. 5

TITLE : Interfacing LCD to Display Message

PROBLEM STATEMENT:

Interface 16x2 LCD to PIC Microcontroller in 8-bit Mode. Write a program in Embedded C language .

OBJECTIVE :

a. To understand the working of Liquid Crystal Display (LCD).
b. To study the LCD interfacing modes and Timing diagram.
c. To study and use of the LCD commands to drive LCD.

S/W PACKAGES AND H/W USED:

MPLAB Software , PIC Development Board

THEORY

1. LCD Interfacing

In recent years the LCD is finding widespread use replacing LED's. This is due to the following reason:

a. The declining prices of LCD.

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

c. In corporation of a refreshing controller into the LCD, thereby reliving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU to keep displaying the data.

d. Ease of programming for characters and graphics.

Most of LCD's available in the market are based on controller HD44780. The LCD display can be interfaced either in 4-bit interface or 8-bit interface mode.

1.1 LCD pin descriptions:

1.1.1 Vcc, Vss and Vee: While Vcc and Vss provide +5V and ground, repectively, Vee is used for controlling LCD contrast.

1. Register Select (RS):

   There are two very important registers inside the LCD. The RS pin is used for their selection as follows.

   a. RS = 0: the instruction command code register is selected, allowing the user to send a command such as clear display, cursor at home.

   b. RS = 1: the data register is selected, allowing the user to send the data to be displayed on the LCD.

1.1.3 Read/write (R/W):

R/W input allows the user to write information to the LCD or read information from it. R/W = 1 when reading, R/W = 0 when writing.

1.1.4 Enable (EN):

The enable pin is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high to low pulse must be applied to the pin in order for the LCD to latch in the data present at the data pins. This pulse must be a minimum of 450ns wide.

1.1.5 Data bus (D0 – D7):

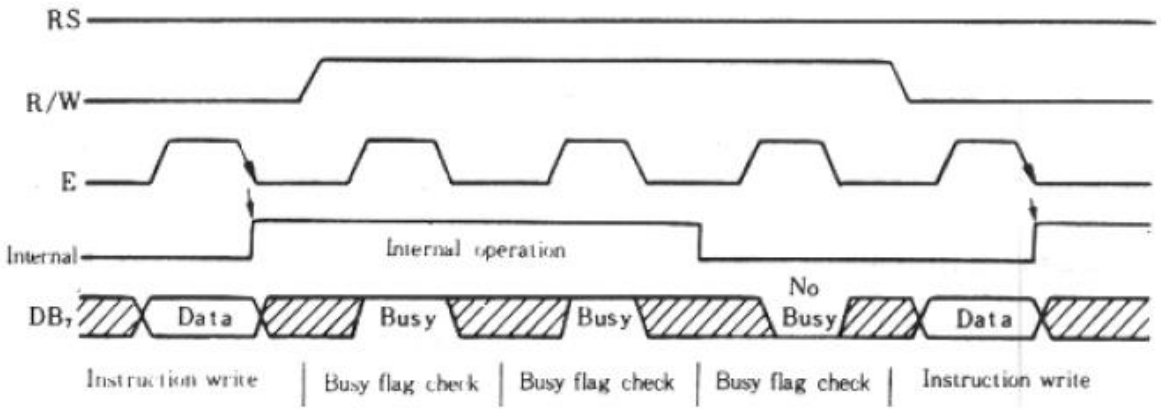
The 8-bit data pins, D0-D7 are used to send the information to the LCD or read the contents of the LCD's internal registers. To display the numbers and letters, we send ASCII codes to these pins while making RS=1. There are also instruction command codes that can be sent to the LCD to clear the display or blink the cursor. We also use RS = 0 to check the busy flag bit to see if the LCD is ready to receive

information. The busy flag is D& and can be read when R/W = 1 and RS=0. When D7 =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.

1.2 Pin Assignment of 16x2

Pin number	Symbol	Level	I/O	Function
1	Vss	-	-	Power supply (GND)
2	Vcc	-	-	Power supply (+5V)
3	Vee	-	-	Contrast adjust
4	RS	0/1	I	0 = Instruction input 1 = Data input
5	R/W	0/1	I	0 = Write to LCD module 1 = Read from LCD module
6	E	1, 1->0	I	Enable signal
7	DB0	0/1	I/O	Data bus line 0 (LSB)
8	DB1	0/1	I/O	Data bus line 1
9	DB2	0/1	I/O	Data bus line 2
10	DB3	0/1	I/O	Data bus line 3
11	DB4	0/1	I/O	Data bus line 4
12	DB5	0/1	I/O	Data bus line 5
13	DB6	0/1	I/O	Data bus line 6
14	DB7	0/1	I/O	Data bus line 7 (MSB)
15	VB+	1	-	Backlight Supply
16	VB-	0	-	

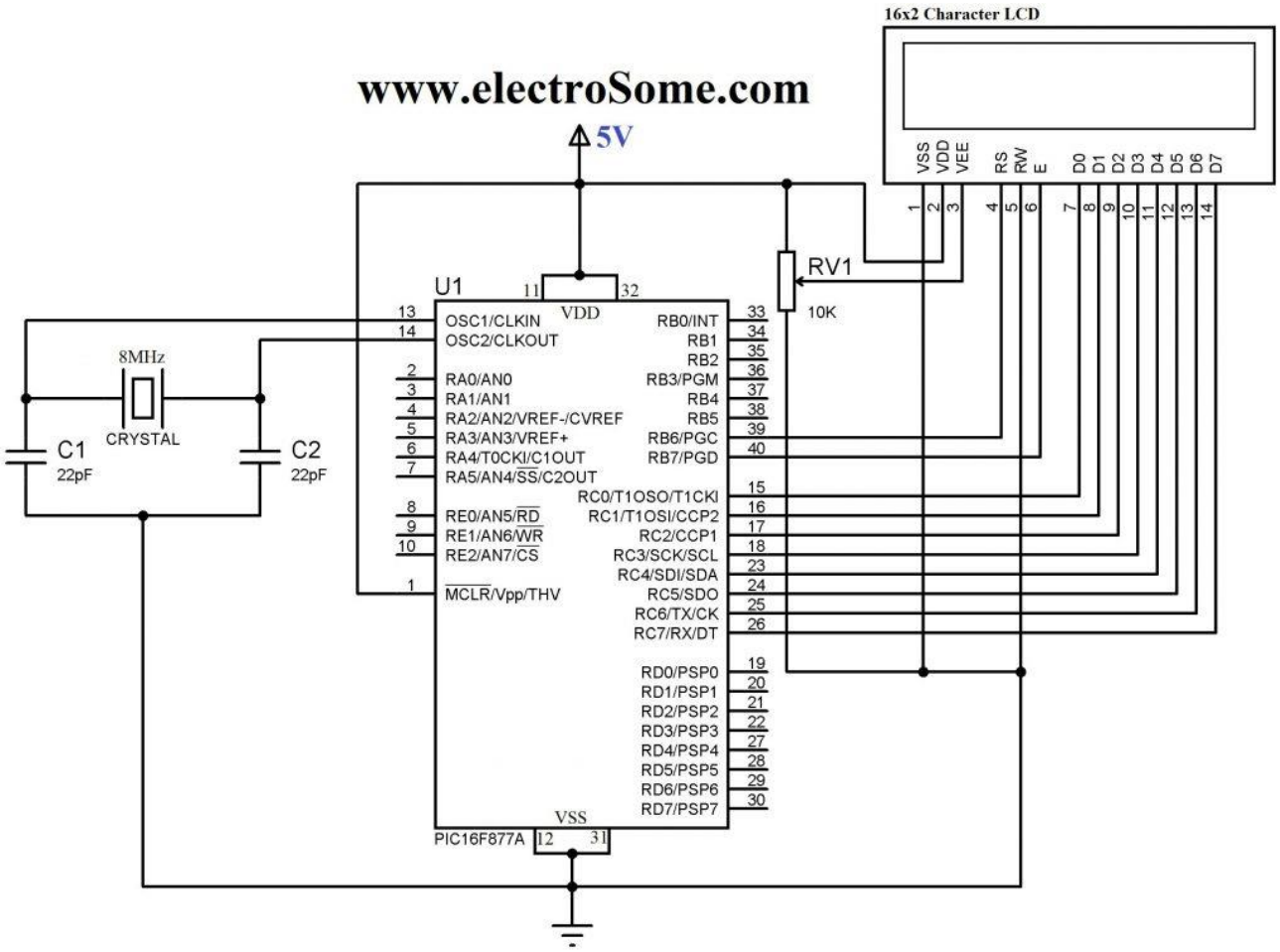
1.3 Timing Diagram for 16x2 LCD



1.4 LCD command codes

Sr. No.	Command to LCD instruction	Code (Hex)
01	Clear display screen	01
02	Return home	02
03	Decrement cursor (shift cursor to left)	04
04	Increment cursor (shift cursor to right)	06
05	Shift display right	05
06	Shift display left	07
07	Display off, cursor off	08
08	Display off, cursor on	0A
09	Display on, cursor off	0C
10	Display on cursor blinking	0E
11	Shift cursor position to left	10
12	Shift cursor position to right	14
13	Shift the entire display to left	18
14	Shift the entire display to right	1C
15	Force cursor to beginning of 1st line.	80
16	Force cursor to beginning of 2nd line.	C0
17	2 lines and 5x7 matrixes.	38

2. Interfacing Diagram



3. Algorithm

1. As LCD data bus is connected to PORTD and handshaking signal EN RS, R/W, configure PORT E.

2. Initialize LCD.

a. Write a commands {2 line and 5X7 matrix (0x38), Display ON cursor OFF (0x0C), Increment & shift cursor right (0x06), Clear display screen (0x01) to LCD}.

3**. Display message on first line.**

a. Write a command 0x80 to LCD.

b. Load ASCII character from the given string.

c. Write data to LCD

d. Repeat 2 and 3 sixteen times.

4. Display message on Second line.

a. Write a command 0xC0 to LCD.

b. Load ASCII character from the given string.

c. Write data to LCD

d. Repeat 2 and 3 sixteen times.

5. Endless Loop

Write a command or Write a data

1. Send 8-bit command or data on data port

2. Make RS pin low (for command operation), Make RS pin high (for data operation)

3. Make RW pin low

4. Latch data (Enable latch & Disable latch)

CONCLUSION:

Questions:

1. Explain LCD Control lines.
2. What is the function of RS, RW, Enable.
3. Address of 1st line and 2nd line of LCD.
4. Write an embedded C program of LCD with algorithm.
5. Why R/W pin is always connected to ground

STES

RMD,SINHGAD SCHOOL OF EGINEERING, WARJE 58

EXPERIMENT NO: 06

Name of the Student:-_____

Roll No._____ Subject:-_____

Date of Practical Performed:-_____ Staff Signature with Date & Marks

TITLE : Generation of Square wave using timer with interrupt.

PROBLEM STATEMENT:

Write a program to generate a square wave of 10 Hz on the port pin RB0 (pin no. 33). Use timer0 interrupt for ON period and OFF period delay.

OBJECTIVE :

a. To understand the basic concepts of Timer and Counter
b. To study in detail Timer0 of PIC Microcontroller
c. To Study interrupt structure of PIC Microcontroller
d. To use timer interrupt and its related SFR.

S/W PACKAGES AND H/W USED:

MPLAB IDE, C18 Compiler, Explore PIC Development Board

THEORY:

1. Timer

The microcontroller oscillator uses quartz crystal for its operation. Even though it is not the simplest solution, there are many reasons to use it. The frequency of such oscillator is precisely defined and very stable, so that pulses it generates are always of the same width, which makes them ideal for time measurement. Such oscillators are also used in quartz watches. If it is necessary to measure time between two events, it is sufficient to count up pulses generated by this oscillator. This is exactly what the timer does. Most programs use these miniature electronic 'stopwatches'. These are commonly 8- or 16-bit SFRs the contents of which are automatically incremented by each coming pulse. Once a register is completely loaded and overflowed, an interrupt may be generated! If the timer uses an internal quartz oscillator for its operation then it can be used to measure time between two events (if the register value is A at the moment measurement starts, and B at the moment it terminates, then the elapsed time is equal to the result of subtraction B - A). If registers use pulses coming from external source then such a timer is turned into a counter.

1.1 How does the timer operate?

In practice, pulses generated by the quartz oscillator are once per each machine cycle, directly or via a prescaler, brought to the circuit which increments the number stored in the timer register. If one instruction (one machine cycle) lasts for four quartz oscillator periods then this number will be incremented a million times per second (each microsecond) by embedding quartz with the frequency of 4MHz. It is easy to measure short time intervals, up to 256 microseconds, in the way described above because it is the largest number that one register can store. This restriction may be easily overcome in several ways such as by using a slower oscillator, registers with more bits, prescaler or interrupts. The first two solutions have some weaknesses so it is more recommended to use prescaler or interrupt.

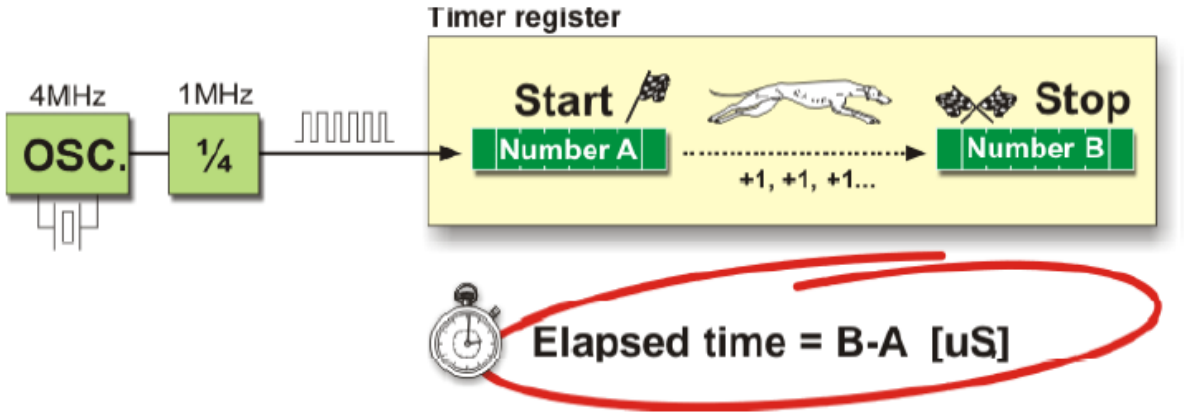


Figure 1.1 : Timer Operation

1.2 Using a prescaler in timer operation

A prescaler is an electronic device used to reduce frequency by a predetermined factor. In order to generate one pulse on its output, it is necessary to bring 1, 2 , 4 or more pulses on its input. Most microcontrollers have one or more prescaler built in and their division rate may be changed from within the program. The prescaler is used when it is necessary to measure longer periods of time. If one prescaler is shared by timer and watchdog timer, it cannot be used by both of them simultaneously.

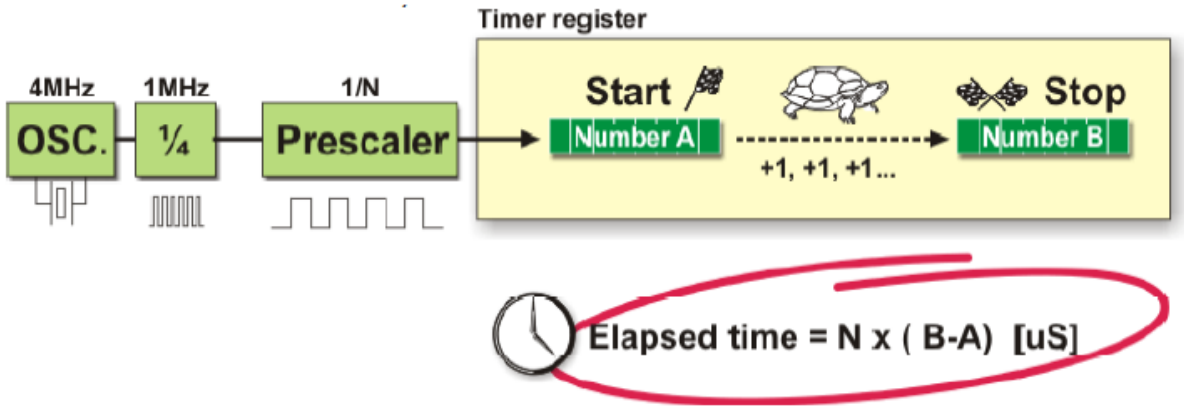


Figure 1.2: Use of prescaler

1.3 Using interrupt in timer operation

If the timer register consists of 8 bits, the largest number it can store is 255. As for 16-bit registers it is the number 65535. If this number is exceeded, the timer will be automatically reset and counting will start at zero again. This condition is called an overflow. If enabled from within the program, the overflow can cause an interrupt, which gives completely new possibilities. For example, the state of registers used for counting seconds, minutes or days can be changed in an interrupt routine. The whole process (except for interrupt routine) is automatically performed behind the scenes, which enables the main circuits of the microcontroller to operate normally. This figure 1.3 illustrates the use of an interrupt in timer operation. Delays of arbitrary duration, having almost no influence on the mainprogram execution, can be easily obtained by assigning the prescaler to the timer.

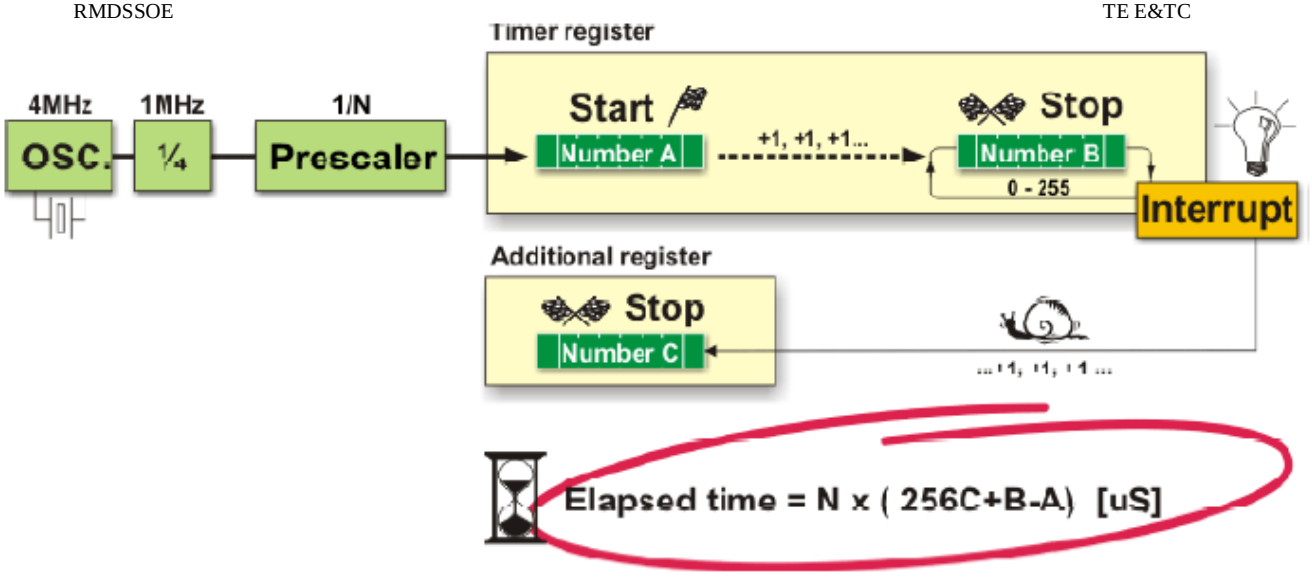


Figure 1.2: Timer interrupt operation

2. Counters

If the timer receives pulses from the microcontroller input pin, then it turns into a counter. Obviously, it is the same electronic circuit able to operate in two different modes. The only difference is that in this case pulses to be counted come over the microcontroller input pin and their duration (width) is mostly undefined. This is why they cannot be used for time measurement, but for other purposes such as counting products on an assembly line, number of axis rotation, passengers etc. (depending on sensor in use).

3. Timers / Counters in PIC Microcontroller

There are 3 - 5 timers on board in PIC microcontroller. These timers can also be used as counters when external pulses are applied. The timers are programmable, and sometimes share with other peripheral devices. These are named as TMR0, TMR1, TMR2, TMR3 and TMR4. There are few other timers, not to be discussed here like Watchdog Timer and Brown-Out timers. These timers are useful in measuring the time delays in various events as well as counting and timing external events.

Parameter	Timer 0	Timer 1 & 3	Timer 2 & 4
Size of timer register	8-bit or 16-bit	16-bit	8-bit
Clock Source (Internal)	$F_{osc}/4$	$F_{osc}/4$	$F_{osc}/4$
Clock Source (External)	T0CKI Pin	T13CKI Pin / T1OSC	None
Clock Scaling (Prescaler)	Prescaler 8-bits (1:2 -> 1:256)	Prescaler 2-bits (1:1 -> 1:8)	Prescaler (1:1, 1:4, 1:8) Postscaler (1:1 -> 1:16)
Interrupt Event	On Overflow	On Overflow	TMR Reg matches with PR2

Table 1.1: Comparison of Timers in PIC Microcontroller

4. Timer0 Module in PIC Microcontroller

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

The T0CON register (Register 11-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable. A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 1.1. Figure 1.2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

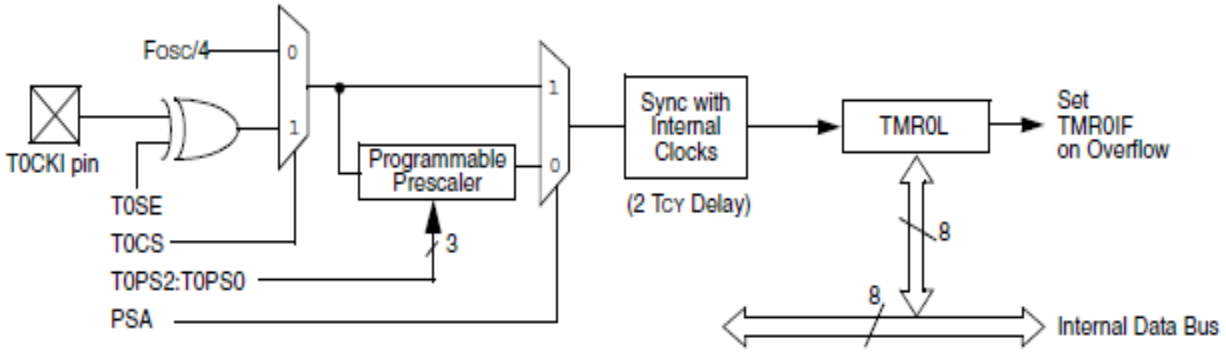


Figure 1.1: Block Diagram of Time0 in 8-bit

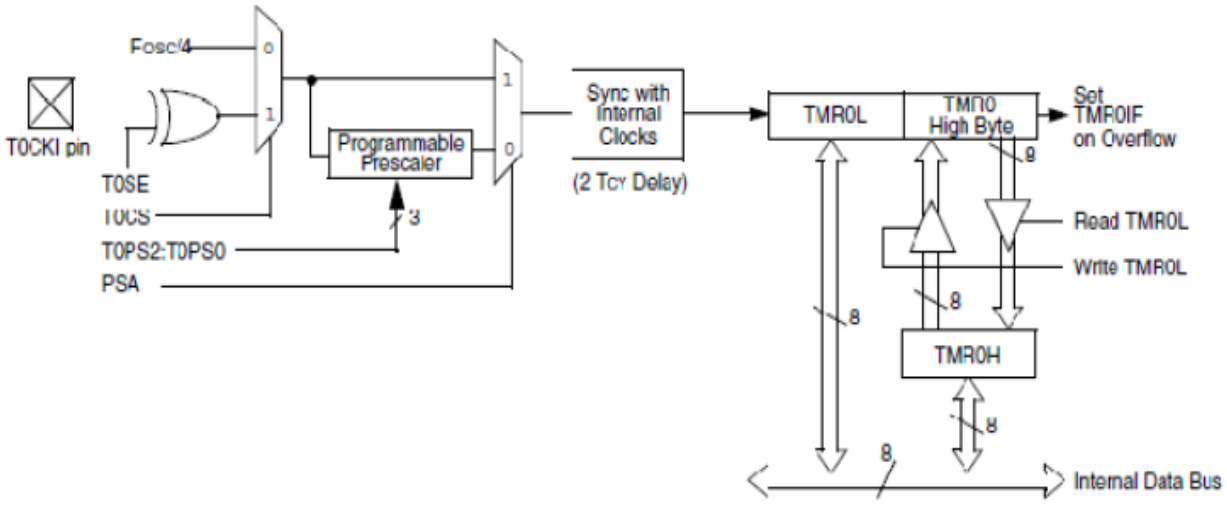


Figure 1.2: Block Diagram of Time0 in 16-bit

4.1 Timer0 Operation

Timer0 can operate as either a timer or a counter; the mode is selected by clearing the T0CS bit (T0CON<5>). In Timer mode, the module increments on every clock by default unless a different prescaler value is selected. If the TMR0 register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0
register.

4. Timer0 Module in PIC Microcontroller

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
The T0CON register (Register 11-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable. A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 1.1. Figure 1.2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

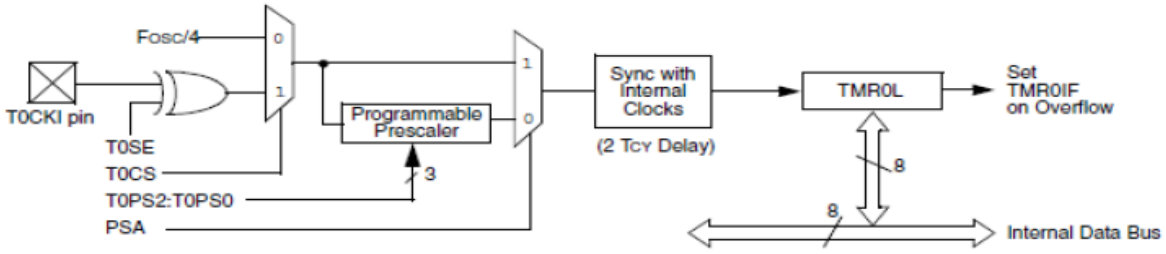


Figure 1.1: Block Diagram of Time0 in 8-bit

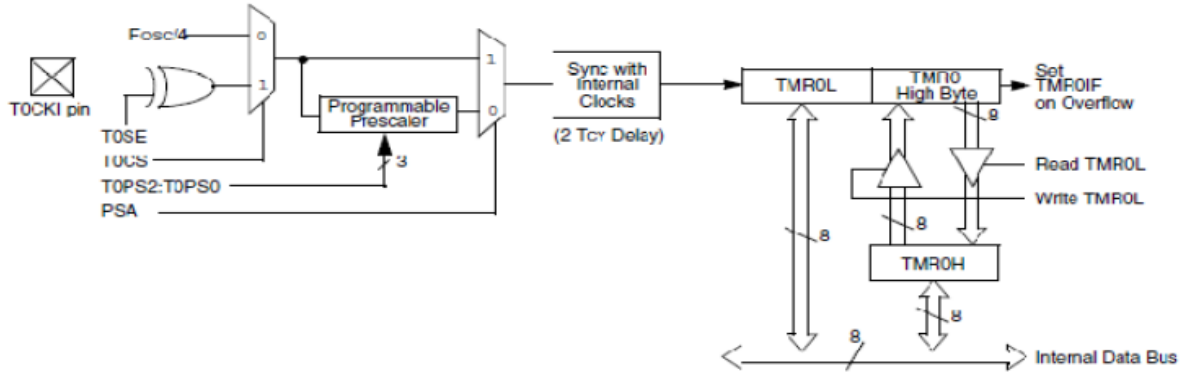


Figure 1.2: Block Diagram of Time0 in 16-bit

4.1 Timer0 Operation

Timer0 can operate as either a timer or a counter; the mode is selected by clearing the T0CS bit (T0CON<5>). In Timer mode, the module increments on every clock by default unless a different prescaler value is selected. If the TMR0 register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register. The Counter mode is selected by setting the T0CS bit (= 1). In Counter mode, Timer0

increments either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (TOSC). There is a delay between synchronization and the onset of incrementing the timer/counter.

4.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable; its value is set by the PSA and T0PS2:T0PS0 bits (T0CON<3:0>) which determine the prescaler assignment and prescale ratio. Clearing the PSA bit assigns the prescaler to the Timer0 module. When it is assigned, prescale values from 1:2 through 1:256, in power-of-2 increments are selectable. When assigned to the Timer0 module, all instructions writing to the TMR0 register clear the prescaler count.

4.3 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the TMR0IE bit (INTCON<5>). Before reenabling the interrupt, the TMR0IF bit must be cleared in software by the Interrupt Service Routine. Since Timer0 is shut down in Sleep mode, the TMR0 interrupt cannot awaken the processor from Sleep.

4.4 Timer0 Register Map:

SFR	Description	Access	Reset Value	Address
T0CON	Timer0 Control Register	Read/Write	0xFF	0xFD5
TMR0L	Timer0 Register Lower Byte	Read/Write	UNKNOWN	0xFD6
TMR0H	Timer0 Register Higher Byte	Read/Write	0x00	0xFD7
INTCON	Interrupt Control Register	Read/Write	0x00	0xFF2
INTCON2	Interrupt Control Register 2	Read/Write	0xFF	0xFF1

4.4. Register (SFR) Description

4.4.1 TMR0L : Used 8-bit and 16bit mode. The register holds the current count value which is updated by clock source. User must write initial value.
4.4.2 TMR0H : Used only in 16-bit mode. The register holds the higher byte current count value which is updated by clock source.

4.4.3 T0CON: Timer0 Control Register

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Bit No.	Control Bit	Description
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 0 = 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 Prescale value 101 = 1:64 Prescale value 100 = 1:32 Prescale value 011 = 1:16 Prescale value 010 = 1:8 Prescale value 001 = 1:4 Prescale value 000 = 1:2 Prescale value

NOTE: All bits are applicable to this exercise.

4.4.4 INTCON: Interrupt Control Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)
bit 7							bit 0

Bit No.	Control Bit	Description
Bit 7	**GIE/GIEH**	**Global Interrupt Enable bit** **When IPEN = 0:** **1 = Enables all unmasked interrupts** **0 = Disables all interrupts** When IPEN = 1: 1 = Enables all high priority interrupts 0 = Disables all high priority interrupts
Bit 6	PEIE/GIEL	Peripheral Interrupt Enable bit
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 1 = 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) 1 = At least one of the RB7:RB4 pins changed state (must be cleared in software) 0 = None of the RB7:RB4 pins have changed state

NOTE: Only highlighted bits are applicable to this exercise.

5. Time delay generation using timer:

The following steps are taken to generate a time delay using polling method
i. Load the proper value in T0CON indicating which timer mode, clock source, prescaler assignment.
ii. Load the registers TMR0H first and then TMR0L with initial count values. Delay generated is Depends upon the initial count value.
iii. Start the timer by setting TMR0ON bit in T0CON.
iv. Keep the monitoring the timer flag (TMR0IF).
v. Get out of the loop when TMR0IF becomes high.
vi. Stop the timer.
vii. Clear the TMR0IF flag for the next round.
viii. Go back to the step 2 to load TMR0H and TMR0L values.

The following steps are taken to generate a time delay using interrupt method
i. Load the proper value in T0CON indicating which timer mode, clock source, prescaler assignment.
ii. Load the registers TMR0H first and then TMR0L with initial count values. Delay generated is Depends upon the initial count value.
iii. Enable the Timer0 Interrupt and Global Interrupt using INTCON Register
iv. Start the timer by setting TMR0ON bit in T0CON.
v. Write the ISR at Interrupt vector 0x0018.
a. In ISR Clear the TMR0IF flag for the next round.
b. In ISR reload TMR0H and TMR0L values.

The size of the time delay depends on two factors, (a) the crystal frequency and (b) the timer's 16-bit register (c) the prescaler value. The largest delay is achieved by the making both TMR0H and TMR0L zero and using maximum prescaler value i.e. 1:256.

Formula for delay calculations using timer for crystal frequency of F_{osc} = 20 MHz, (TMR0H, TMR0L) = $(NNNNN)_{10}$ is as follows.

Clock applied to timer0 = Timer Clock Freq. = $\mathbf{F_T} = F_{osc}/4 = 20x10^6/4 = 5$ MHz

Therefore Time Period $\mathbf{T_F} = 1/5x10^6 = 0.2$

(a) Without Prescaler

Time Delay (Td) = (65536 - NNNNN) x Time Period $\mathbf{T_F}$

Therefore Maximum delay = (65536 – 00000) x 0.2 us = 13.10 ms

(a) With Prescaler (1:256)

Time Delay (Td) = (65536 - NNNNN) x (Timer Period $\mathbf{T_F}$ x Prescaler Value)

Therefore Maximum delay = (65536 – 00000) x (0.2 us x 256) = 3.36 seconds

5.1 Finding the Values to be loaded into timer for desired delay

(a) Without Prescaler

- Divide the desired time delay **Td** by $\mathbf{T_F}$ to get n(n = Td/0.2 us for Fosc = 20MHz)
- Perform 65536 – n. where n is the decimal value from step 1.
- Convert the result of step 2 to hexadecimal, where yyxx is the initial hex value to be loaded into the timer's registers.
- Set TMR0L = xx and TMR0H = yy.

(b) With Prescaler

- Multiply Timer period $\mathbf{T_F}$ by prescaler value to Teq
  $\mathbf{T_{eq} = T_F * Prescaler\ vale}$
- Divide the desired time delay **Td** by $\mathbf{T_{eq}}$ to get n
  n = Td/Teq
- Perform 65536 – n. where n is the decimal value from above.
- Convert the result of above step to hexadecimal, where yyxx is the initial hex value to be loaded into the timer's registers.
- Set TMR0L = xx and TMR0H = yy.

5.2 Calculation for generating square wave of 10Hz.

- Desired frequency of Square Wave Fs = 10 Hz
- Ts = 1/Fs = 1/10 = 0.1 second
- On Period and OFF period of square wave = Ts/2 = 0.05s = 50 ms
- To generate square wave the value of port pin RB0 should be toggled after every 50 ms.
- From above calculation we have to use prescaler (1:4 or 1:8) to generate delay of 50 ms since maximum delay generated by timer is 13.10 ms without using prescaler.

5.2.1 For Operating Frequency Fosc = 20MHz.

- Selecting Prescaler 1:8, Timer Clock source = Fosc/4
- Therefore Timer Clock Freq. = $\mathbf{F_T} = F_{osc}/4 = 20x10^6/4 = 5$ MHz
- $T_{eq} = T_F$ * Prescaler vale = 0.2 us x 8 = 1.6 us
- To get n, n= Td/Teq = 50ms/1.6us = 31250
- $(yyxx)_{10}$ = 65536 – n = 65536 – 31250 = 34286
- $(34286)_{10} = (85EE)_{16}$ should be loaded in TMR0H:TMR0L.

5.2.2 For Operating Frequency Fosc = 48MHz.

- Selecting Prescaler 1:16, Timer Clock source = Fosc/4
- Therefore Timer Clock Freq. = $\mathbf{F_T}$ = $F_{osc}/4$ = $48x10^6/4$ = 12 MHz
- T_{eq} = T_F * Prescaler vale = 0.08333 us x 16 = 1.333 us
- To get n, n= Td/Teq = 50ms/1.333 us = 37501
- $(yyxx)_{10}$ = 65536 – n = 65536 – 37501 = 28034
- $(28034)_{10}$ = $(6D82)_{16}$ should be loaded in TMR0H:TMR0L.

5.2 Calculation for generating square wave of 10Hz.

- Desired frequency of Square Wave Fs = 10 Hz
- Ts = 1/Fs = 1/10 = 0.1 second
- On Period and OFF period of square wave = Ts/2 = 0.05s = 50 ms
- To generate square wave the value of port pin RB0 should be toggled after every 50 ms.
- From above calculation we have to use prescaler (1:4 or 1:8) to generate delay of 50 ms since maximum delay generated by timer is 13.10 ms without using prescaler.

5.2.1 For Operating Frequency Fosc = 20MHz.

- Selecting Prescaler 1:8, Timer Clock source = Fosc/4
- Therefore Timer Clock Freq. = $\mathbf{F_T}$ = $F_{osc}/4$ = $20x10^6/4$ = 5 MHz
- T_{eq} = T_F * Prescaler vale = 0.2 us x 8 = 1.6 us
- To get n, n= Td/Teq = 50ms/1.6us = 31250
- $(yyxx)_{10}$ = 65536 – n = 65536 – 31250 = 34286
- $(34286)_{10}$ = $(85EE)_{16}$ should be loaded in TMR0H:TMR0L.

5.2.2 For Operating Frequency Fosc = 48MHz.

- Selecting Prescaler 1:16, Timer Clock source = Fosc/4
- Therefore Timer Clock Freq. = $\mathbf{F_T}$ = $F_{osc}/4$ = $48x10^6/4$ = 12 MHz
- T_{eq} = T_F * Prescaler vale = 0.08333 us x 16 = 1.333 us
- To get n, n= Td/Teq = 50ms/1.333 us = 37501
- $(yyxx)_{10}$ = 65536 – n = 65536 – 37501 = 28034
- $(28034)_{10}$ = $(6D82)_{16}$ should be loaded in TMR0H:TMR0L.

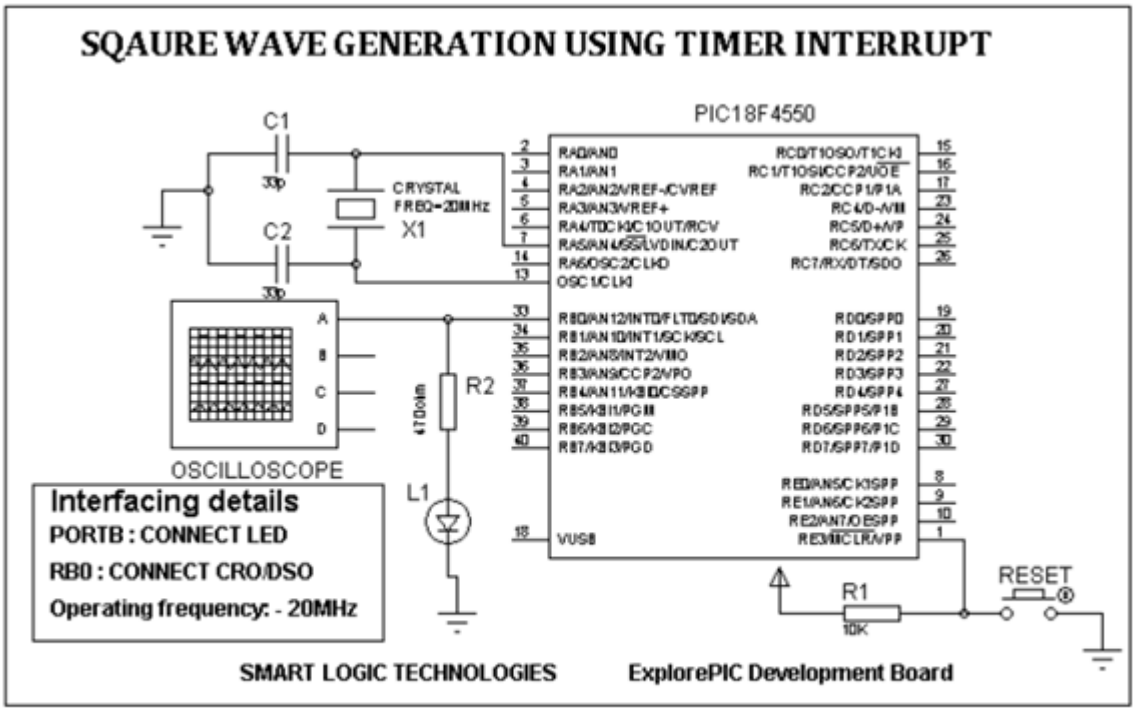


Figure 1.3: Interfacing 4X4 Matrix Keypad, 16x2 LCD to PIC Microcontroller

7. **Algorithm**

1. Configure Port pin RB0 as output and initial value 0.
2. Load the value 0x02 in T0CON indicating which timer in 16-bit mode clk source Fosc/4, prescaler assignment
3. Load the registers TMR0H first and then TMR0Lwith calculated initial count values.
4. Enable the Timer0 Interrupt and Global Interrupt using INTCON Register
5. Start the timer by setting TMR0ON bit in T0CON.
6. Write the ISR at Interrupt vector 0x0018 which contains following.

a. Toggle the PORT pin RB0

b. In ISR Clear the TMR0IF flag for the next round.

c. In ISR reload TMR0H and TMR0L values.

CONCLUSION:

Questions:

1. Write an embedded C program to generate the square wave of 20 ms delay with 50% duty cycle
2. Explain different modes of timers.
3. Explain with diagram Timer1 and Timer 2.
4. Explain delay calculation for 16 bit and 8 bit timer.
5. Why port b is called interrupt port.

STES

RMD,SINHGAD SCHOOL OF EGINEERING, WARJE 58

EXPERIMENT NO: 07

Name of the Student:-_____

Roll No._____ Subject:-_____

Date of Practical Performed:-_____ Staff Signature with Date & Marks

TITLE : On-chip ADC Programming

PROBLEM STATEMENT:

Interface analog voltage 0 – 5v to PIC microcontroller and Write a program in Embedded C to convert analog voltage into digital using on-chip ADC Module. Display the Digital result in BCD / Voltage form on 16x2 LCD display.

OBJECTIVE :

a. To understand the working of A/D converter.
b. To study the on-chip ADC section of PIC Microcontroller.
c. To interface analog input to PIC Microcontroller

S/W PACKAGES AND H/W USED:

MPLAB IDE, C18 Compiler, Explore PIC Development Board

THEORY:

1. ADC Devices:

Analog to digital converters are among the most widely used devices for data acquisitions. Digital computers use binary (discrete) value but in physical world everything is analog (continuous). A physical quantity is converted to electrical signals using device called transducer or also called as sensors. Sensors and many other natural quantities produce an output that is voltage (or current). Therefore we need an analog - to - digital converter to translate the analog signal to digital numbers so that the microcontroller can read and process them.

The ADC chips are either parallel or serial. In parallel ADC, we have 8 or more pins dedicated to bring out the binary data, but in serial ADC we have only one pin for data out.

Some of the major characteristics of ADC are a) resolution b) conversion time c) reference voltage (Vref).

1.1 Resolution

An ADC has an n bit resolution where n can be 8, 10, 16, 0r even 24 bits. The higher resolution ADC provides a smaller step size, where step size is smallest change that can be discerned by an ADC. This is shown in Table 8.1. Although the resolution for ADC chip is designed at the time of its design and cannot be changed, we can control the step size with the help of reference voltage (Vref).

1.2 Conversion time

Conversion time is defined as the time it takes the ADC to convert the analog input to digital (binary) number. The conversion time is dictated by the clock source connected to the ADC in addition to the method used for data conversion and technology used in the fabrication of the ADC chip.

In addition to conversion time, acquisition time is another major factor in judging an ADC. Acquisition time is defined as the time it takes to sample the analog voltage using sample and hold circuit. Sampled analog input is applied to actual conversion unit of ADC.

In some of the ADCs, conversion time doesn't contain the acquisition time but specified separately. Therefore effective conversion time is addition of specified conversion time and specified acquisition time.

1.3 Reference Voltage (Vref)

Vref is an input voltae used for the reference voltage. The voltage connected to this pin, along with resolution of the ADC chip, gives us the step size.

step size = Vref / 2^n where n= no. of bits

For example if the analog input range needs to be 0 to 3 Volts, Vref is connected to 3 Volts. That gives 3V/1024 = 2.92 mV step size for 10-bit ADC.

Vref (V)	Vin (V)	Step Size (mV)
5.00	0 to 5	5/1024 = 4.88
2.56	0 to 2.56	2.56/1024 = 2.5
1.024	0 to 1.024	1.024/1024 = 1

Table 8.1: Relation between Vref, Vin and step size

2. 10-Bit Analog-To-Digital Converter (A/D) Module of PIC uC

The Analog-to-Digital (A/D) converter module has 10 inputs for the 28-pin devices and 13 for the 40/44-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

2.2 Pin Description

Channel 10 (AN10)	34	RB1/AN10/INT1/SCK/SCL

Signal	Pin No.	Symbol
Channel 0 (AN0)	2	RA0/AN0
Channel 1 (AN1)	3	RA1/AN1
Channel 2 (AN2)	4	RA2/AN2/Vref-/CVref
Channel 3 (AN3)	5	RA3/AN3/Vref+
Channel 4 (AN4)	7	RA5/AN4/SS/HLVDIN/C2OUT
Channel 5 (AN5)	8	RE0/AN5/CK1SPP
Channel 6 (AN6)	9	RE1/AN6/CK2SPP
Channel 7 (AN7)	10	RE2/AN7/OESPP
Channel 8 (AN8)	35	RB2/AN8/INT2/VMO
Channel 9 (AN9)	36	RB3/AN9/VPO

Channel 11 (AN11)	37	RB4/AN11/KBIO/CSSPP
Channel 12 (AN12)	33	RB0/AN12/INT0/FLT0/SDI/SDA

2.3 ADC Register Map:

SFR	Description	Access	Reset Value	Address
ADCON0	A/D Control Register 0	Read/Write	0x00	0xFC2
ADCON1	A/D Control Register 1	Read/Write	0x00	0xFC1
ADCON2	A/D Control Register 2	Read/Write	0x00	0xFC0
ADRESH	A/D Result High Register	Read	unknown	0xFC4
ADRESL	A/D Result Low Register	Read	unknown	0xFC3

2.4 ADC Register Description

2.4.1 A/D Control Register 0 (ADCON0)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Bit No.	Control Bit	Description
Bit 7 - 6	Unimplemented	Read as '0'
Bit 5 - 2	CHS3:CHS0	Analog Channel Select bits

0000 = Channel 0 (AN0)	0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)	0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)	0101 = Channel 5 (AN5)
0110 = Channel 6 (AN6)	0111 = Channel 7 (AN7)
1000 = Channel 8 (AN8)	1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)	1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)	1101 = Unimplemented
1110 = Unimplemented	1111 = Unimplemented

Bit 1	GO/DONE	A/D Conversion Status bit 1 = A/D conversion in progress; 0 = A/D Idle
Bit 0	ADON	A/D On bit 1 = A/D converter module is enabled 0 = A/D converter module is disabled

NOTE: Only highlighted bits are applicable to this exercise.

ADCON0 Register in our Program

bit7-6 This two bits are Unimplimented so we write this bit as 00

bit5-2 Analog input is connected to Channel 1 (AN1). so we have to write CHS3:CHS0 analog channal select bits as 0001

bit1 Start Conversion by setting this bit. Later on poll the same conversion status. Bit0 To turn ON ADC set this bit 1.

2.4.2 A/D Control Register 1 (ADCON1)

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG0	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Bit No.	Control Bit	Description
Bit 7 - 6	Unimplemented	Read as '0'
Bit 5	**VCFG0**	Voltage Reference Configuration bit (VREF- source) 1 = VREF- (AN2); **0 = VSS**
Bit 4	**VCFG0**	Voltage Reference Configuration bit (VREF+ source) 1 = VREF+ (AN3); **0 = VDD**
bit 3-0	**PCFG3:PCFG0**	A/D Port Configuration Control bits shown in table 8.3 **1101 : Analog channel AN1**

NOTE: Only highlighted bits are applicable to this exercise.

ADCON1 Register in our Program

Bit7-6 This two bits are Unimplimented so we write this bit as 00

Bit 5 In our application (VREF-) supply as 0 volt so we write this bit as 0.

Bit 4 In our application (VREF+) supply as 5 volt so we write this bit as 0.

Bit3-0 Here we are using only one analog channal AN1. Therefor we write this bit as PCFG3:PCFG0 - 1101.

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Table 8.3: PCFG3:PCFG0: A/D Port Configuration Control bits:

2.4.3 A/D Control Register 2 (ADCON2)

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

Bit No.	Control Bit	Description
Bit 7	**ADFM**	**A/D Result Format Select bit** **1 = Right justified;** 0 = Left justified
Bit 6	**Un**	**Unimplemented** Read as '0'
Bit 5–3	**ACQT2:ACQT0**	**A/D Acquisition Time Select bits** 000=0 TAD | **001=2 TAD** | 010=4 TAD | 011 =6 TAD 100=8 TAD | 101=12 TAD | 110=16 TAD | 111 =20 TAD
bit 2–0	**ADCS2:ADCS0**	**A/D Conversion Clock Select bits** 000=FOSC/2 | 001=FOSC/8 | 010=FOSC/32 | 011=FRC* 100=FOSC/4 | 101=FOSC/16 | **110=FOSC/64** | 111=FRC* FRC* - clock derived from A/D RC oscillator

NOTE: Only highlighted bits are applicable to this exercise.

Therefore for 10bit ADC

Typical conversion time = 11 TAD = 11 * 0.7 us = 7.7 us

AD Clock Source (TAD)		Maximum Device Frequency
Operation	**ADCS2:ADCS0**	**PIC18FXXXX**
2 TOSC	000	2.86 MHz
4 TOSC	100	5.71 MHz
8 TOSC	001	11.43 MHz
16 TOSC	101	22.86 MHz
32 TOSC	010	45.71 MHz
64 TOSC	110	48.0 MHz
RC**(3)**	x11	1.00 MHz

Table 8.4 TAD vs. DEVICE OPERATING FREQUENCIES

2.4.3.3 A/D Acquisition Time Select bits (ACQT2:ACQT0)

Figure 8.1 shows the acquisition time (TACQ) is depends on amplifier settling time, capacitor charging time and temperature coefficient.

$$TACQ = TAMP + TC + TCOFF$$

Where

TAMP : Amplifier Settling Time = 0.2 us (Typical value)

TC : Holding Capacitor Charging Time

$= -(CHOLD)(RIC + RSS + RS)\ \ln(1/2048)\ \mu s$

$= -(25\ pF)\ (1\ k\Omega + 2\ k\Omega + 2.5\ k\Omega)\ \ln(0.0004883)\ \mu s$ (Typical value)

$= 1.05\ \mu s$

We need right justified data. Therefore ADFM = '1'.

If you write 1 then result is **right justified** as shown below;

15	14	13	12	11	10	9	8	
—	—	—	—	—	—			ADRESH
7	6	5	4	3	2	1	0	
								ADRESL

If you write 0 then result is **left justified** as shown below;

15	14	13	12	11	10	9	8	
								ADRESH
7	6	5	4	3	2	1	0	
		—	—	—	—	—	—	ADRESL

TCOFF: Temperature Coefficient = (Temp – 25°C)(0.02 µs/°C)
= (85°C – 25°C)(0.02 µs/°C) (Typical temp)
= 1.2 µs

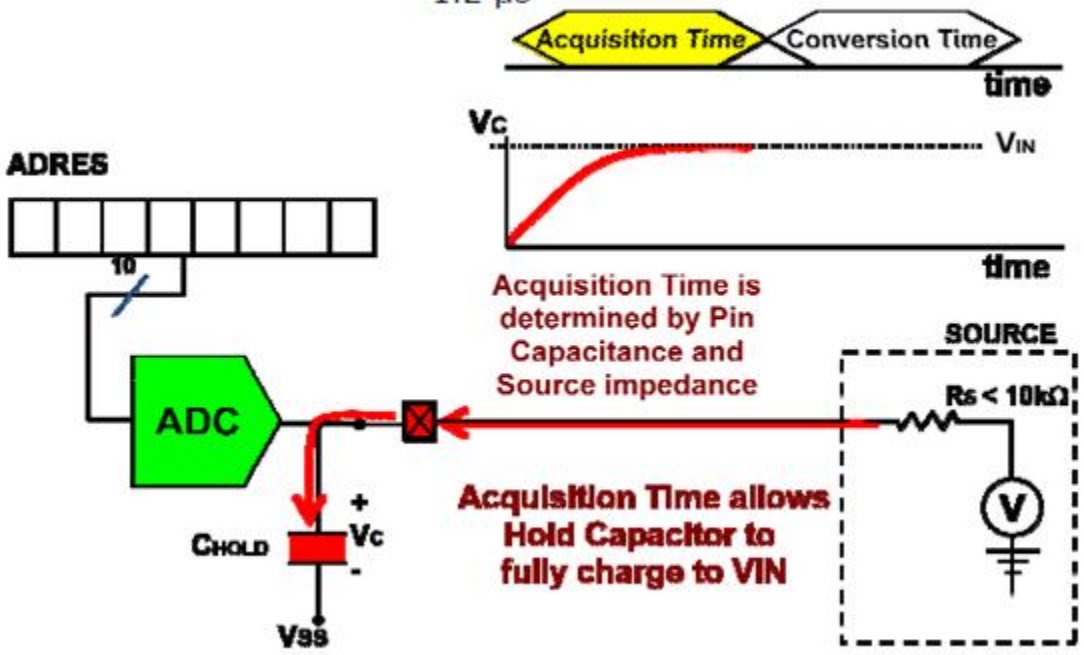


Figure 8.1 : Acquisition time of ADC module

2.4.3.2 A/D Conversion Clock Select bits (ADCS2:ADCS0)

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable. There are seven possible options for TAD as shown in table 8.4

For correct A/D conversions, the A/D conversion clock (TAD) must be as short as possible but greater than the minimum TAD. Table 8.4 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

TAD = 4 * Tosc where Tosc = 1/Fosc
Minimum TAD = 0.7 us (to be maintained as per datasheet)

Therefore minimum **calculated acquisition time** is

TACQ = 0.2 µs + 1.05 µs + 1.2 µs = 2.45 µs

Derived A/D acquisition time ≥ Calculated acquisition time TACQ

Derived A/D acquisition time = n * TAD

where n is depends on A/D Acquisition Time Select bits (ACQT2:ACQT0). So select A/D Acquisition Time Select bits (ACQT2:ACQT0) such that the we meet above timings.

2.5 Calculations

a. For Fosc = 20MHz

Tosc = 1/20MHz = 50 ns

Selected ADC clock source **ADCS2:ADCS0 = 110 i.e Fosc/64**

TAD = 64Tosc = 64 * 50 ns

TAD = 3.2 us which is much greater than minimum TAD = 0.7 us

Conversion Time = 11TAD = 11 * 3.2 us
= **35.2 us**

Selected A/D Acquisition Time Select bits **ACQT2:ACQT0 = 001 i.e n= 2**

Derived A/D acquisition time = n * TAD = 2 * TAD
= 2 * 3.2 us
= 6.4 us

Which is much greater than **calculated acquisition time = 2.4 us**

Effective conversion time = acquisition time + conversion time
= 35.2 us + 6.4 us
= 41.6 us

b. For Fosc = 48MHz

Tosc = 1/48MHz = 20.83 ns

Selected ADC clock source **ADCS2:ADCS0 = 110 i.e Fosc/64**

TAD = 64Tosc = 64 * 20.83 ns

TAD = 1.33 us which is greater than minimum TAD = 0.7 us

Conversion Time = 11TAD = 11 * 1.3 us
= **14.66 us**

Selected A/D Acquisition Time Select bits **ACQT2:ACQT0 = 001 i.e n= 2**

Derived A/D acquisition time = n * TAD = 2 * TAD
= 2 * 1.33 us
= 2.67 us

Which is greater than **calculated acquisition time = 2.4 us**

Effective conversion time = acquisition time + conversion time
= 14.66 us + 2.67 us
= 17.33 us

2.6 ADC operation

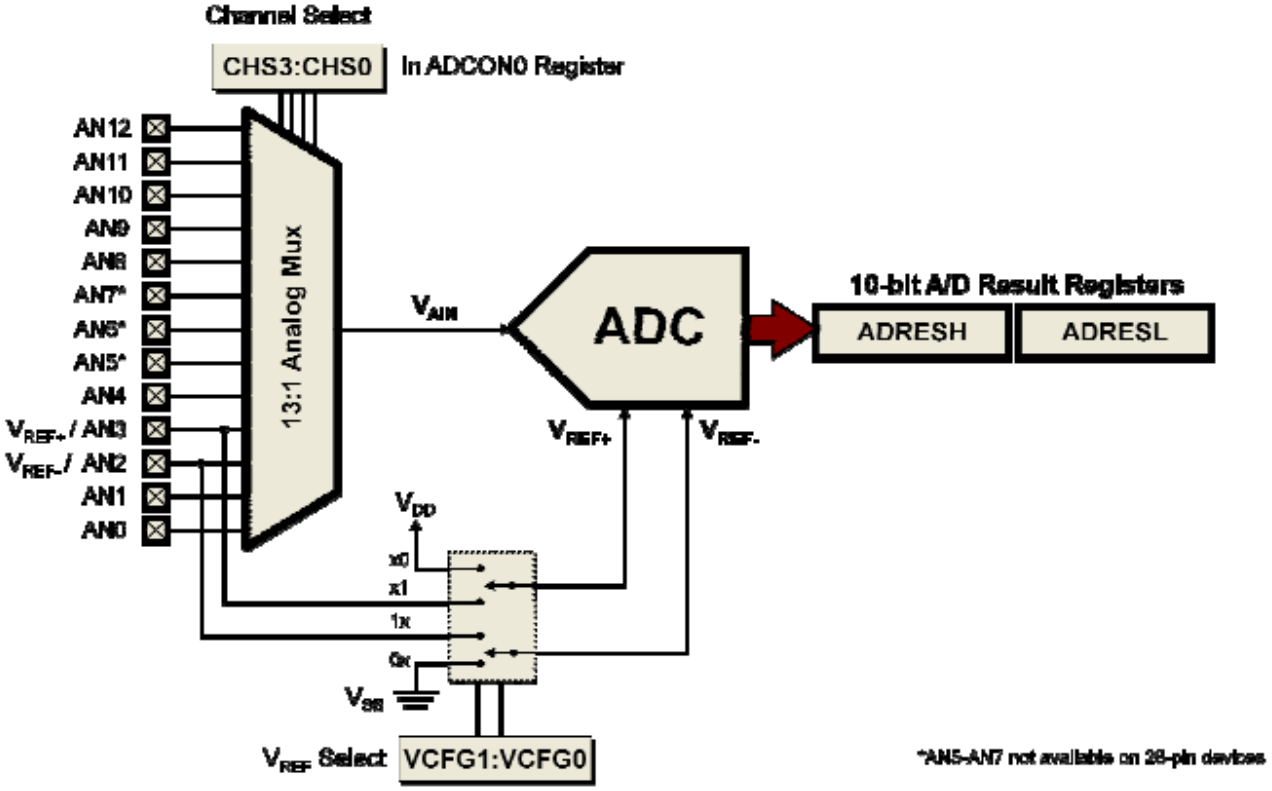
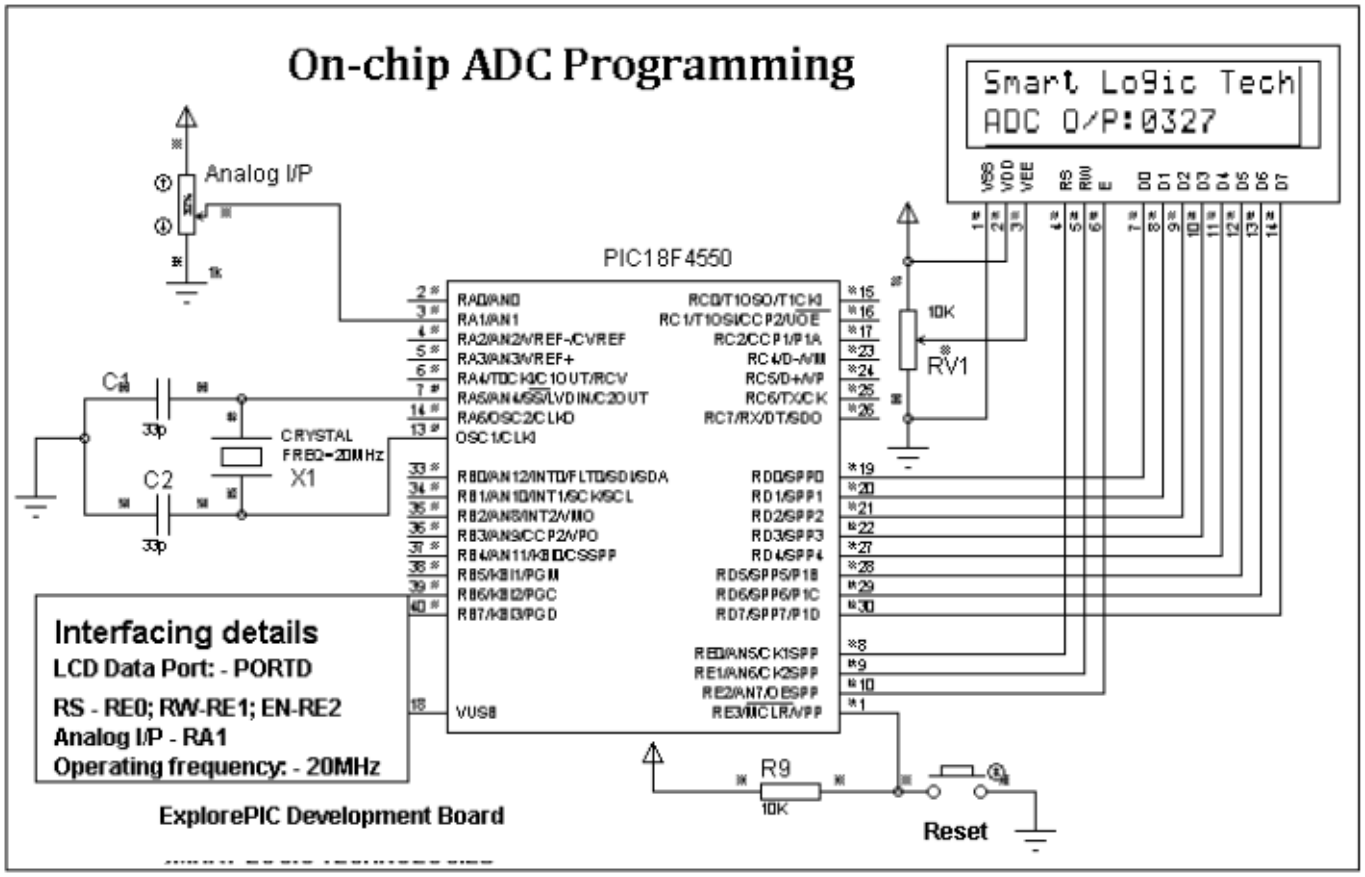


Figure 8.1: Block Diagram of on-chip ADC Module

The following steps should be followed to perform an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/DONE bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/DONE bit to be cleared
     OR
   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 3 TAD is required before the next acquisition starts.

4. Interfacing diagram



5. Conclusion:

Questions:

1. Daw an interfacing diagram to interface ADC .
2. write an embedded C program to accept data on any channel
3. Explain port configuration bit of ADC.
4. Explain ADCON register withdiagram.

STES

RMD,SINHGAD SCHOOL OF EGINEERING, WARJE 58

EXPERIMENT NO: 08

Name of the Student:-_____

Roll No._____ Subject:-_____

Date of Practical Performed:-_____ Staff Signature with Date & Marks

TITLE: Generation of PWM Signal

PROBLEM STATEMENT:

Interface DC Motor (9V – 12V ; 200 – 300 rpm) to PIC microcontroller. Write a program in Embedded C to control the speed of DC motor using on-chip PWM Module of PIC Microcontroller.

OBJECTIVE :

a. To understand the working of PWM.

b. To study the on-chip CCP Module (PWM section) of PIC Microcontroller.

c. To interface DC motor, via Driver IC L293D, to PIC Microcontroller

d. Apply the PWM signal to control the speed of DC Motor

S/W PACKAGES AND H/W USED:

MPLAB IDE, C18 Compiler, Explore PIC Development Board, DC Motor, CRO

THEORY:

1. PWM Technique:

2. CCP Module of PIC uC

Depending upon the device selected there are 1 – 4 CCP (Capture/Compare/PWM) modules in PIC microcontroller. Each module contains a 16-bit register, which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register.

Each Capture/Compare/PWM module is associated with a control register (generically, CCPxCON) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte). All registers are both readable and writable.

The CCP modules utilize Timers 1, 2 or 3, depending on the mode selected. Timer1 and Timer3 are available to modules in Capture or Compare modes, while Timer2 is available for modules in PWM mode.

The pin assignment for CCP2 (Capture input, Compare and PWM output) can change, based on device configuration. The CCP2MX Configuration bit determines which pin CCP2 is multiplexed to. By default, it is assigned to RC1 (CCP2MX = 1). If the Configuration bit is cleared, CCP2 is multiplexed with RB3. Changing the pin assignment of CCP2 does not automatically change any requirements for configuring the port pin. Users must always verify that

the appropriate TRIS register is configured correctly for CCP2 operation, regardless of where it is located.

2.1 Capture Mode Applications

- Event arrival time recording
- Period measurement
- Pulse-width measurement
- Interrupt generation
- Event counting
- Time reference
- Duty cycle measurement

2.2 Compare Mode Applications

Generate …

- Single pulse
- Train of pulses
- Periodic waveform Start ADC conversion Time reference

3. PWM Mode of CCP Module

In Pulse-Width Modulation (PWM) mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP2 pin is multiplexed with a PORTB or PORTC data latch, the appropriate TRIS bit must be cleared to make the CCP2 pin an output. A PWM output has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

The simplified block diagram of the CCP1 configured to generate PWM is shown in Figure 5.1, with example waveforms in Figure 5. 2. The block diagram for CCP2 is similar to CCP1. It can be seen that Figure 5.1 contains CCPR1L, Timer 2, comparator and the PR2 register working together. Although not shown, Timer 2 is still driven from the on-chip oscillator, through its own prescaler. The comparator output drives an R–S flip-flop. When the value in PR2 equals Timer 2, then the comparator clears the timer and sets the flip-flop, whose output

goes high. This is seen in Figure 5.2. This action sets the PWM period.

Having established the PWM period, let us consider how the pulse width is determined. A second Compare register arrangement is introduced to do this. This is made up of the CCPR1H register, plus a second comparator. As the logic of the diagram shows, every time this comparator finds equal input values, it resets the output flip-flop, clearing the output to zero. It is this comparator that determines the pulse width. Again, this is shown in Figure 5.2. To change the pulse width, the programmer writes to the CCPR1L register, which acts as a buffer. Its value is transferred to CCPR1H only when a PWM cycle is complete, to avoid output errors in the process. The block diagram is made more complex because three of the registers are 'stretched', to make them potentially 10-bit instead of 8-bit. This increases the resolution. CCPR1L uses two bits of the CCP1CON register. CCPR1H is extended with an internal 2-bit latch, while the extension to Timer 2 is as described in Note 1 of Figure 5.1. Because of these two extra bits, in its 10-bit version it is effectively clocked direct from the internal oscillator signal, undivided. If the prescaler is used, then it acts on this frequency, not the usual Fosc/4. Notice, however, that the PR2 register remains at eight bits. This means that the PWM period has only an 8-bit equivalent resolution.

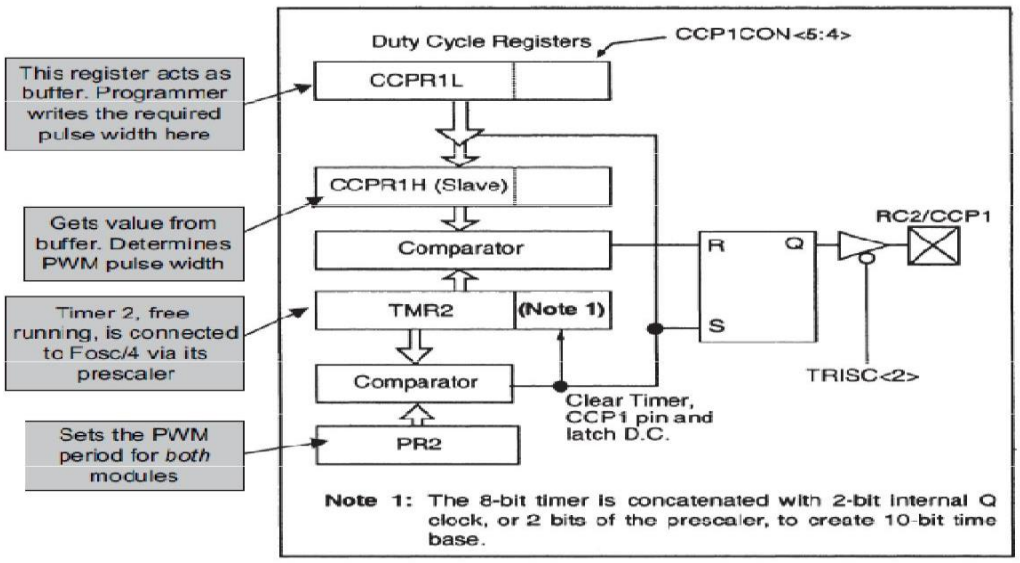


Figure 5.1 : PWM Block Diagram

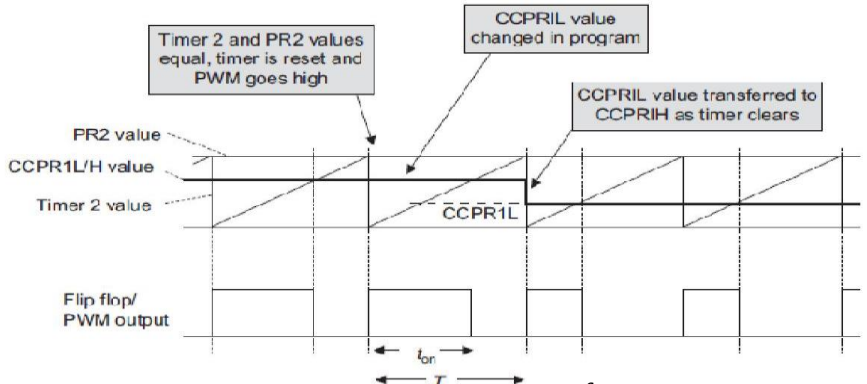


Figure 5.2: PWM output waveform

3.1 PWM Period and Duty Cycle

The PWM period T is determined by the interaction of the PR2 register and the ei bits of Timer 2. It may be calculated as follows:

PWM Period = T_{pwm} = (PR2 + 1) x (Timer 2 input Clock)

PWM Period = T_{pwm} = (PR2 + 1) x (4 * Tosc * TMR2 Prescaler value

Therefore to find the PR2 for desired PWM Period

$$PR2 = \left\{\left(\frac{\mathbf{Tpwm}}{\mathbf{4} * \boldsymbol{Tosc} * (\boldsymbol{TMR2\ Prescaler\ Value})}\right) - \mathbf{1}\right\}$$

Above equation can be represented in terms of frequency as below.

$$PR2 = \left\{\left(\frac{\boldsymbol{Fosc}}{\mathbf{4} * \boldsymbol{Fpwm} * (\boldsymbol{TMR2\ Prescaler\ Value})}\right) - \mathbf{1}\right\}$$

Where,

Timer 2 input Clock = 4 * Tosc * TMR2 Prescaler value
Fosc = Input Crystal Oscillator Frequency
Tosc = input clock period = 1/Fosc
Fpwm = PWM frequency
Tpwm = PWM Period = 1/Fpwm

The PWM pulse width **Ton** is determined by the interaction of the extended CCPR register (all 10 bits of it) and the extended (10-bit) Timer 2. It may be calcula as follows:

Ton = (Pulse width register) x (PWM Timer input clock period)

Where, 'PWM timer input clock period' is the period of the clock input to extended Timer 2 and 'pulse width register' is the value in the extended CCPR register.

Hence,

Ton = (CCPR1L: CCP1CON<5:4>) x (Tosc * TMR2 Prescaler value)

Note that there is not here a factor of four with the Tosc term, as we are using bits of Timer 2. Rearranging the above equation, to find value of pulse wi register **(CCPR1L: CCP1CON<5:4>).**

$$\boldsymbol{CCPR1L{:}\,CCP1CON} < 5{:}4 > = \left\{\left(\frac{\boldsymbol{Ton}}{\boldsymbol{Tosc} * (\boldsymbol{TMR2\ Prescaler\ Value})}\right)\right\}$$

Above equation can be represented in terms of frequency and percentage Duty cycle (%DCpwm) as below.

$$\boldsymbol{CCPR1L{:}\,CCP1CON} < 5{:}4 > = \left\{\left(\frac{\boldsymbol{\%DCpwm} * \boldsymbol{Fosc}}{\mathbf{100} * \boldsymbol{Fpwm} * (\boldsymbol{TMR2\ Prescaler\ Value})}\right)\right\}$$

Note: Choose $TMR2_{PRE}$ to ensure that PR2 is in the range of 0 to 255 for the desi PWM frequency and CCPR1L:CCP1CON<5:4> is in the range of 0 to 1023 for desired PWM duty cycle.

3.2 Pin Description

Signal	Pin No.	Symbol
CCP2(1)	16	RC1/T1OSI/CCP2(1)/UOE
CCP1	17	RC2/CCP1/P1A
CCP2(1)	36	RB3/AN9/CCP2(1)/VPO
Note 1: RB3 is the alternate pin for CCP2 multiplexing.		

3.3 PWM Register Map:

SFR	Description	Access	Reset Value	Address
CCP1CON	Standard CCP1 Control Reg.	Read/Write	0x00	0xFBD
CCPR1L	CCP1 Register low byte	Read/Write	unknown	0xFBE
CCPR1H	CCP1 Register high byte	No access	unknown	0xFBF
CCP2CON	Standard CCP2 Control Reg.	Read/Write	0x00	0xFBA
CCPR2L	CCP2 Register high byte	Read/Write	unknown	0xFBB
CCPR2H	CCP2 Register high byte	No access	unknown	0XFBC
T2CON	Timer 2 Control Register	Read/Write	0x00	0xFCA
TMR2	Timer 2 Register	Read/Write	0x00	0xFCC
PR2	Timer 2 Period register	Read/Write	0xFF	0xFCB

3.4 PWM Mode Register Description

3.4.1 Standard CPPx Control Register (CCPxCON – CCP1CON/CCP2CON)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—(1)	—(1)	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

Bit No.	Control Bit	Description
Bit 7 - 6	Unimplemented	Read as '0'
Bit 5 - 4	**DCxB1:DCxB0**	**PWM Duty Cycle Bit 1 and Bit 0 for CCPx Module** Unused in Capture mode & Compare mode: **PWM mode:** These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.
Bit 3 - 0	**CCPxM3:CCPxM0**	**CCPx Module Mode Select bits** 0000 = CCP disabled (resets CCPx module) 0001 = Reserved 0010 = Compare mode: toggle output on match (CCPxIF bit is set) 0011 = Reserved 0100 = Capture mode: every falling edge 0101 = Capture mode: every rising edge 0110 = Capture mode: every 4th rising edge 0111 = Capture mode: every 16th rising edge 1000 = Compare mode: initialize CCPx pin low; on

		compare match, force CCPx pin high (CCPxIF bit is set) 1001 = Compare mode: initialize CCPx pin high; on compare match, force CCPx pin low (CCPxIF bit is set) 1010 = Compare mode: generate software interrupt on compare match (CCPxIF bit is set, CCPx pin reflects I/O state) 1011 = Compare mode: trigger special event, reset timer, start A/D conversion on CCP2 match (CCPxIF bit is set) **11xx = PWM mode**

NOTE: Only highlighted bits are applicable to this exercise.

3.4.2 Timer 2 Control Register

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Bit No.	Control Bit	Description
Bit 7	Unimplemented	Read as '0'
Bit 6 - 3	T2OUTPS3:T2OUTPS0:	Timer2 Output Postscaler Select bits (1:1 to 1:16) 0000 = 1:1 Postscaler 0001 = 1:2 Postscaler • • 1111 = 1:16 Postscaler
Bit 2	TMR2ON:	Timer2 On bit 1 = Timer2 is on, 0 = Timer2 is off
Bit 1 - 0	T2CKPS1:T2CKPS0	Timer2 Clock Prescaler Select bits 00 = Prescaler is 1:1 01 = Prescaler is 1:4 1x = Prescaler is 1:16

4. Timer 2 Module

The Timer2 module timer incorporates the following features:

- 8-bit timer and period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable Postscaler (1:1 through 1:16)
- Interrupt on TMR2 to PR2 match
- Optional use as the shift clock for the MSSP module

The module is controlled through the T2CON register which enables or disables the timer and configures the prescaler and Postscaler. Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption. A simplified block diagram of the module is shown in Figure 9.3.

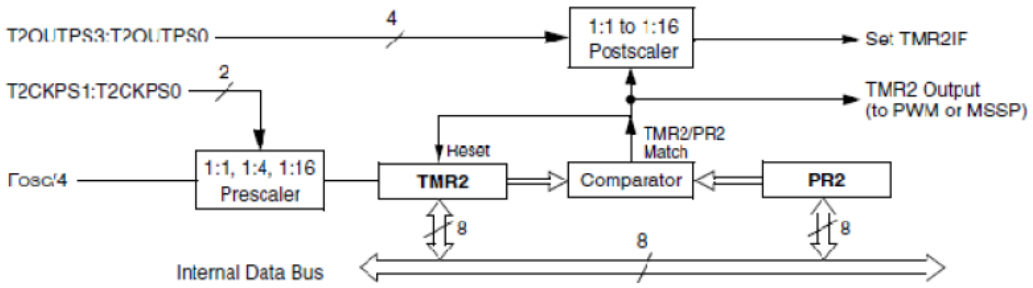


Figure 9.3 Simplified Block Diagram of Timer2 Module

4.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (FOSC/4). A 2-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by- 16 prescale options. These are selected by the prescaler control bits, T2CKPS1:T2CKPS0 (T2CON<1:0>). The value of TMR2 is compared to that of the period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/ Postscaler.

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh. TMR2 is not cleared when T2CON is written. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset, MCLR Reset, WDT Reset or Brown-out)

4.2 Timer2 Interrupt

Timer2 also can generate an optional device interrupt. The Timer2 output signal (TMR2 to PR2 match) provides the input for the 4-bit output counter/postscaler. This counter generates the TMR2 match interrupt flag which is latched in TMR2IF (PIR1<1>). The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE (PIE1<1>). A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS3:T2OUTPS0 (T2CON<6:3>).

4.3 TMR2 Output

The unscaled output of TMR2 is available primarily to the CCP modules, where it is used as a time base for operations in PWM mode. Timer2 can be optionally used as the shift clock source for the MSSP module operating in SPI mode.

5. Calculation for PWM operation

Desired PWM Frequency = 4 KHz

Therefore

$$\text{PWM Period} = \frac{1}{\text{PWM Frequency}} = \frac{1}{4 \text{ X } 10^3} = 250 \text{ X } 10^{-6}$$

PWM period = 250 us

a. For Fosc = 20 MHz

$$\text{Tosc} = \frac{1}{\text{Fosc}} = \frac{1}{20 \text{ x} 10^6}$$

Tosc= 0.05 us

TMR2 Prescaler Value = 16,

$$\text{PR2} = \left\{\left(\frac{\text{PWM Period}}{4 * \text{Tosc} * (\text{TMR2 Prescale Value})}\right) - 1\right\}$$

$$\text{PR2} = \left\{\left(\frac{250 \text{ x } 10^{-6}}{4 \text{ x } 0.05 \text{ x } 10^{-6} \text{ x } 16}\right) - 1\right\}$$

$$\text{PR2} = \left\{\left(\frac{250 \text{ x } 10^{-6}}{3.2 \text{ x } 10^{-6}}\right) - 1\right\}$$

$$\mathbf{PR2 = \{78 - 1\} = \{77\} = 0x4E}$$

Calculate the Duty Cycle (CCPRXL:CCPCON<5:4>) 20%, 40%, 60% and 80% for the PWM Frequency of 4KHz,

$$\boldsymbol{CCPR1L{:}\,CCP1CON} < 5{:}4 > = \left\{\left(\frac{\%\boldsymbol{DCpwm} * \boldsymbol{Fosc}}{\boldsymbol{100} * \boldsymbol{Fpwm} * (\boldsymbol{TMR2\ Prescaler\ Value})}\right)\right\}$$

- Duty Cycle **20%**

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = \left\{\frac{20 * \ 20\text{x}10^6}{100 * 4\ x\ 10^3\ * 16}\right\}$$

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = 62 = \ 0x3E \ \cdots \boxed{0000|0011|1110} \cdots \boxed{\boxed{0000|1111|}\boxed{|10|}}$$

$$\text{CCPR1L} = \mathbf{0x0F} \ : \ \text{CCP1CON} < 5{:}4 > = \mathbf{10}$$

- Duty Cycle **40%**

$$(\text{CCPR1L:CCP1CON} < 5:4 >) = \left\{\frac{40 * 20\text{x}10^6}{100 * 4\,x\,10^3 * 16}\right\}$$

$$(\text{CCPR1L:CCP1CON} < 5:4 >) = 125 = 0x7D \cdots \boxed{0000|0111|1101} \cdots \boxed{\boxed{0001|1111|}\boxed{|01|}}$$

$$\text{CCPR1L} = \boldsymbol{0x1F} \;:\; \text{CCP1CON} < 5:4 > = \mathbf{01}$$

- Duty Cycle **60%**

$$(\text{CCPR1L:CCP1CON} < 5:4 >) = \left\{\frac{60 * 20\text{x}10^6}{100 * 4\,x\,10^3 * 16}\right\}$$

$$(\text{CCPR1L:CCP1CON} < 5:4 >) = 187 = 0xBB \cdots \boxed{0000|1011|1011} \cdots \boxed{\boxed{0010|1110|}\boxed{|11|}}$$

$$\text{CCPR1L} = \boldsymbol{0x2E} \;:\; \text{CCP1CON} < 5:4 > = \mathbf{11}$$

- Duty Cycle **80%**

$$(\text{CCPR1L:CCP1CON} < 5:4 >) = \left\{\frac{80 * 20\text{x}10^6}{100 * 4\,x\,10^3 * 16}\right\}$$

$$(\text{CCPR1L:CCP1CON} < 5:4 >) = 250 = 0xFA \cdots \boxed{0000|1111|1010} \cdots \boxed{\boxed{0011|1110|}\boxed{|10|}}$$

$$\text{CCPR1L} = \boldsymbol{0x3E} \;:\; \text{CCP1CON} < 5:4 > = \mathbf{10}$$

	Duty Cycle	CCPR1L Value	CCP1CON<5:4> DC1B1:DC1B0
Fosc = 20 MHz PWM = 4 Khz PR2 = 0x4E	20%	0b00001111 (0x0F)	0b10 (0x02)
	40%	0b00011111 (0x1F)	0b01 (0x01)
	60%	0b00101110 (0x2E)	0b11 (0x03)
	80%	0b00111110 (0x3E)	0b10 (0x02)

b. For Fosc = 48 MHz

$$\text{Tosc} = \frac{1}{\text{Fosc}} = \frac{1}{48\text{ x}10^6}$$

Tosc= 20.83 ns
PWM period = 250 us
TMR2 Prescaler Value = 16,

$$\text{PR2} = \left\{\left(\frac{\text{PWM Period}}{4 * \text{Tosc} * (\text{TMR2 Prescale Value})}\right) - 1\right\}$$

$$\text{PR2} = \left\{\left(\frac{250\text{ x }10^{-6}}{4\text{ x }20.83\text{ x }10^{-9}\text{ x }16}\right) - 1\right\}$$

$$\text{PR2} = \left\{\left(\frac{250\text{ x }10^{-6}}{1.33\text{ x }10^{-6}}\right) - 1\right\}$$

$$\mathbf{PR2} = \{\mathbf{187} - \mathbf{1}\} = \{\mathbf{186}\} = \mathbf{0xBA}$$

Calculate the Duty Cycle (CCPRXL:CCPCON<5:4>) 20%, 40%, 60% and 80% for the PWM Frequency of 4KHz,

$$\boldsymbol{CCPR1L{:}\,CCP1CON} < 5{:}4 > = \left\{\left(\frac{\%\boldsymbol{DCpwm} * \boldsymbol{Fosc}}{\mathbf{100} * \boldsymbol{Fpwm} * (\boldsymbol{TMR2\ Prescaler\ Value})}\right)\right\}$$

- Duty Cycle **20%**

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = \left\{\frac{20 * 48\text{x}10^6}{100 * 4\,x\,10^3 * 16}\right\}$$

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = 150 = 0x96 \cdots \boxed{0000|1001|0110} \cdots \boxed{\boxed{0010|0101|}\boxed{|10|}}$$

$$\text{CCPR1L} = \mathbf{0x25} \ \vdots \ \text{CCP1CON} < 5{:}4 > = \mathbf{10}$$

- Duty Cycle **40%**

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = \left\{\frac{40 * 48\text{x}10^6}{100 * 4\,x\,10^3 * 16}\right\}$$

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = 300 = 0x12C \cdots \boxed{0001|0010|1100} \cdots \boxed{\boxed{0100|1011|}\boxed{|00|}}$$

$$\text{CCPR1L} = \mathbf{0x4B} \ \vdots \ \text{CCP1CON} < 5{:}4 > = \mathbf{00}$$

- Duty Cycle **60%**

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = \left\{\frac{60 * 48\text{x}10^6}{100 * 4\,x\,10^3 * 16}\right\}$$

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = 450 = 0x1C2 \cdots \boxed{0001|1100|0010} \cdots \boxed{\boxed{0111|0000|}\boxed{|10|}}$$

$$\text{CCPR1L} = \mathbf{0x70} \ \vdots \ \text{CCP1CON} < 5{:}4 > = \mathbf{10}$$

- Duty Cycle **80%**

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = \left\{\frac{80 * 48\text{x}10^6}{100 * 4\,x\,10^3 * 16}\right\}$$

$$(\text{CCPR1L:CCP1CON} < 5{:}4 >) = 600 = 0x258 \cdots \boxed{0010|0101|1000} \cdots \boxed{\boxed{1001|0110|}\boxed{|00|}}$$

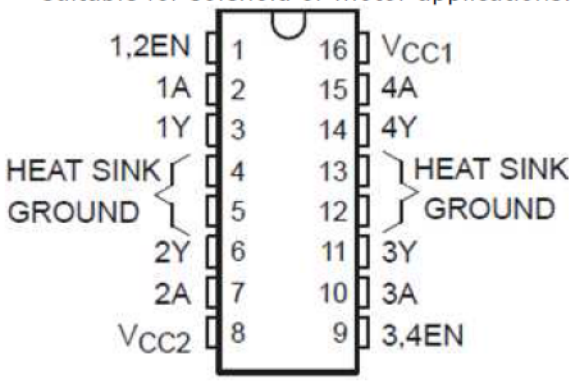
$$\text{CCPR1L} = \mathbf{0x96} \ \vdots \ \text{CCP1CON} < 5{:}4 > = \mathbf{00}$$

Fosc = 48 MHz PWM = 4 Khz PR2 = 0xBA	**Duty Cycle**	**CCPR1L Value**	**CCP1CON<5:4> DC1B1:DC1B0**
	20%	0b00100101 (0x25)	0b10 (0x02)
	40%	0b00011011 (0x1B)	0b00 (0x00)
	60%	0b01110000 (0x70)	0b10 (0x02)
	80%	0b10010110 (0x96)	0b00 (0x00)

6. Quadruple half H Drivers (L293D)

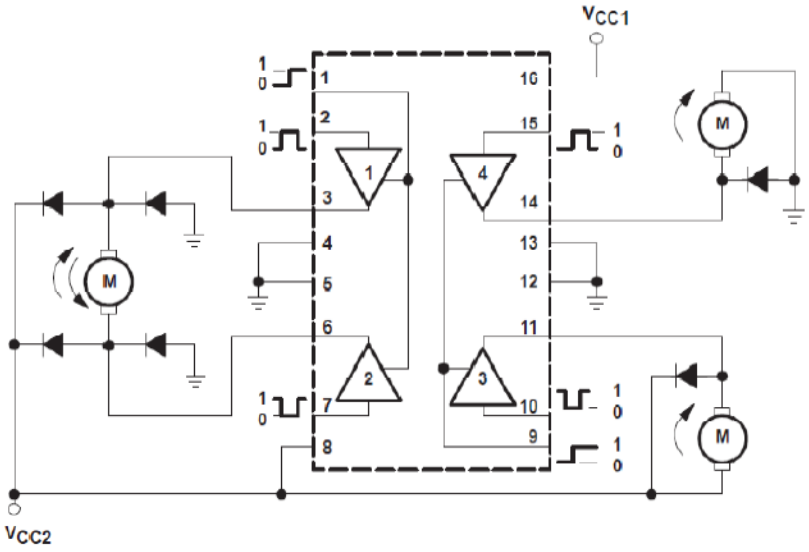
The L293D is quadruple high-current half-H drivers. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. The device is designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications. All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source.

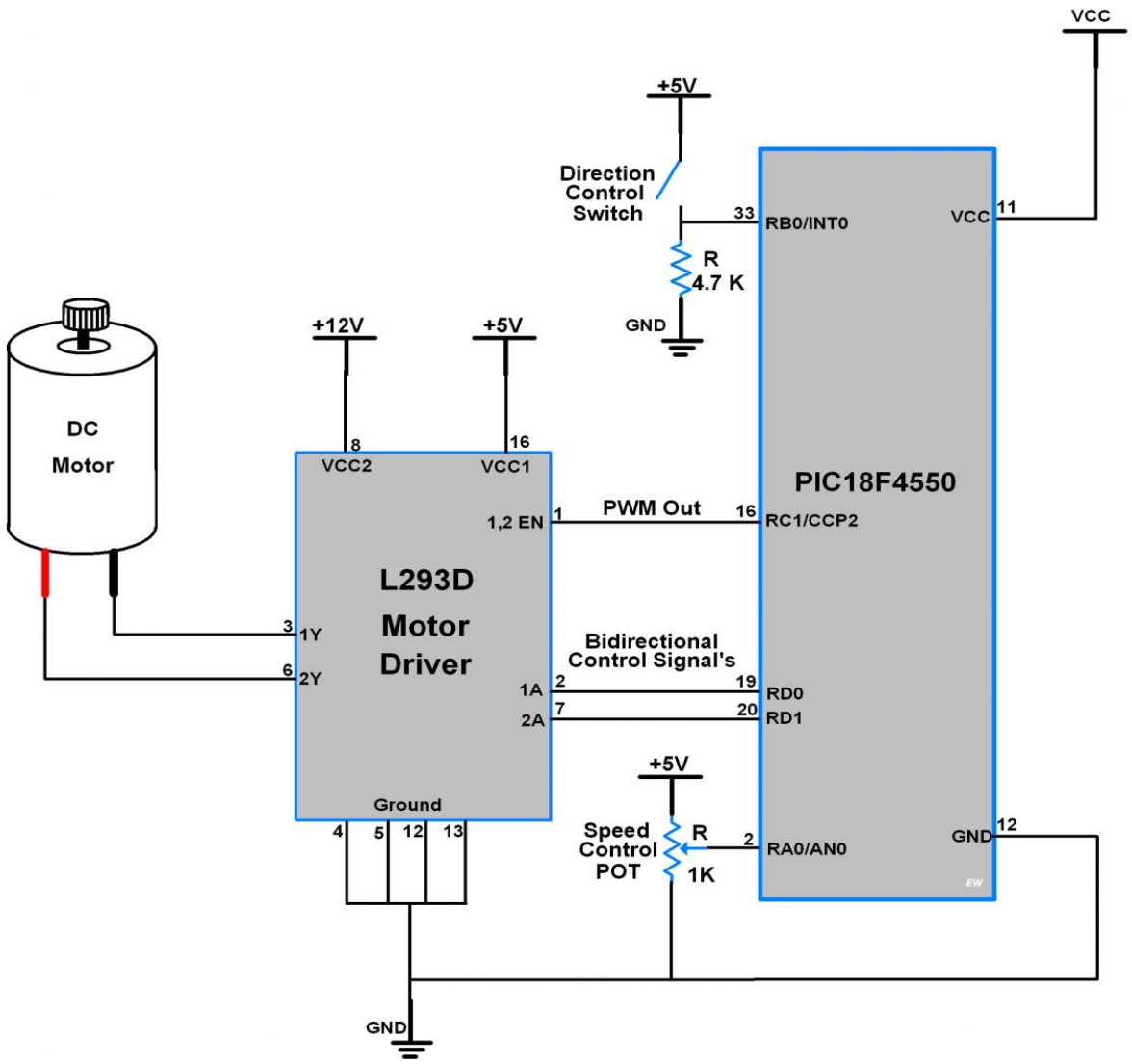
Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN as shown in pin diagram and function table. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.



FUNCTION TABLE
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z





PIC18F4550 Interface with DC Motor

Conclusion:

Questions:

1. Explain the PWM mode of PIC18 F in Details
2. Write a program for 1KHz, 10 % duty cycle PWM waveform
3. Write a c code for speed control of DC motor.
4. Write a short note on CCP module.