Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Microcontroller – TE (2019 Course) (Semester I)

Course: Microcontroller Lab Manual

Prepared by

Pranita Bhosale

Assistant Professor

Department of E&TC Engineering,

Dr. D.Y. Patil Institute of Technology,

Pimpri, Pune.

Dr. D. Y. Patil Unitech Society's

Dr. D. Y. Patil Institute of Technology

Pimpri, Pune

Department of Electronics & Telecommunication Engineering

Vision of the Institute

Empowerment Through Knowledge

Mission of the Institute

Developing human potential to serve the nation by

- *Dedicated efforts for quality Education*
- *Yearning to promote research and development.*
- *Persistent endeavor to imbibe moral and professional ethics.*
- *Inculcating the concept of emotional intelligence.*
- *Emphasizing extension work to reach out to the society.*
- *Treading the path to meet the future challenges*

Dr. D. Y. Patil Unitech Society's

Dr. D. Y. Patil Institute of Technology

Pimpri, Pune

Department of Electronics & Telecommunication Engineering

Vision of the Department

To be a distinct department building globally competent Electronics and Telecommunication professionals

Mission of the Department

- *Nurturing the spirit of innovation & creativity by providing conducive learning*
- *Enhancing potential of students to be globally competent by providing platform to build skills in advanced technologies and inculcate life-long learning*
- *Enabling students to imbibe social as well as ethical values as inner strength through educational experiences and collaborations*

Dr. D. Y. Patil Unitech Society's

Dr. D. Y. Patil Institute of Technology

Pimpri, Pune

Department of Electronics & Telecommunication Engineering

Programme Educational Objectives (PEOs)

PEO1:

Possess strong basics of science and engineering as well as critical problem skills to innovate and solve real-life engineering problems.

PEO2:

Acquire technical competency in Electronics & Telecommunication field leading to higher studies, career/technical profession and

PEO3:

Exhibit proficiency in constantly evolving multidisciplinary approach and conduct themselves in professional and ethical manner at all levels.

PEO4:

Demonstrate attributes for need based learning and utilize their engineering skills along with effective communication skills and spirit of team work in social context.

Dr. D. Y. Patil Unitech Society's

Dr. D. Y. Patil Institute of Technology

Pimpri, Pune

Department of Electronics & Telecommunication Engineering

Programme Specific Outcomes (PSOs)

PSO1:

Identify the problem and creat a solution relevant to Electronics & Telecommunication Engineering.

PSO2:

Act effectively, ethically and responsibly to tackle societal and environmental issues related to Electronics & Telecommunication Engineering.

PSO3:

Upgrade to the latest trends and accept the advancement of the technologies in the field of Electronics & Telecommunication Engineering.

Dr. D. Y. Patil Unitech Society's

Dr. D. Y. Patil Institute of Technology

Pimpri, Pune

Department of Electronics & Telecommunication Engineering

Programme Outcomes (POs)

PO1	**Engineering Knowledge**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem Analysis**	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design & Development of Solutions**	Design solutions for complex engineering problems anddesign system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern Tools Usage**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment & Sustainability**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual & Team work**	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication**	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management & Finance**	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments
PO12	**Lifelong Learning**	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

T.E. – (Semester I)

Course: Microcontroller

LAB MANUAL

INDEX

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

T.E. – (Semester I)

Subject: Microcontroller

COURSE OBJECTIVES

• Understand architecture and features of 8051 and PIC18FXX Microcontroller.

• Learn interfacing of real-world peripheral devices with microcontroller.

• Explore different features of PIC 18F Microcontroller with Architecture.

• Use concepts of timers and interrupts of PIC 18 in programming.

• Design and develop microcontroller based embedded application.

• Demonstrate real life applications using PIC 18.

LAB OBJECTIVES

- To acquire knowledge on Microcontrollers and interfacing devices.
- To understand the impact of 8051 and PIC microcontrollers in engineering applications.
- Learn about Architecture and operation of 8051 and PIC18 microcontroller.
- To learn to perform basic operations of 8051 and PIC18 microcontroller.
- To use Keil µVision for programming of 8051 Programming, execution and debugging
- To use MPLAB for PIC Programming, execution and debugging.
- To use SST Flash to import hex file in 8051 trainer kit
- To use PIC Loader to import hex file in 8051 trainer kit
- To explain writing embedded C language programs using subroutines for generation of delays, Counters, configuration of SFRs for serial communication and timers.
- To perform interfacing of stepper motor and dc motor for controlling the speed.
- To interface various peripherals
- To handle interrupts.
- To generate delay using timers.

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

T.E. – (Semester I)

Subject: Microcontroller

COURSE OUTCOME

CO1: Understand the fundamentals of microcontroller and programming.

CO2: Interface various electronic components with microcontrollers.

CO3: Analyze the features of PIC 18F XXXX.

CO4: Describe the programming details in peripheral support.

CO5: Develop interfacing models according to applications.

CO6: Evaluate the serial communication details and interfaces.

LAB OUTCOMES

- Create embedded C language programs for data transfer, arithmetic instructions.
- Create embedded C language programs using subroutines for generation of delays, counters, configuration of SFRs for serial communication and timers.
- Perform interfacing of stepper motor and dc motor for controlling the speed.
- Program 8051/PIC18 microcontroller to meet the requirements of the user.
- Interface peripherals like switches, LEDs, 7 segment, LCD, stepper motor, DC Motor
- Handle Timers, interrupts
- Design a microcontroller development board to meet the requirements of the user
- Work with a small team to carryout experiments using microcontroller concepts and prepare reports that present lab work.

Savitribai Phule Pune University, Pune
T.E. (Electronics& Telecommunication Engineering) 2019 Course
(With effect from Academic Year 2021-22)

Semester-V

Course Code	Course Name	Teaching Scheme (Hours/Week)			Examination Scheme and Marks						Credit			
		Theory	Practical	Tutorial	In-Sem	End-Sem	TW	PR	OR	Total	TH	PR	TUT	Total
304181	Digital Communication	03	-	-	30	70	-	-	-	100	03	-	-	03
304182	Electromagnetic Field Theory	03	-	01	30	70	25	-	-	125	03	-	01	04
304183	Database Management	03	-	-	30	70	-	-	-	100	03	-	-	03
304184	Microcontrollers	03	-	-	30	70	-	-	-	100	03	-	-	03
304185	Elective - I	03	-	-	30	70	-	-	-	100	03	-	-	03
304186	Digital Communication Lab	-	02	-	-	-	-	50	-	50	-	01	-	01
304187	Database Management Lab	-	02	-	-	-	-	-	25	25	-	01	-	01
304188	Microcontroller Lab	-	02	-	-	-	-	50	-	50	-	01	-	01
304189	Elective I Lab	-	02	-	-	-	-	25	-	25	-	01	-	01
304190	Skill Development	-	02	-	-	-	25	-	-	25	-	01	-	01
304191A	Mandatory Audit Course 5 &	-	-	-	-	-	-	-	-	-	-	-	-	-
	Total	**15**	**10**	**01**	**150**	**350**	**50**	**125**	**25**	**700**	**-**		-	-
										Total Credit	**15**	**05**	**01**	**21**

Elective -I

1) Digital Signal Processing
2) Electronic Measurements
3) Fundamentals of JAVA Programming
4) Computer Networks

Savitribai Phule Pune University Third Year of E & Tc Engineering (2019 Course) 304188: Microcontroller Lab		
Teaching Scheme:	**Credit**	**Examination Scheme:**
Practical: 02 hrs. / week	**01**	**Practical: 50 Marks**
Prerequisite Courses, if any: -		
Companion Course, if any: Microcontroller		

List of Laboratory Experiments

Group A (Any Three)	
1.	Simple programs on Memory transfer.
2.	Parallel port interacting of LEDS—Different programs (flashing, Counter, BCD, HEX, Display of Characteristic)
3.	Interfacing of Multiplexed 7-segment display (counting application)
4.	Waveform Generation using DAC
5.	Interfacing of Stepper motor to 8051- software delay using Timer
Group B (Any Three)	
6.	Write a program for interfacing button, LED, relay & buzzer as follows
7.	Interfacing of LCD to PIC 18FXXXX
8.	Interfacing of 4X4 keypad and displaying key pressed on LCD.
9.	Generate square wave using timer with interrupt
Group C (Any Two)	
11.	Interfacing serial port with PC both side communication.
12.	Interface analog voltage 0-5V to internal ADC and display value on LCD
13.	Generation of PWM signal for DC Motor control.
14.	Interfacing OF RTC using I2C protocol

Virtual LAB Links:

http://vlabs.iitb.ac.in/vlabs-dev/labs/8051-Microcontroller-Lab/labs/index.php

Note: Additional 2 experiments to be performed using the virtual labs.

Dr. D. Y. Patil Institute of Technology, Pimpri, Pune
Department: Electronics & Telecommunication Engineering

Class: THIRD Year **Semester and Year: 5th Sem (2022-23)**
Subject: Microcontroller Lab **Academic Year: 2022-2023**
Teaching Scheme: **Examination Scheme:**
Lectures: 03 hrs/week **In semester exam : 30 Marks**
Practical: 02 hrs/week **End semester exam : 70 Marks**

Lab Planning

Lecture No.	Topic to be covered	Teaching Media / Methodology/ Activity	References	Blooms Taxonomy Level	CO's, PO'S addressed
1	Simple programs on Memory transfer.	Microcontroller kit, Keil µVision 4, Flash Magic	T1, R1, R2	L1	CO1 PO (1, 3, 5, 9, 12)
2	Parallel port interacting of LEDS— Different programs (flashing, Counter, BCD, HEX, Display of Characteristic)		T1, R1, R2	L3	CO1 PO (1, 3, 5, 9, 12)
3	Interfacing of Multiplexed 7-segment display (counting application)		T1, R1, R2	L3	CO1 PO (1, 3, 5, 9, 12)
4	Interfacing of Stepper motor to 8051-software delay using Timer		T1, R1, R2	L3	CO1 PO (1, 3, 5, 9, 12)
5	Write a program for interfacing button, LED, relay & buzzer	Microcontroller kit, MP Lab, PIC Loader	T2, R3, R4	L3	CO 3-5, PO (1, 3, 5, 9, 12)
6	Interfacing of LCD to PIC 18FXXXX		T2, R3, R4	L3	CO 3-5, PO (1, 3, 5, 9, 12)
7	Generate square wave using timer with interrupt		T2, R3, R4	L3	CO 3-5, PO (1, 3, 5, 9, 12)
8	Interface analog voltage 0-5V to internal ADC and display value on LCD	Microcontroller kit, MP Lab, PIC	T2, R3, R4	L3	CO 3-5, PO (1, 3, 5, 9, 12)

9	Generation of PWM signal for DC Motor control.	Loader	T2, R3, R4	L3	CO 3-5, PO (1, 3, 5, 9, 12)
10	Interfacing of 8051 Microcontroller with various display devices.	Virtual Lab (Simulation)	T1, R1, R2	L3	CO1 PO (1, 3, 5, 9, 12)
11	Interfacing of 8051 Microcontroller with DC Motor		T1, R1, R2	L3	CO1 PO (1, 3, 5, 9, 12)

References can be either of this-

Text Books:

1. Mahumad Ali Mazadi, Janice Gillispie Mazadi, Rolin D McKinlay, "The 8051 Microcontroller & Embedded Systems (Using Assembly and C)", PHI, 2nd Edition
2. Mahumad Ali Mazadi, Rolin D McKinlay and Danny Causey, "PIC Microcontroller & Embedded System", Pearson Education, 3rd Edition

Reference Books:

1. Kenneth J. Ayala, 'The 8051 Microcontroller Architecture, Programming and Applications', Cengage Learning, 3rd Edition
2. Ajay Deshmukh, "Microcontrollers Theory and Applications", TATA McGraw Hill, 4th Edition
3. Peatman, John B, "Design with PIC Microcontroller", Pearson Education PTE, 1st Edition
4. Data Sheet of PIC 18Fxxxx series

NPTEL VIDEO/PDF LIST

MOOC / NPTEL Courses:

1. NPTEL Course "Microcontroller and Applications"

Link of the Course: https://nptel.ac.in/courses/117/104/117104072/

https://nptel.ac.in/courses/108/105/108105102/

Pranita Bhosale
(Subject Teacher)

Dr. D. G. Bhalke
(HoD)

Dr. D Y Patil Institute of Technology, Pimpri Pune
Department of Electronics & Telecommunication Engineering

Subject – Microcontroller

Academic Year 2022-23 - Semester 1

VIRTUAL LAB

Objectives:

1. To provide remote-access to simulation-based Labs in various disciplines of Science and Engineering.

2. To enthuse students to conduct experiments by arousing their curiosity. This would help them in learning basic and advanced concepts through remote experimentation.

3. To provide a complete Learning Management System around the Virtual Labs where the students/ teachers can avail the various tools for learning, including additional web-resources, video-lectures, animated demonstrations and self-evaluation.

List of Experiments:

1. Interfacing of 8051 Microcontroller with various display devices.
2. Interfacing of 8051 Microcontroller with DC Motor

Aim

Interfacing of 8051 Microcontroller with various display devices.

About experiment –

- This experiment includes the interfacing of LEDs and Seven segment display with 8051 Microcontroller.
- After completion of this experiment, students will be able to
  1. Interface an 8051 microcontroller with a display device and can perform the desired task.
  2. Program a 8051 microcontroller using assembly language.

This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

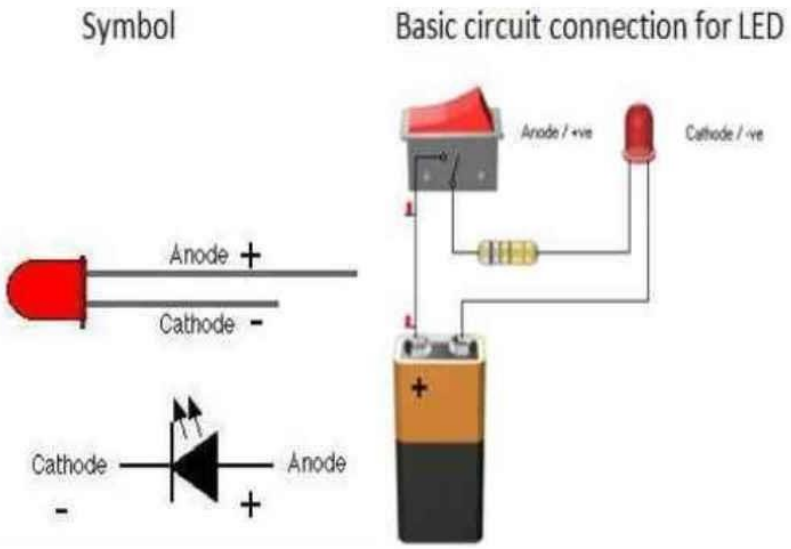
Theory

Task 1:

Blinking LED using 8051 Microcontroller.

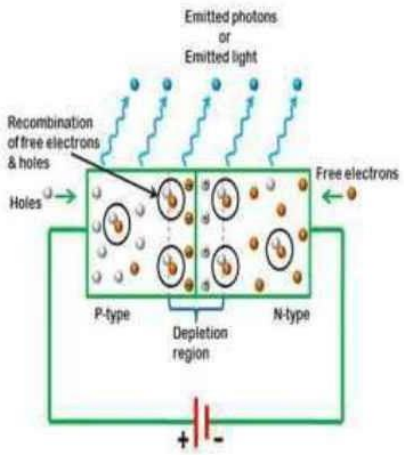
Theory –

A light-emitting diode (LED) is a semiconductor device or simply a PN junction diode that emits visible light when an electric current passes through it in the forward direction. The amount of light output is directly proportional to the forward current till a certain value of current.

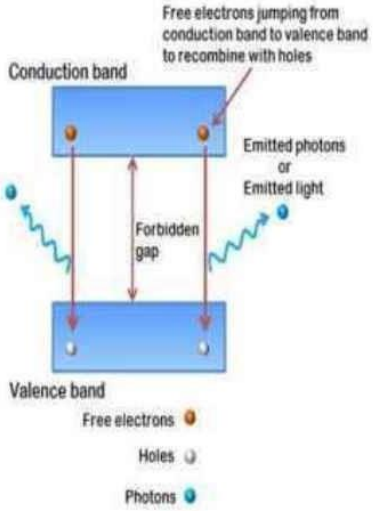


How Light Emitting Diode (LED) works?

Light Emitting Diode (LED) works only in forward bias condition. When Light Emitting Diode (LED) is forward biased, the free electrons from n-side and the holes from p-side are pushed towards the junction. When free electrons reach the junction or depletion region, some of the free electrons recombine with the holes in the positive ions. We know that positive ions have less number of electrons than protons. Therefore, they are ready to accept electrons. Thus, free electrons recombine with holes in the depletion region. In a similar way, holes from p-side recombine with electrons in the depletion region.



Light Emitting Diode (LED)



Process of light emission in LED

Image source : www.physics-and-radio-electronics.com (http://www.physics-and-radio-electronics.com/electronic-devices-and-circuits/semiconductor-diodes/lightemittingdiodeledconstructionworking.html)

Because of the recombination of free electrons and holes in the depletion region, the width of the depletion region decreases. As a result, more charge carriers will cross the p-n junction. Some of the charge carriers from p-side and n-side will cross the p-n junction before they recombine in the depletion region. For example, some free electrons from n-type semiconductor cross the p-n junction and recombine with holes in p-type semiconductor. In a similar way, holes from p-type semiconductor cross the p-n junction and recombine with free electrons in the n-type semiconductor. Thus, recombination takes place in the depletion region as well as in p-type and n-type semiconductor. The free electrons in the conduction band release energy in the form of light before they recombine with holes in the valence band.

How LED will be connected to 8051 practically?

The below diagram shows the interfacing of a LED with a port pin of microcontroller Interfacing_of_led

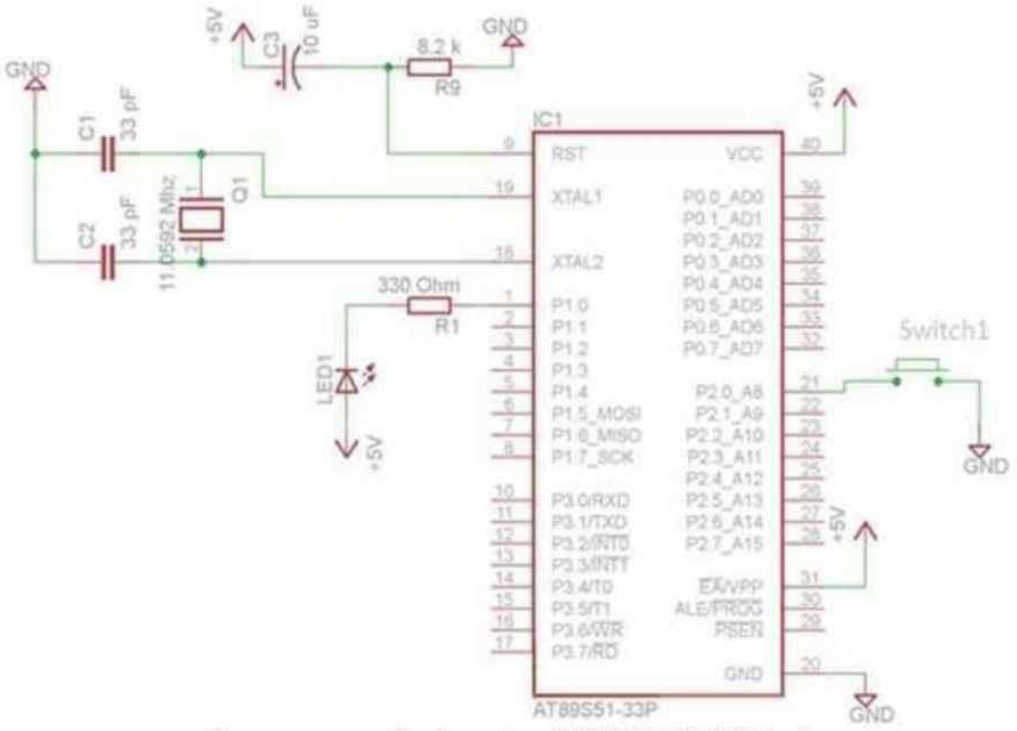


Image source : Designed on CADSOFT EAGLE tool

Task 2:

Display a digit on seven segment display using 8051 microcontrollers.

Theory –

The 7-segment display consists of seven LEDs arranged in a rectangular fashion. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used within the same package which is the indication of a decimal point(DP), when two or more 7-segment displays are connected together numbers greater than ten can be displayed.
So by forward biasing the appropriate pins of the LED segments in a particular order, some segments will be glowing and others will remain as it is, allowing the desired character pattern of the number to be generated on the display. This then allows us to display each of the ten decimal digits 0 to 9 on the same 7-segment display.

Now accordingly terminals are taken common, so there are two types of display:
1. Common Cathode display
2. Common Anode display

1. The Common Cathode (CC) –

In the common cathode display, all the cathode connections of the LED segments are joined together to logic "0" or ground. The individual segments are illuminated by application of a "HIGH", or logic "1" signal via a current limiting resistor to forward bias the individual Anode terminals (a-g).

Common Cathode 7-segment Display

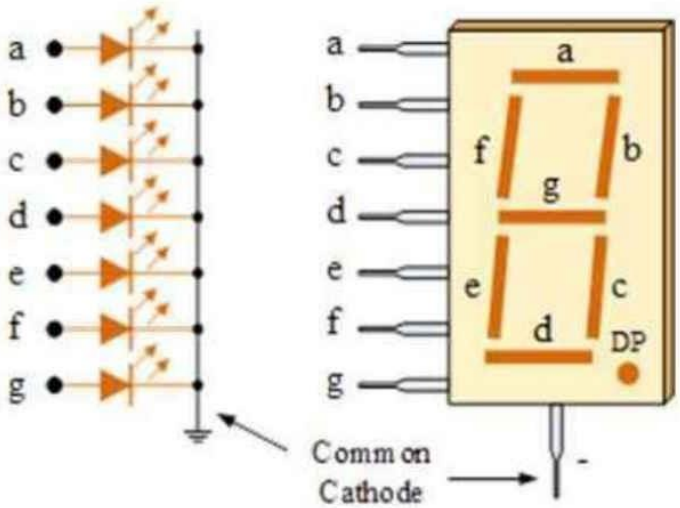


Image source : www.electronics-tutorials.ws (http://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html)

Commom Cathode							
Decimal Digit	Individual Segments Illuminated						
	a	b	c	d	e	f	g
0	1	1	1	1	1	1	
1		1	1				
2	1	1		1	1		1
3	1	1	1	1			1
4		1	1			1	1
5	1		1	1		1	1
6	1		1	1	1	1	1
7	1	1	1				
8	1	1	1	1	1	1	1
9	1	1	1	1		1	1

Common Cathode Decoding Table

2. The Common Anode (CA) –

In the common anode display, all the anode connections of the LED segments are joined together to logic "1". The individual segments are illuminated by applying a ground, logic "0" or "LOW" signal via a current limiting resistor to the Cathode of the particular segment (a-g).

Common Anode 7-segment Display

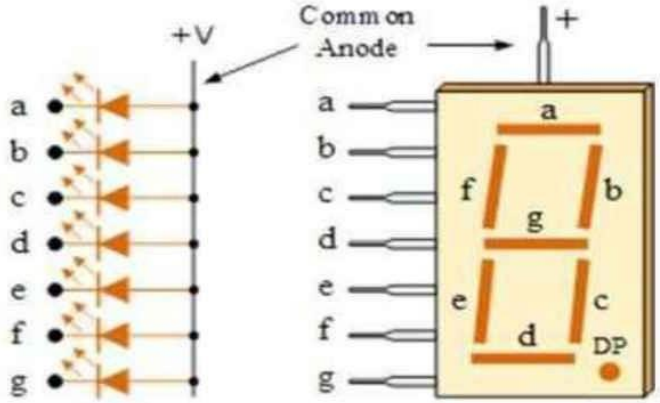


Image source : www.electronics-tutorials.ws (http://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html)

Commom Anode							
Decimal Digit	Individual Segments Illuminated						
	a	b	c	d	e	f	g
0	0	0	0	0	0	0	
1		0	0				
2	0	0		0	0		0
3	0	0	0	0			0
4		0	0			0	0
5	0		0	0		0	0
6	0		0	0	0	0	0
7	0	0	0				
8	0	0	0	0	0	0	0
9	0	0	0	0		0	0

Common Anode Decoding Table

In general, common anode displays are more popular as many logic circuits can sink more current than they can source. Also note that a common cathode display is not a direct replacement in a circuit for a common anode display and vice versa, as it is the same as connecting the LEDs in reverse, and hence light emission will not take place.

Depending upon the decimal digit to be displayed, the particular set of LEDs is forward biased. For instance, to display the numerical digit 0, we will need to light up six of the LED segments corresponding to a, b, c, d, e and f. Then the various digits from 0 through 9 can be displayed using a 7-segment display as shown below.

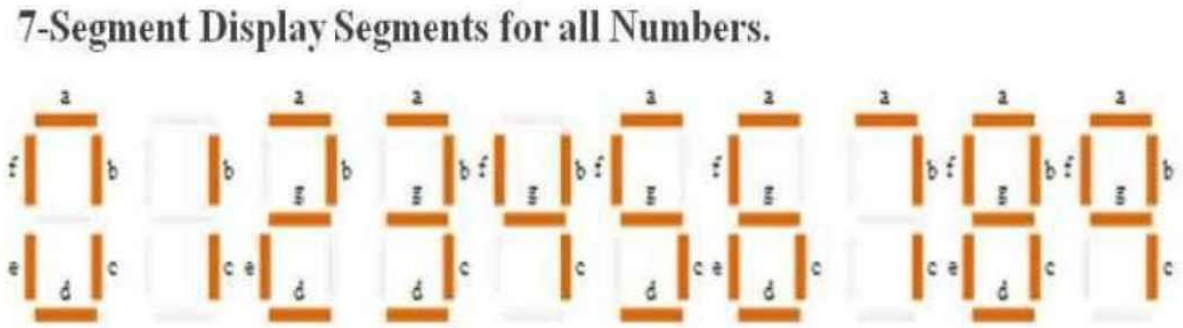


Image source : www.electronics-tutorials.ws (http://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html)

Refer the table given below to know the hex values for displaying a specific number on the 7 segment display during the simulation.

Decimal	ABCDEFG	A	B	C	D	E	F	G
0	0×7E	1	1	1	1	1	1	0
1	0x30	0	1	1	0	0	0	0
2	0x6D	1	1	0	1	1	0	1
3	0x79	1	1	1	1	0	0	1
4	0x33	0	1	1	0	0	1	1
5	0x5B	1	0	1	1	0	1	1
6	0x5F	1	0	1	1	1	1	1
7	0x70	1	1	1	0	0	0	0
8	0x7F	1	1	1	1	1	1	1
9	0x7B	1	1	1	1	0	1	1

Image source : aimagin.com (http://aimagin.com/blog/how-to-drive-a-7-segment-led/)

Documentation about 8051 Microcontroller

- 8051 Overview.pdf (../../src/pdfs-docs/8051 overview.pdf)
- 8051 Hardware Overview.pdf (../../src/pdfs-docs/8051-1.pdf)
- 8051 Instruction Set.pdf (../../src/pdfs-docs/8051IS.pdf)

This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

Pre Test

1) What diode is used in seven-segment displays?

- ○ Zener
- ○ Laser
- ○ Schottky
- ◉ LED

2) Which of the following IC is related to seven segment display?

- ○ IC 7832
- ◉ IC 7447
- ○ IC 7408
- ○ IC 7490

3) Seven segment displays are widely used in digital clocks

- ◉ True
- ○ False

4) The LED is usually made of materials like _____.

- ◉ GaAs
- ○ C and Si
- ○ GeAs
- ○ None of the above

Evaluate

1) Correct

2) Correct

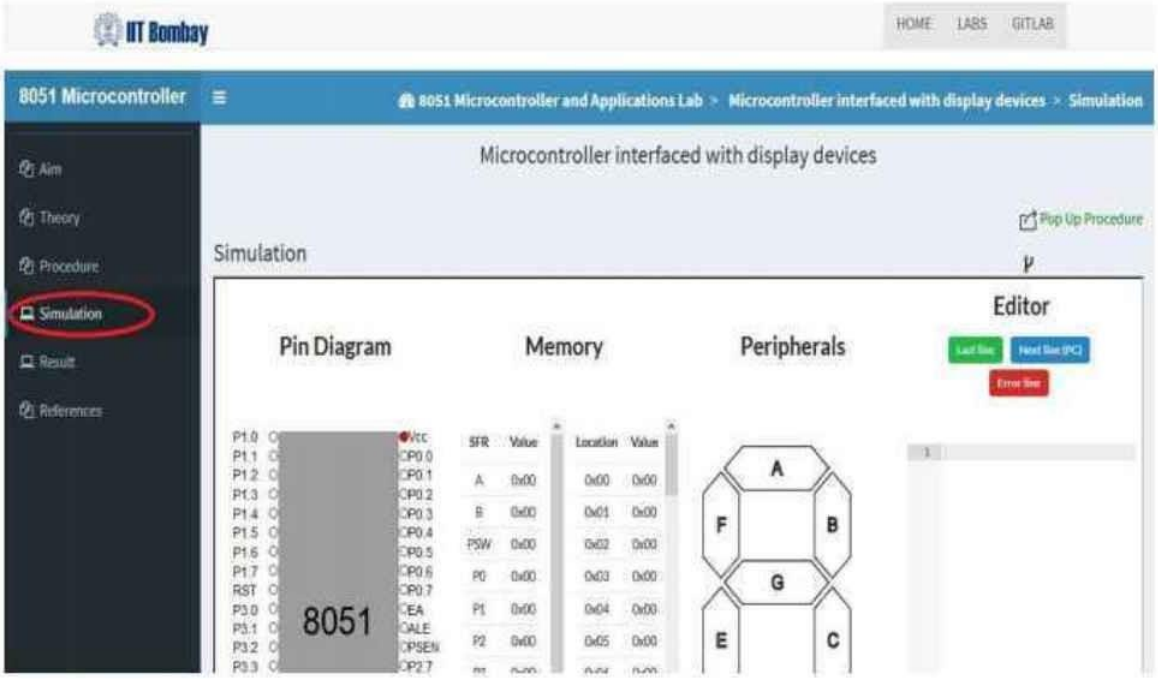
3) Correct

4) Correct

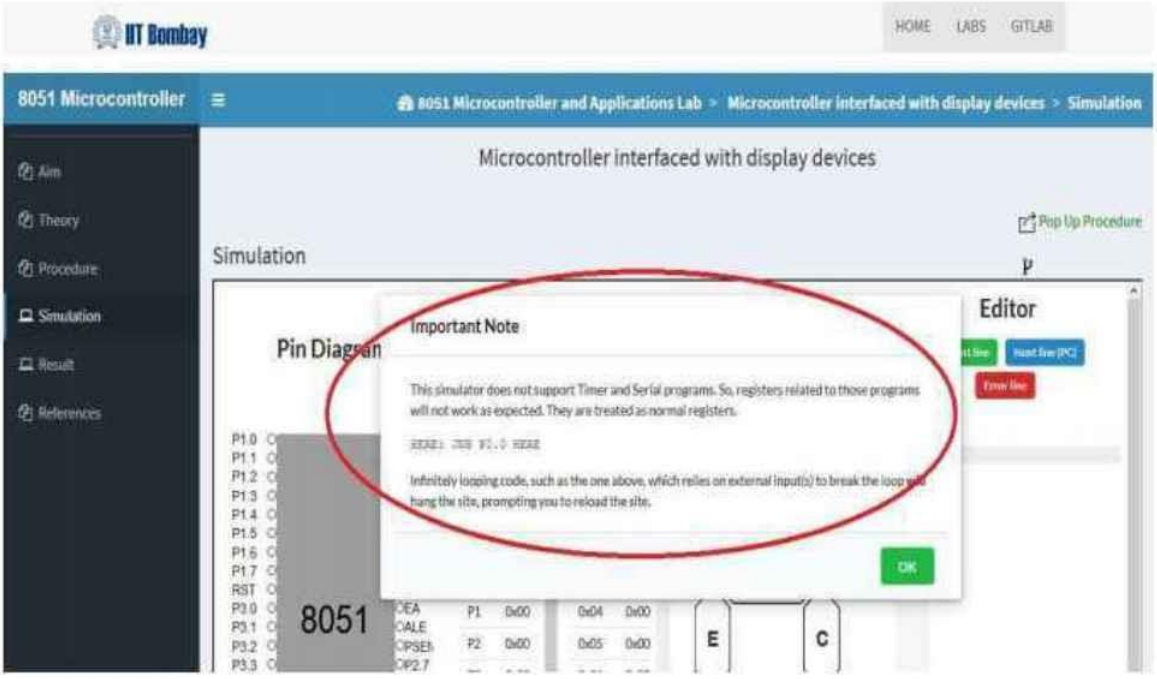
This lab is contributed by (../license/index for

Procedure

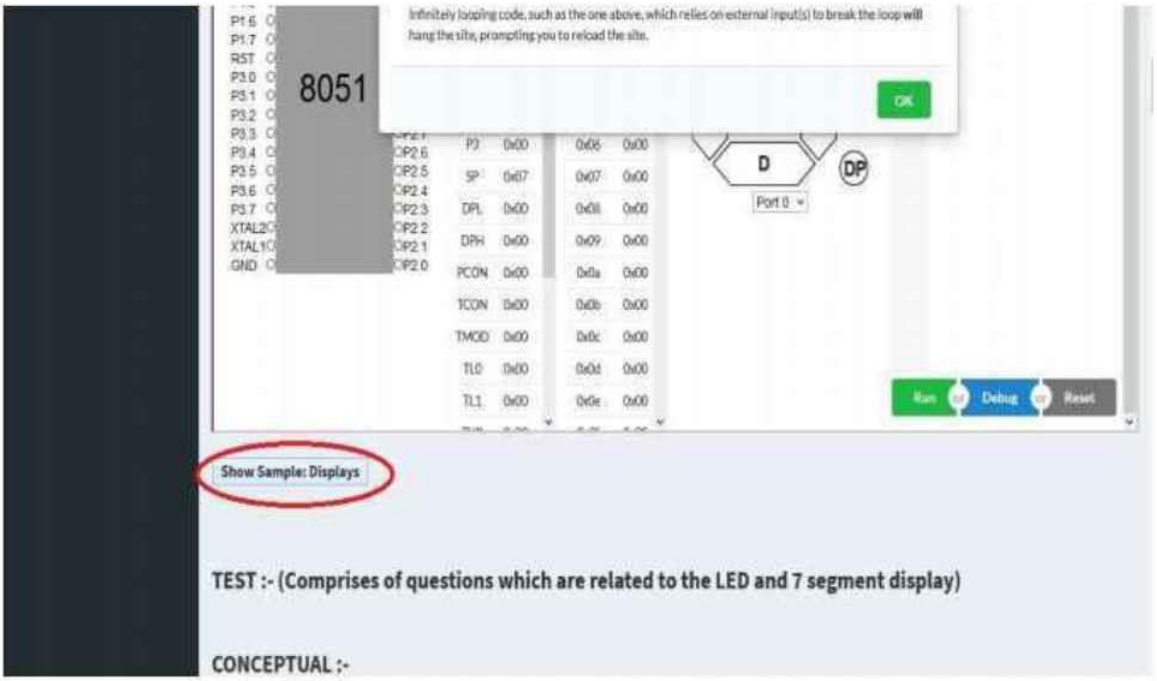
1. Go to the simulator section to perform an experiment.

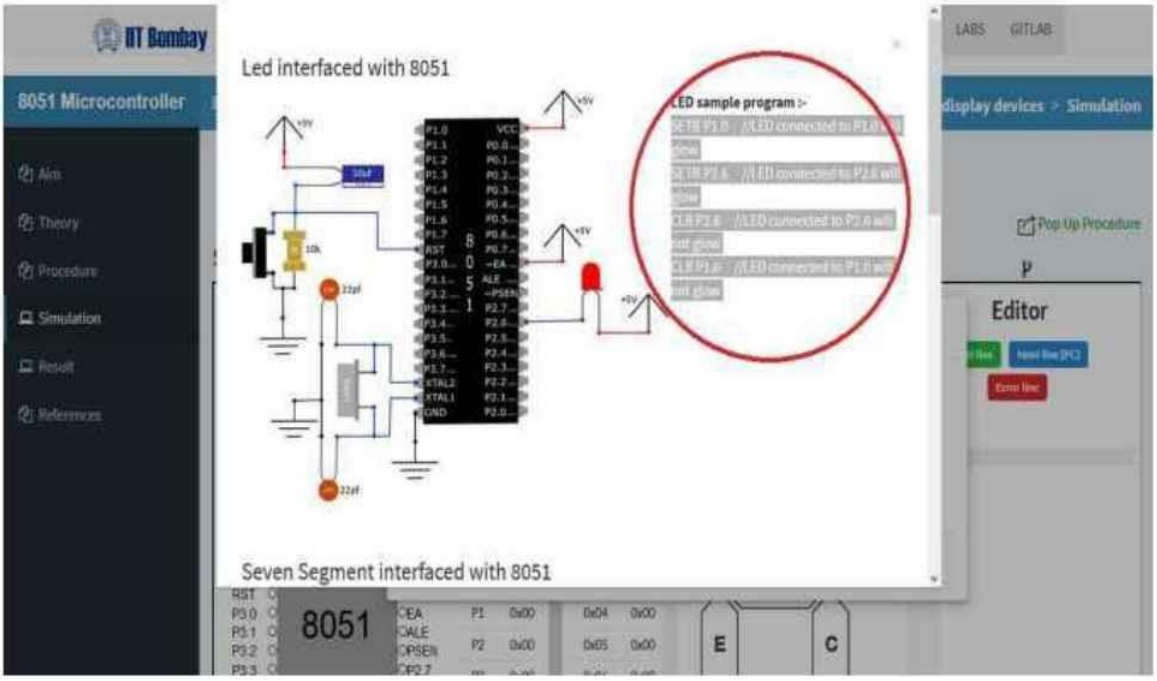


2. Read all the instructions popping up from simulator window carefully.

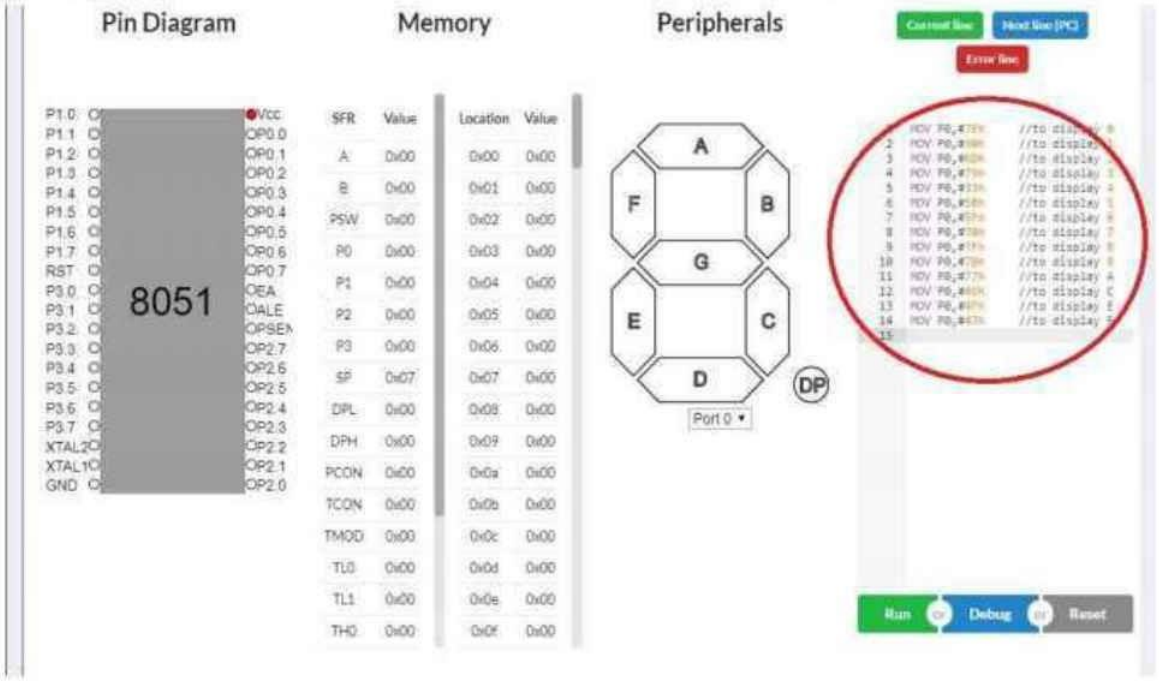


3. Click on show samples button provided below. Understand the interfacing diagram and sample code.



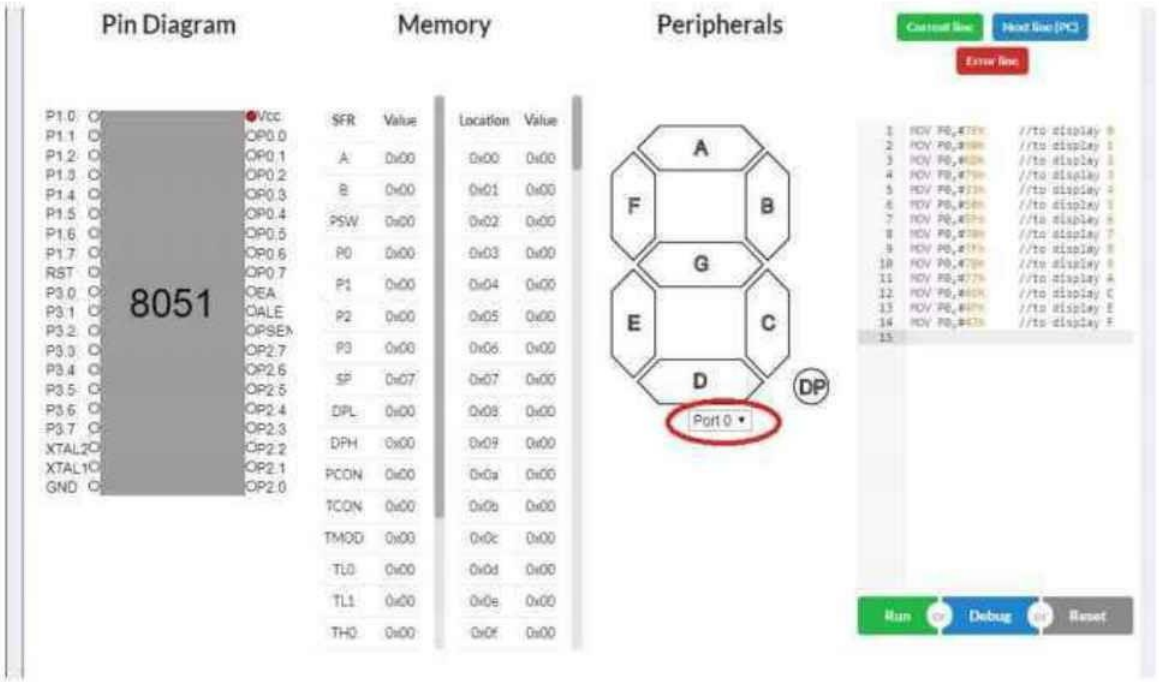


4. Copy program from samples and paste in Text editor in simulator window.

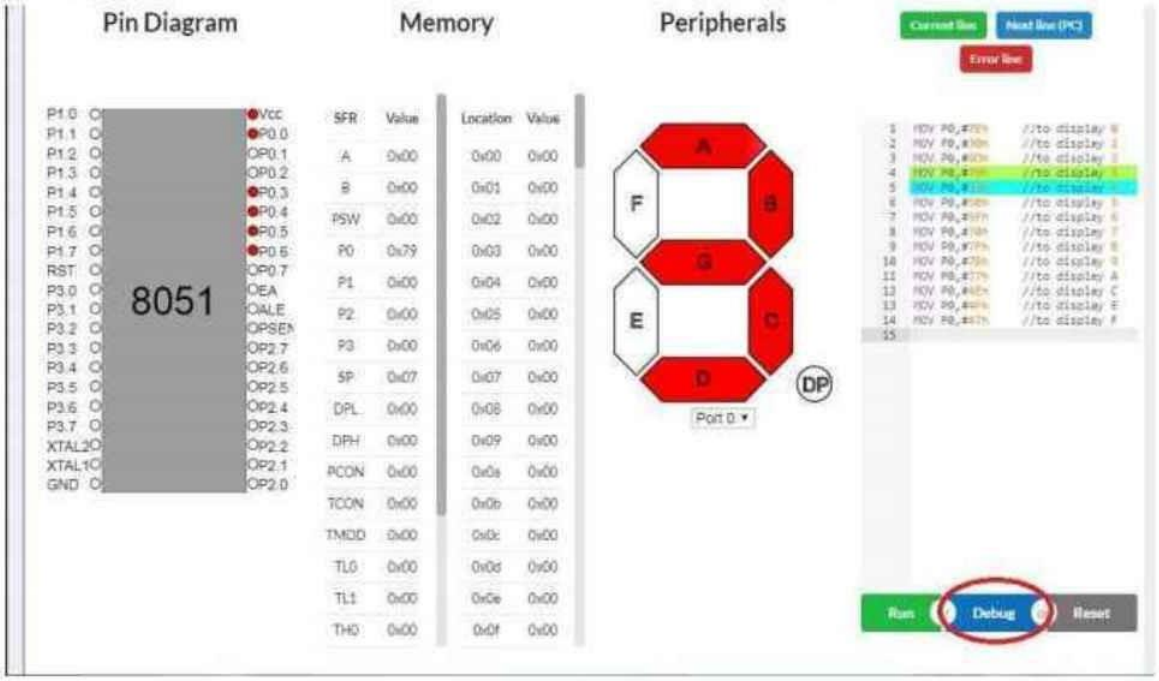


5. Or write your own assembly language code in Text editor.

6. Select appropriate Port according to your code.

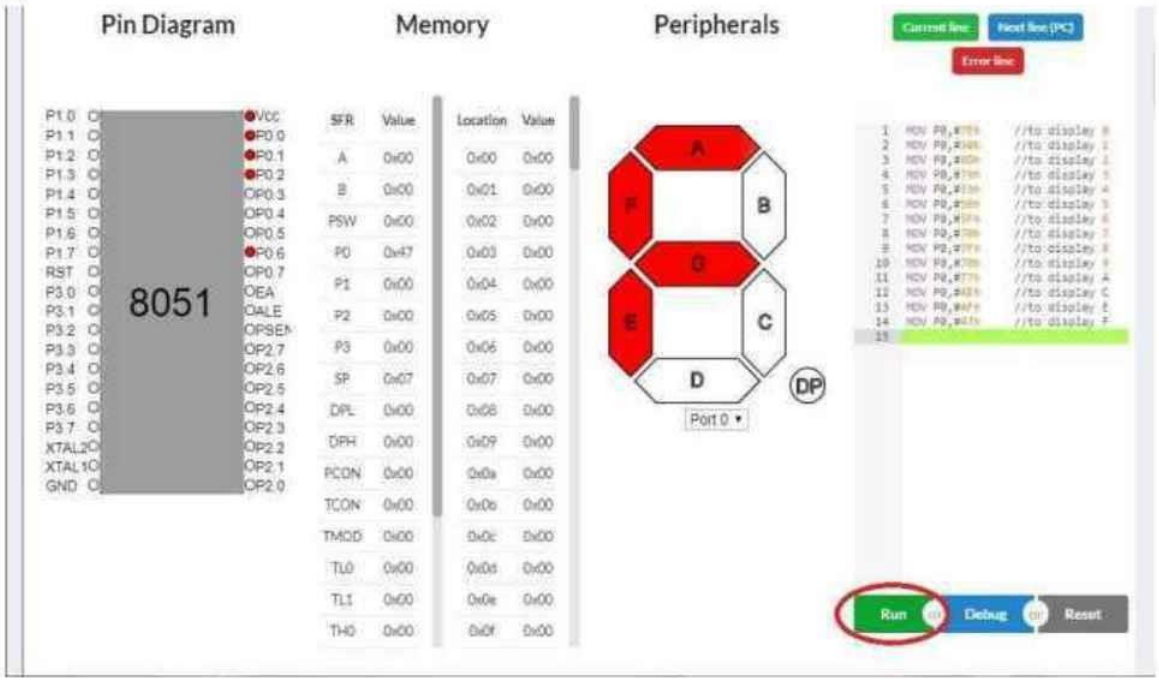


7. To check syntax errors on each line,debug option is provided.



8. If your code output is depending on timing sequence, please use debug funtion.It will show changes in output step by step.

9. To get the final output, run the simulator after debugging and assembling the code with no error.



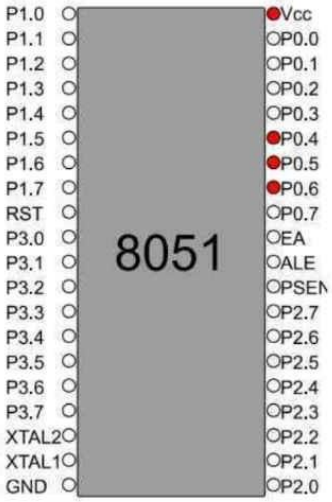
10. Solve test questions given below the simulator section.

11. Submit the answers and a PDF will be generated.(rename it as per your Rollno and Experiment no.)

This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

Simulation

Pin Diagram

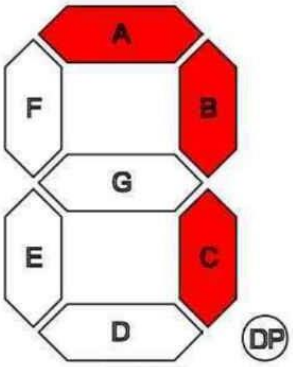


Memory

P2	0x00
P3	0x00
SP	0x07
DPL	0x00
DPH	0x00
PCON	0x00
TCON	0x00
TMOD	0x00
TL0	0x00
TL1	0x00
TH0	0x00
TH1	0x00
SCON	0x00
SBUF	0x00
IE	0x00
IP	0x00

Location	Value
0x00	0x00
0x01	0x00
0x02	0x00
0x03	0x00
0x04	0x00
0x05	0x00
0x06	0x00
0x07	0x00
0x08	0x00
0x09	0x00
0x0a	0x00
0x0b	0x00
0x0c	0x00
0x0d	0x00
0x0e	0x00

Peripherals



Port 0

Editor

Current line Next line (PC)

Error line

```
1  MOV P0,#7Eh     //to disp
2  MOV P0,#30h     //to disp
3  MOV P0,#6Dh     //to disp
4  MOV P0,#79h     //to disp
5  MOV P0,#33h     //to disp
6  MOV P0,#5Bh     //to disp
7  MOV P0,#5Fh     //to disp
8  MOV P0,#70h     //to disp
9  MOV P0,#7Fh     //to disp
10 MOV P0,#7Bh     //to disp
11 MOV P0,#77h     //to disp
12 MOV P0,#7Fh     //to disp
13 MOV P0,#4Eh     //to disp
14 MOV P0,#7Eh     //to disp
15 MOV P0,#4Fh     //to disp
16 MOV P0,#47h     //to disp
17
18
```

Run or Debug or R

Show Sample: Displays

Print Simulation Window

TEST :- (Comprises of questions which are related to the LED and seven segment display)

Instructions:-

1)There are three sections.(conceptual questions, problem solving questions, analytical questions)

2)Click on Tabs to attempt questions in each section.

3)Please attempt all the questions and write the conclusion before submitting.

4)To check and update the progress bar,please click on any of the tab after attempting questions in a section.

5)Please note:- To get 100% progress, attempting all the sections along with the conclusion is necessary.

Progress Bar

0%

Conceptual Problem solving Analytical

Conclusion

Click on submit to generate a PDF of the test.

Submit Answers

This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

Post Test

Solve the following small quiz

Q1. If the junction temperature of LED is increased,the radiant output power _____.

A ○ Increases
B ◉ Decreases
C ○ Remains the same
D ○ None of the above

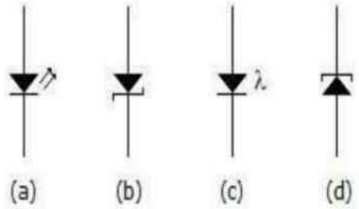
Ans is B

Q2. An eight segment is present in the seven segment display, what is the use?

A ○ Has no use
B ◉ For displaying non-integer numbers
C ○ For displaying integer numbers
D ○ None of the above

Ans is B

Q3. Refer to the figure given below and choose the appropriate symbol for an LED?



A ○ d
B ○ c
C ○ b
D ◉ a

Ans is D

Q4. To display a digit on seven-segment display,we have to apply its _____ at the input.

A ○ 8 bit binary equivalent
B ○ Grey code
C ○ BCD
D ◉ None of the above

Ans is D

Q5. An LED is forward-biased. The diode should be ON, but no light is visible. What can be the possible reason for it?

A ○ The applied voltage is too small
B ○ The series resistor is too small
C ○ The applied voltage is too high
D ◉ None of the above

Ans is A

Q6. Is the seven segment display visible in the simulator section implemented using the common anode display concept

A ◉ No
B ○ yes
C ○ Not sure

Ans is A

Q7. LED is used in fancy electronic devices such as toys emitting

A ○ X-rays
B ○ Ultraviolet light

C ○ Visible light
D ◉ Radio waves

Ans is D

Q8. LED is used in fancy electronic devices such as toys emitting _____

A ○ X-rays
B ○ Ultraviolet light
C ◉ Visible light
D ○ Radio waves

Ans is C

Print The Page

This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

Aim

Interfacing of 8051 Microcontroller with DC motor.

About experiment –

This experiment includes the interfacing of DC motor with 8051 microcontroller. After completion of this experiment, students will be able to

- Interface a 8051 microcontroller with a DC motor.
- Program a 8051 microcontroller using assembly language.

This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

Theory

Task 1: DC Motor interfacing with 8051 microcontroller

Theory:-

Image source : www.daenotes.com (http://www.daenotes.com/electronics/digital-electronics/digital-to-analog-converters)

A DC motor is an electrical mechanism that converts direct current electrical power into mechanical power.

The principle of working of a DC motor is that "whenever a current carrying conductor is placed in a magnetic field, it experiences a mechanical force". The direction of this force is given by Fleming's left-hand rule.

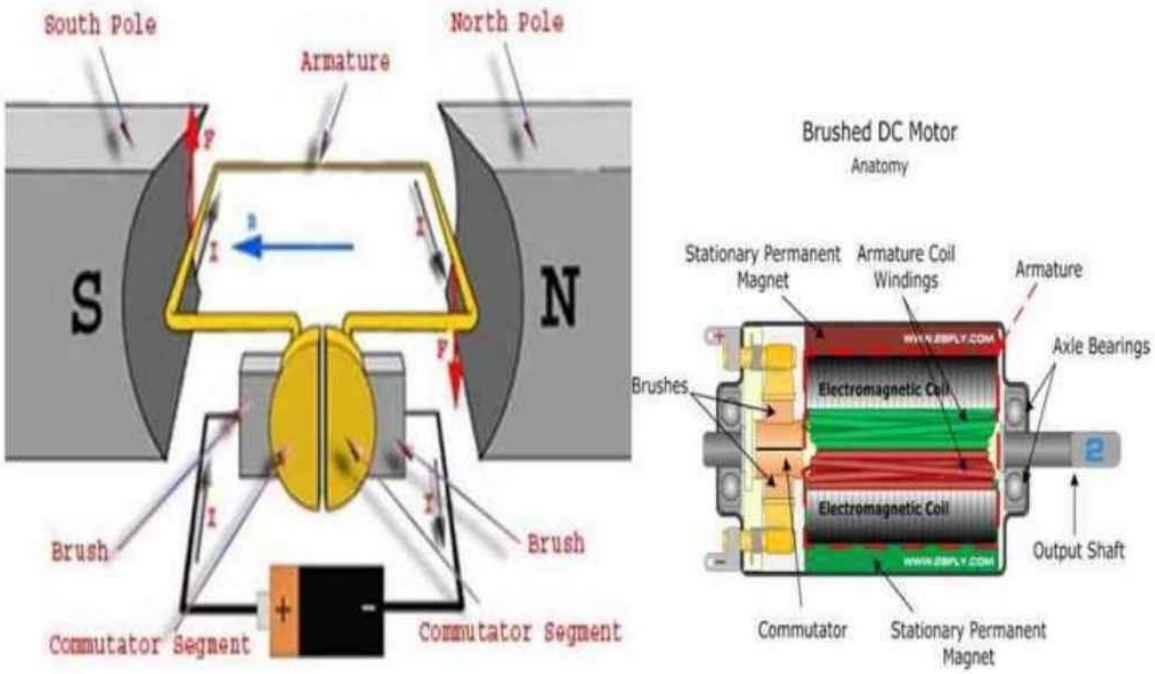
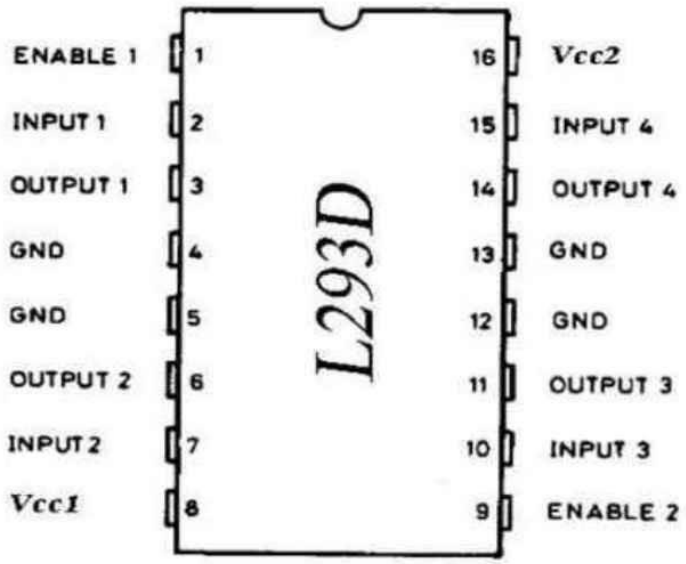


Image source : www.motors-biz.com (http://www.motors-biz.com/news/newsDetail/466.html)

Hence by just reversing the voltage levels at the two pins of DC motor, we can reverse the direction of rotation of the shaft.
If two pins are at same levels i.e. 0v-0v or 12v-12v, the shaft will not move.

The need for the motor driver :

In practice 8051 microcontroller output pin sources low current not sufficient to drive the dc motor, also isolation is required in the interfacing of inductive devices. Hence, we normally use driver ICs such as ULN2003, L293D, transistors.



FUNCTION TABLE
(each driver)

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

H = high level, L = low level,
X = irrelevant,
Z = high impedance (off)

Image source : www.motors-biz.com (http://www.motors-biz.com/news/newsDetail/466.html)

Motor drivers are used preferably to

1. Increases voltage and current sourcing capability.
2. Provides isolation to 8051 from inductive effects of motors.
3. Motor drivers have inbuilt H bridges so as to reverse the direction of rotation, no rewiring is required.
4. En pin is provided for PWM speed control.

Interfacing diagram of DC motor with 8051 Microcontroller

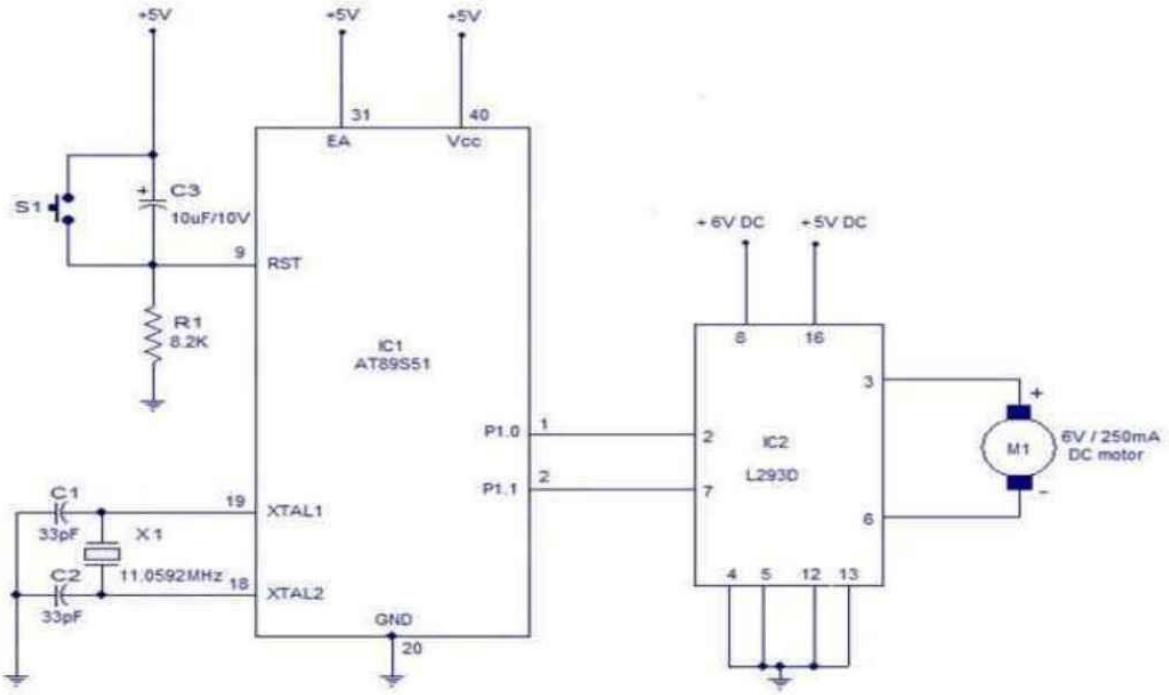


Image source : www.circuitstoday.com (http://www.circuitstoday.com/interfacing-dc-motor-to-8051)

Documentation about 8051 Microcontroller

- 8051 Overview.pdf (..\..\src\pdfs-docs\8051 overview.pdf)
- 8051 Hardware Overview.pdf (..\..\src\pdfs-docs\8051-1.pdf)
- 8051 Instruction Set.pdf (..\..\src\pdfs-docs\8051IS.pdf)

This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

Pre Test

1) The 8051 is the original member of the MCS-51 family

- (●) True
- ( ) False

2) The original 8051 was available in _____ versions.

- ( ) One
- ( ) Two
- (●) Three
- ( ) Four

3) The original 8051 had ____ bytes of on-chip data RAM.

- ( ) 28
- (●) 128
- ( ) 256
- ( ) 312

4) The original 8051 is a _____.

- ( ) 12 bit microcontroller
- (●) 8 bit microcontroller
- ( ) 20 bit microcontroller
- ( ) 5 bit microcontroller

Evaluate

1) Correct

2) Correct

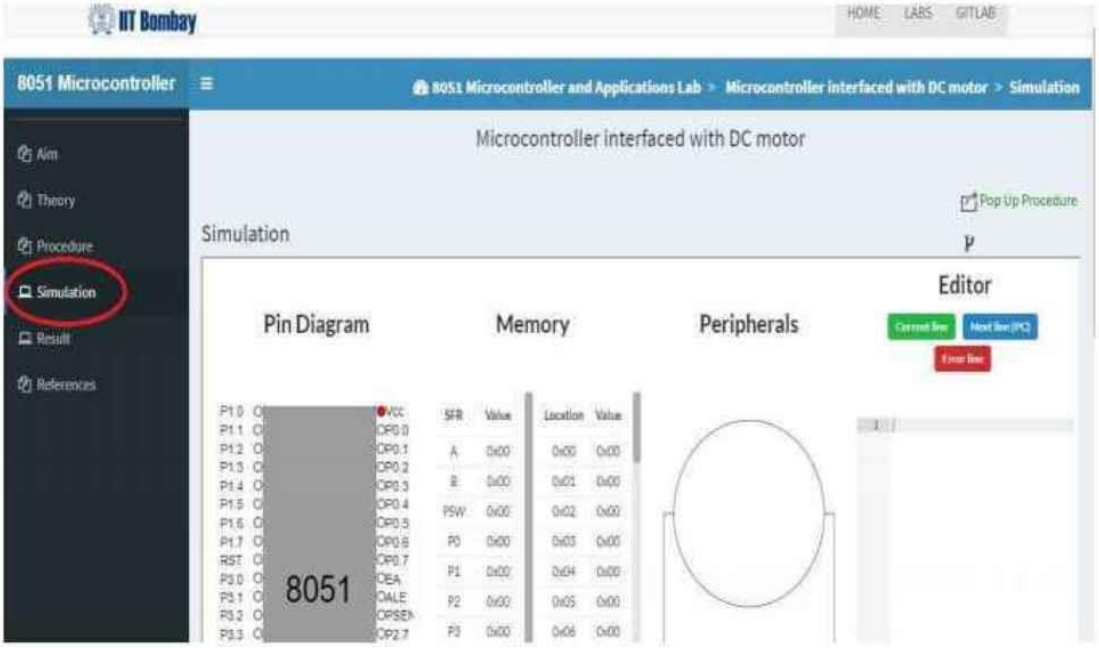
3) Correct

4) Correct

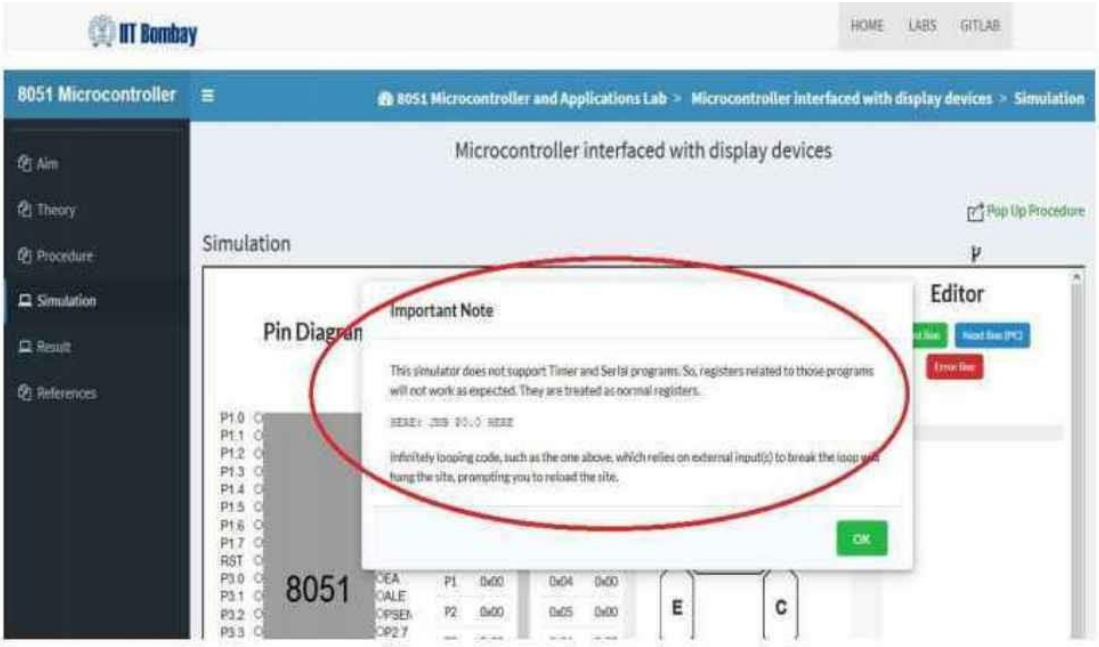
This lab is contributed by (../license/index for

Procedure

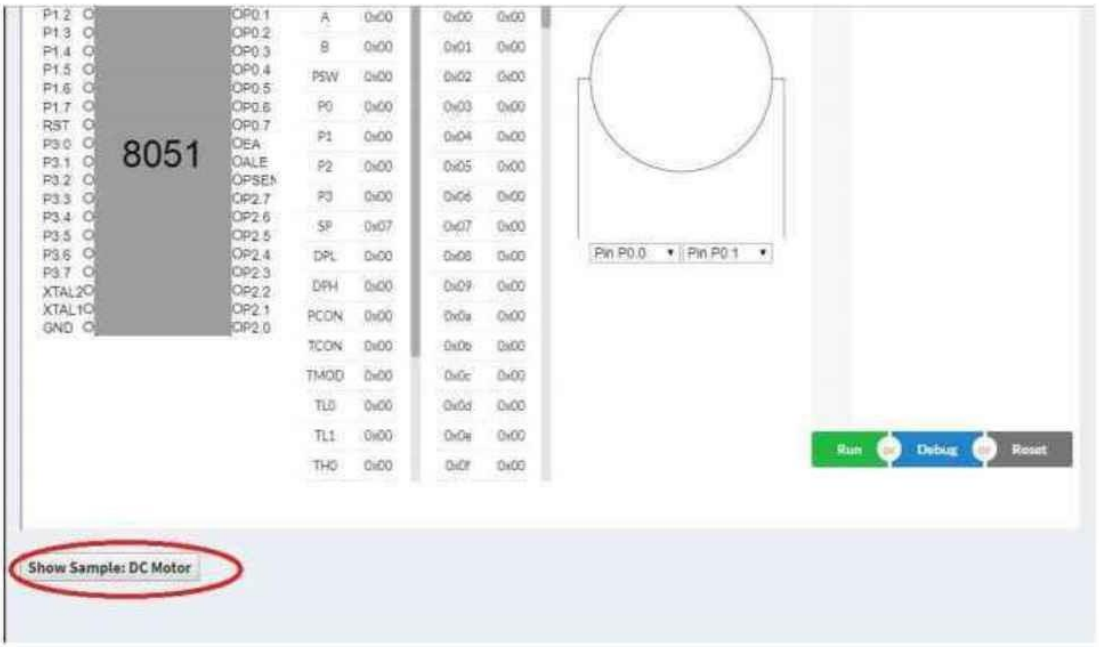
1. Go to the simulator section to perform an experiment.

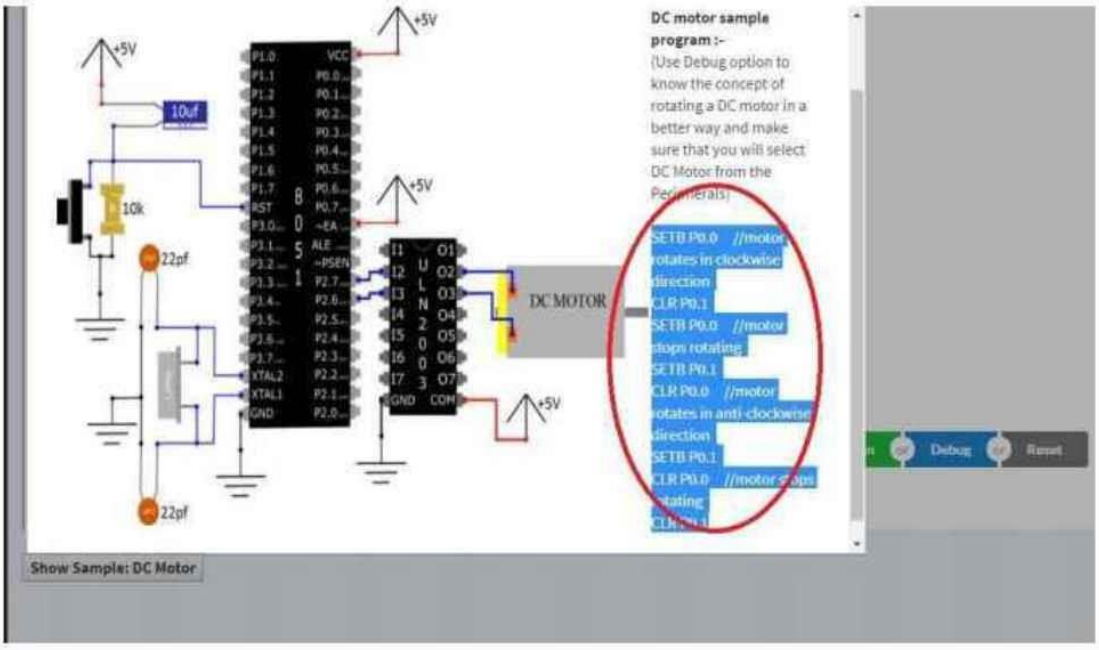


2. Read all the instructions popping up from the simulator window carefully.

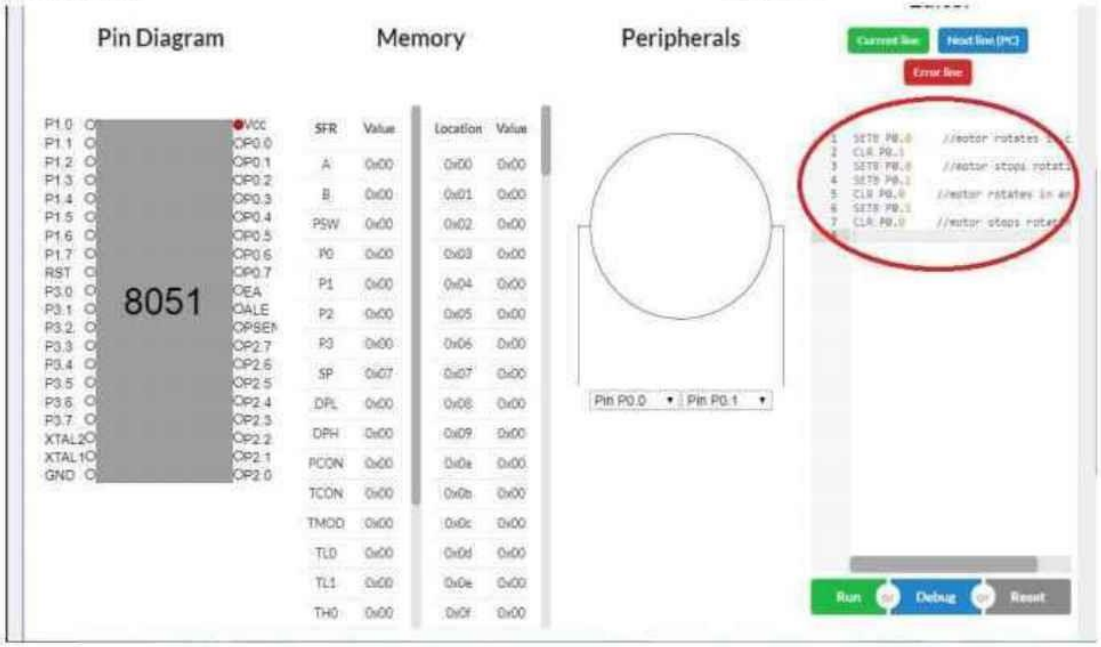


3. Click on show samples button provided below. Understand the interfacing diagram and sample code.



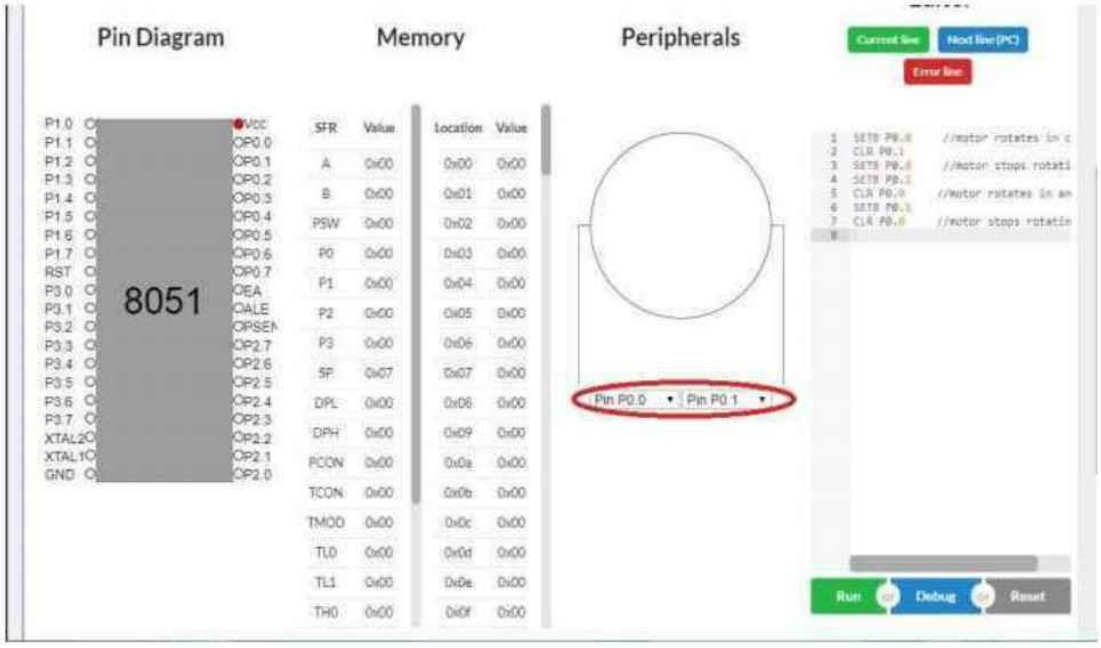


4. Copy program from samples and paste in the Text editor in simulator window.

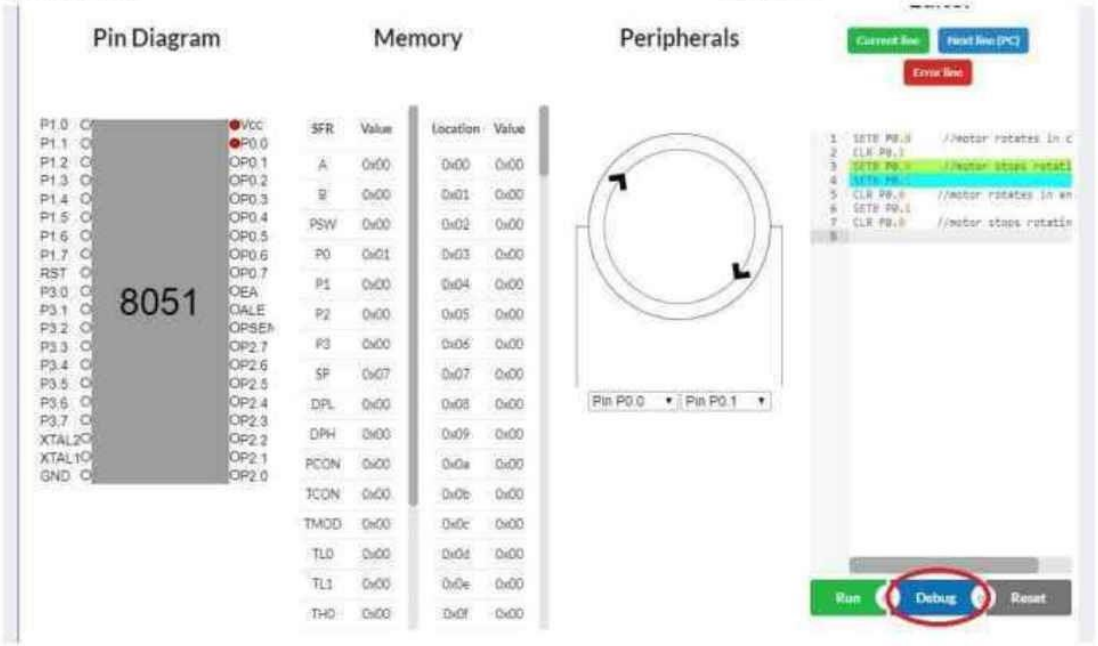


5. Or write your own assembly language code in the Text editor.

6. Select appropriate Port pins according to your code.

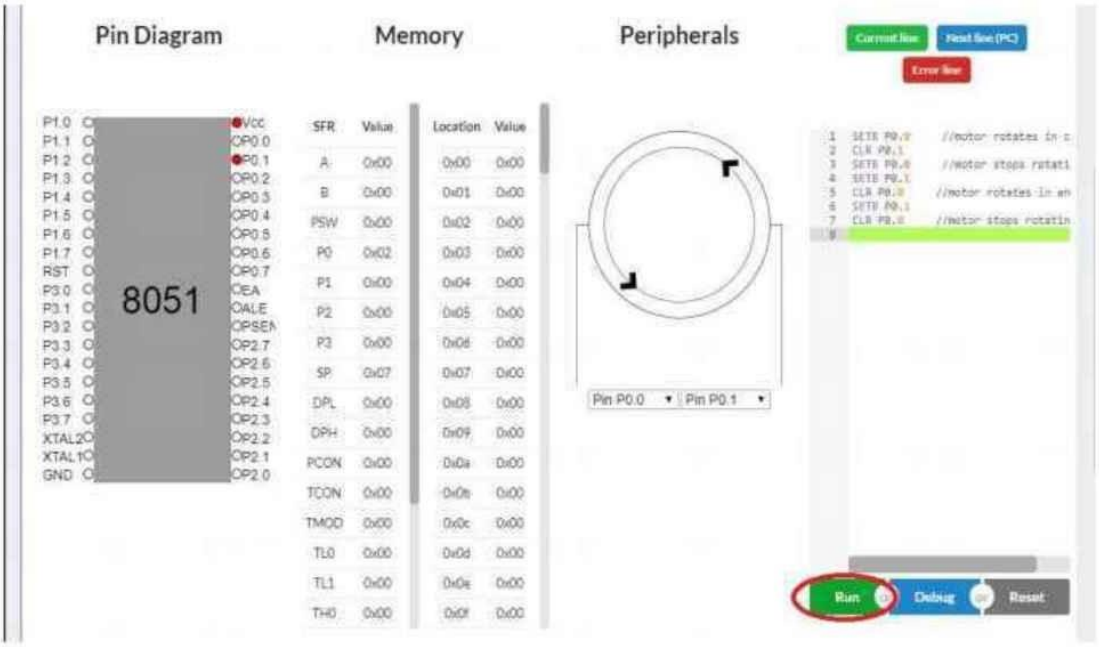


7. To check syntax errors on each line, debug option is provided.



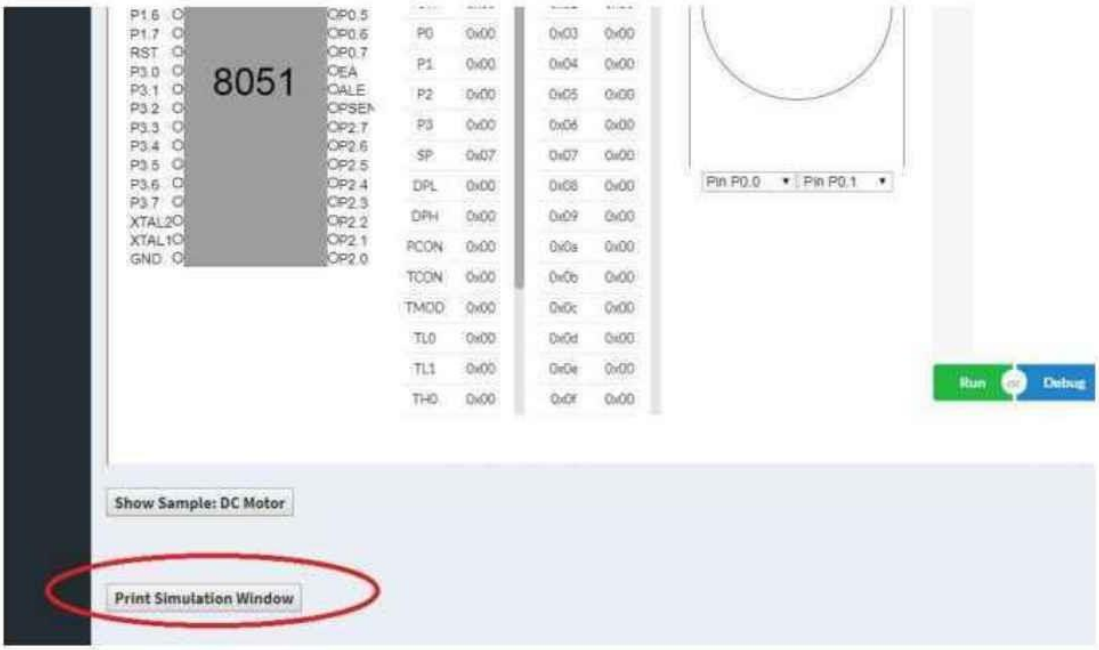
8. If your code output is depending on timing sequence, use debug function. It will show changes in output step by step.

9. To get the final output, run the simulator after debugging and assembling the code with no error.



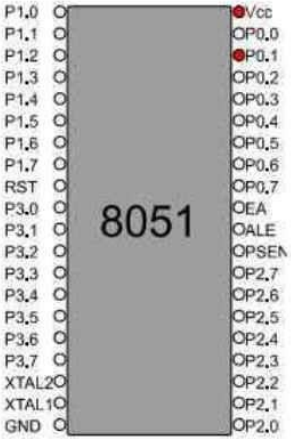
10. Solve test questions given below the simulator section (if any).

11. Take a printout of the simulator window. (button is provided at the bottom)



This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

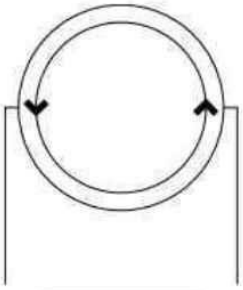
Pin Diagram



Memory

SFR	Val	Location	V
A	0x	0x00	C
B	0x	0x01	C
PSW	0x	0x02	C
P0	0x	0x03	C
P1	0x	0x04	C
P2	0x	0x05	C
P3	0x	0x06	C
SP	0x	0x07	C
DPL	0x	0x08	C
DPH	0x	0x09	C
PCON	0x	0x0a	C
TCON	0x	0x0b	C
TMOD	0x	0x0c	C
TL0	0x	0x0d	C
TL1	0x	0x0e	C

Peripherals



Pin P0.0
Pin P0.1

Editor

Current line

Next line (PC)

Error line

```
1  SETB P0.0     //motor r
2  CLR P0.1
3  SETB P0.0     //motor s
4  SETB P0.1
5  CLR P0.0      //motor rc
6  SETB P0.1
7  CLR P0.0      //motor st
8  CLR P0.1
```

Run or Debug or Rese

Pre Test

1) What is the advantage of interfacing DC motor with 8051 microcontrollers?

○Controlling the direction of the motor
○Controlling the speed of the motor
◉Options A and B
○None of the above

2) A motor driver circuit is used as an interface between the DC motor directly to the controller.

◉True
○False

3) Which of the following statements are true for the usefulness of motor drivers?

○Increases voltage and current sourcing capability.
○Provides isolation to 8051 from inductive effects of motors.
○Motor drivers have inbuilt H bridges so as to reverse the direction of rotation, no rewiring is required.
◉All of the above

4) The principle of working of a DC motor is that _____.

◉whenever a current carrying conductor is placed in a magnetic field, it experiences a mechanical force
○whenever a conductor is placed in a magnetic field, it experiences a mechanical force
○whenever a current carrying conductor is placed in an electric field, it experiences a mechanical force
○whenever a current carrying conductor is placed in a magnetic field, it experiences a gravitational force

Evaluate

1) Correct

2) Correct

3) Correct

4) Correct

This lab is contributed by (../license/index for 8051 Microcontroller and Applications Lab.html)

Dr. D Y Patil Institute of Technology, Pimpri Pune
Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 1

TITLE: Simple programs on Memory transfer

AIM/ OBJECTIVE: Embedded C Programs on Memory Transfer

HARDWARE/SOFTWARE USED: Keil uvision3

Theory:

Brief Introduction of Microcontroller:

Microcontrollers are essentially microprocessors (CPU) with on chip program and data memory along with ports and data memory along with ports, timers, serial communication features, etc., all in a single –packaged IC. This integration makes the microcontrollers power efficient, compact and more reliable. MCS-48 was the first microcontroller designed by Intel, which was extensively, used in IBM PC keyboards. MCS- 51 was the next microcontroller designed by Intel and made available by 1980.

The MCS-51 family of microcontrollers was designed around Harvard architecture, which treats program memory and data memory separately. Presently, many manufacturers offer their own unique microcontrollers of 8-, 16- or 32 bit capabilities. These are available in various types of packages, like TQFP or dual-in line Package (DIP), etc. Most microcontrollers offer some protection against unwanted software piracy. Power management also forms an important feature of microcontrollers, as; in general, these are powered from battery sources.

Numbering Convention of MCS-51 Microcontrollers

8	X	Y	Z

X

0	PROM
7	EPROM
9	FLASH

Y

3	No on-chip program memory
5	On chip program memory

Z

	Program Memory	Data memory	Timer/counter
1	4k	128 bytes	2
2	8k	256 bytes	3
4	16k	256 bytes	5

8051 → X=0 (PROM), Y=5(on chip program memory) and Z=1 (Program memory =4K, data memory=128 bytes and timer/counter =2)

8751→ X=7 (EPROM), Y=5(on chip program memory) and Z=1 (Program memory =4K, data memory=128 bytes and timer/counter =2)

AT89C51-12PC → AT for Atmel, 9 before C indicates FLASH type ROM; C before the 51 stands for CMOS, which has a low power consumption, ; 5 indicate on-chip program memory, 1 indicates program memory =4K, data memory=128 bytes and timer/counter =2 ; "12" indicates 12 MHz ; "P" is for plastic DIP package ; "C" is for commercial.

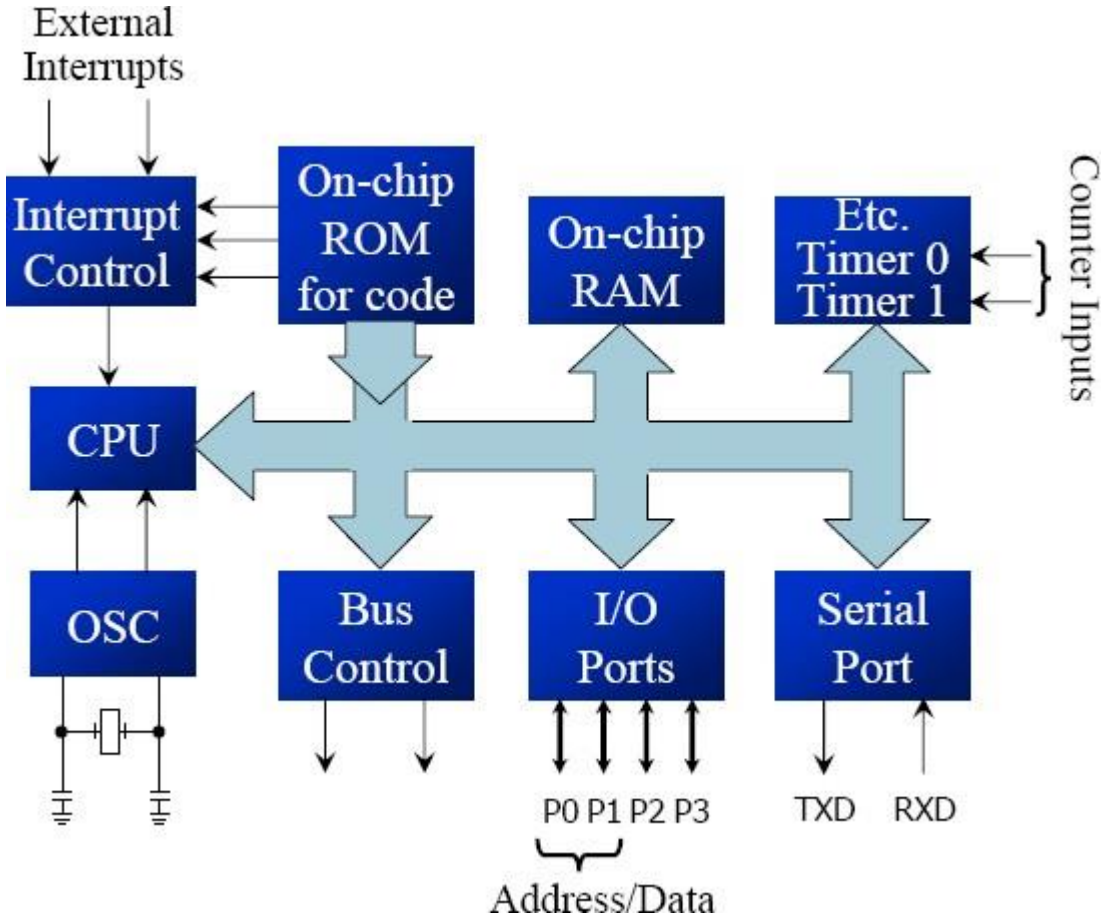
Microcontroller

The fixed amount of on-chip ROM, RAM,and number of I/O ports makes them ideal for many applications in which cost and space are critical In many applications, the space it takes, the power it consumes, and the price perunit are much more critical considerations than the computing power.

- **8-bit microcontrollers**
  - Motorola's 6811
  - Intel's 8051
  - Zilog's Z8
  - Microchip's PIC

- **Intel introduced 8051, referred as MCS-51, in 1981**
  - The 8051 is an 8-bit processor
  - The CPU can work on only 8 bits of data at atime
  - The 8051 had 128 bytes of RAM
  - 4K bytes of on-chip ROM
  - Two timers
  - One serial port
  - Four I/O ports, each 8 bits wide
  - 6 interrupt sources

The 8051 became widely popular after allowing other manufactures to makeand market any flavor of the 8051, but remaining code-compatible.

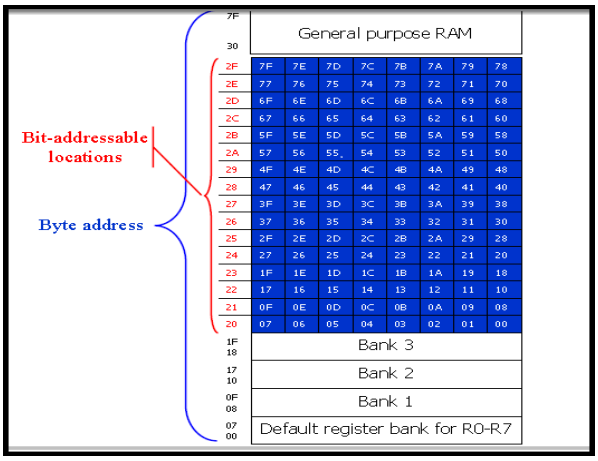


- **Feature 8051 8052 8031:-**

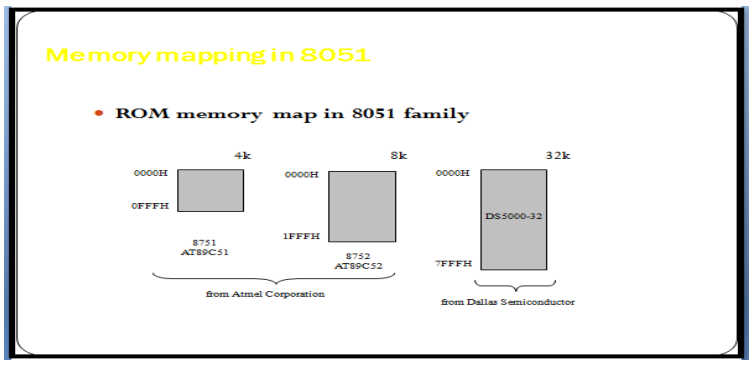
	8051	8052	8031
Interrupt sources	6	8	6
Serial port	1	1	1
I/O pins	32	32	32
Timers	2	3	2
RAM (bytes)	128	256	128
ROM (on-chip program space in bytes)	4K	8K	0K

- **Registers in Microcontroller:**

The most widely used registers A (Accumulator) For all arithmetic and logic instructions B, R0, R1, R2, R3, R4, R5, R6, R7 DPTR (data pointer), and PC (program counter)



(**Internal RAM of 8051**)



***C program to add two 8 bit numbers**

```
#include<reg51.h>
void main(void)
{
  unsigned char x,y,z;
  x=0x0A;
  y=0x05;
  P1=0x00;
  z=x+y;
  P1=z;
}
```

***C program to subtract two 8 bit numbers**

```
#include<reg51.h>
void main(void)
{
  unsigned char x,y,z;
  x=0x08;
  y=0x04;
  P1=0x00;
  z=x-y;
  P1=z;
}
```

***C program to multiply two 8 bit numbers**

```
#include<reg51.h>
void main(void)
{
  unsigned char x,y,z;
  x=0x10;
  y=0x02;
  P1=0x00;
  z=x*y;
  P1=z;
}
```

***C program to divide two 8 bit numbers**

```
#include<reg51.h>
void main(void)
{
  unsigned char x,y,z;
  x=0x10;   y=0x02;
  P1=0x00;
  z=x/y;
  P1=z; }
```

***C program to data transfer**

```
#include <reg51.h>
void main(void)
{
    unsigned char mynum[]= "012345ABCD";
    unsigned char z;
    for(z=0; z<=9; z++)
    P1= mynum[z];
}
```

CONCLUSION:

DPU

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 2

TITLE: Parallel port interacting of LEDS

AIM/ OBJECTIVE: Parallel port interacting of LEDS—Different programs (flashing, Counter, BCD, HEX, Display of Characteristic) using 8051 Microcontroller

HARDWARE USED: Microcontroller kit, Power supply, connecting wires.

SOFTWARE USED: Keil µVision 4, Flash Magic.

Theory:

LEDs are by far the most widely used means of taking output. They find huge application as indicators during experimentations to check the validity of results at different stages. They are very cheap and easily available in a variety of shape, size and colours. LEDs are connected to the port of the microcontroller. LEDs need approximately 10mA current to flow through them in order to glow at maximum intensity. However the output of the controller is not sufficient enough to drive the LEDs, so if the positive leg of the LED is connected to the pin and the negative to ground as shown in the figure, the LED will not glow at full illumination. To overcome this problem LEDs are connected in the reverse order and they run on negative logic i.e., whenever 1 is given on any pin of the port, the LED will switch off and when logic 0 is provided the LED will glow at full intensity. The circuit of LED interfacing with microcontroller is as shown below:

Program:

a) The program for blinking of 8 LED's interfaced with port P1 of 8051.

```
#include <reg51.h>
void Delay(unsigned int);
void main(void)
{
 while(1)  //INFINITE LOOP
```

```
 {
   P1=0x0FF;
  Delay(250);
   P1=0x00;
Delay(250);
 }
}
void Delay(unsigned int xtime)
{
 unsigned int i;
 unsigned char j;
        for(i=0;i<xtime;i++)  for(j=0;j<165;j++);
}
```

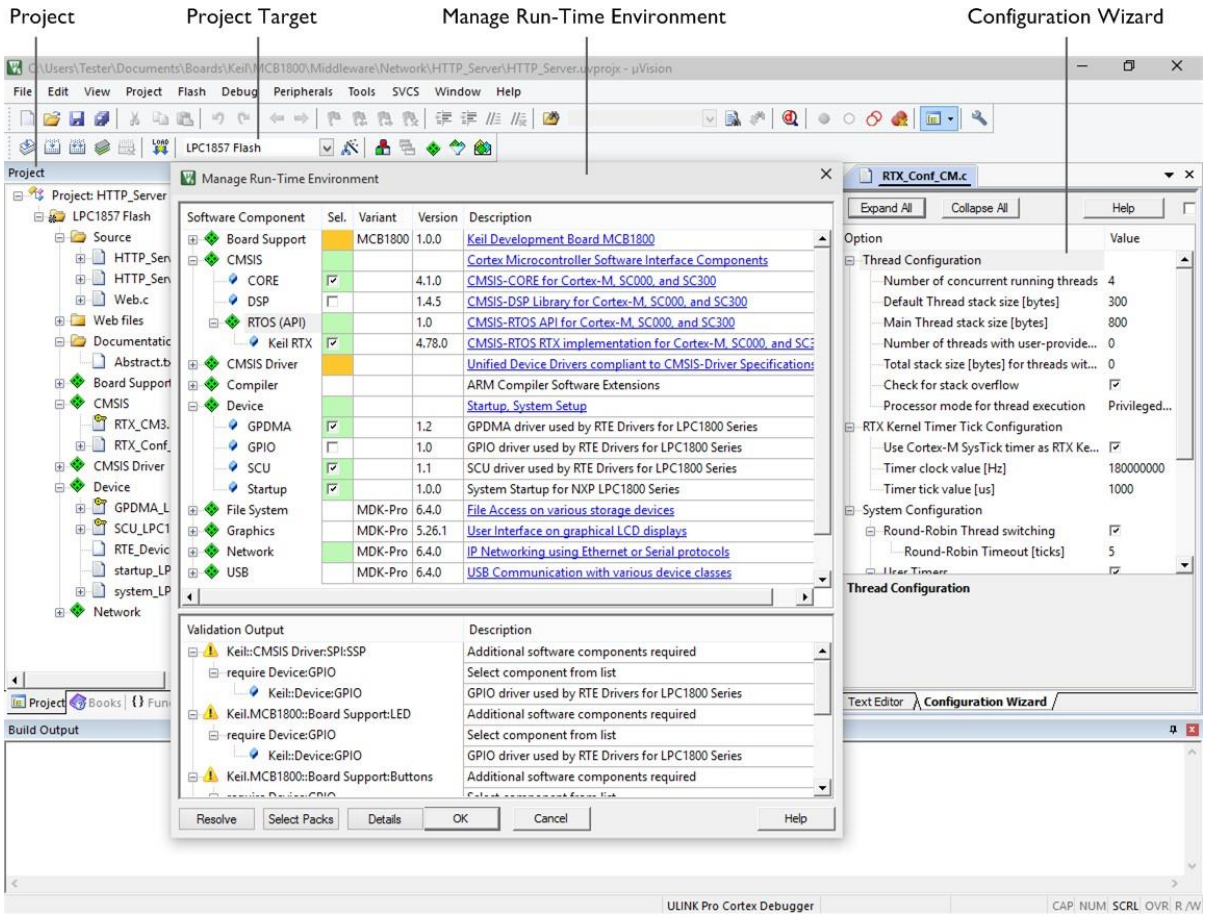
b) The program for toggling of 8 LED's interfaced with port P1 of 8051.

```
#include <reg51.h>
void Delay(unsigned int);
void main(void)
{
 while(1)  //INFINITE LOOP
 {
P1=0x0AA;
 Delay(250);
P1=0x00;
Delay(250);
 }
}
void Delay(unsigned int xtime)
{
        unsigned int i;
 unsigned char j;
        for(i=0;i<xtime;i++)
        for(j=0;j<165;j++);
}
```

WORKING OF SOFTWARE:

1. Keil µVision:

The µVision IDE combines project management, run-time environment, build facilities, source code editing, and program debugging in a single powerful environment. µVision is easy-to-use and accelerates your embedded software development. µVision supports multiple screens and allows you to create individual window layouts anywhere on the visual surface.



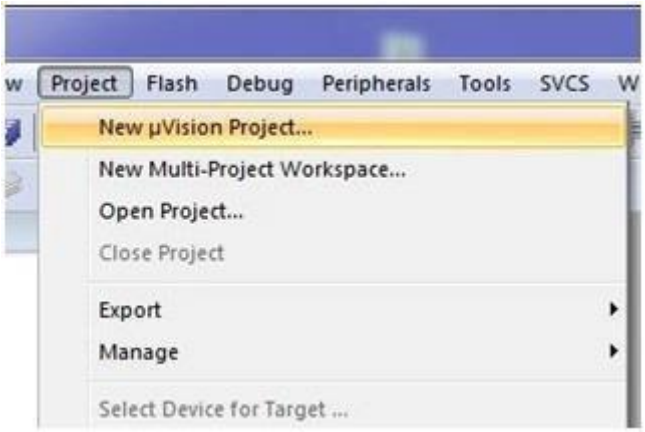
Procedure of using Keil:

Step 1: Download the Keil Uvision IDE

For learning purposes, you can try the evaluation version of Keil which has code limitation of 2K bytes. You must download the C51 version for programming on 8051 microcontroller architecture.

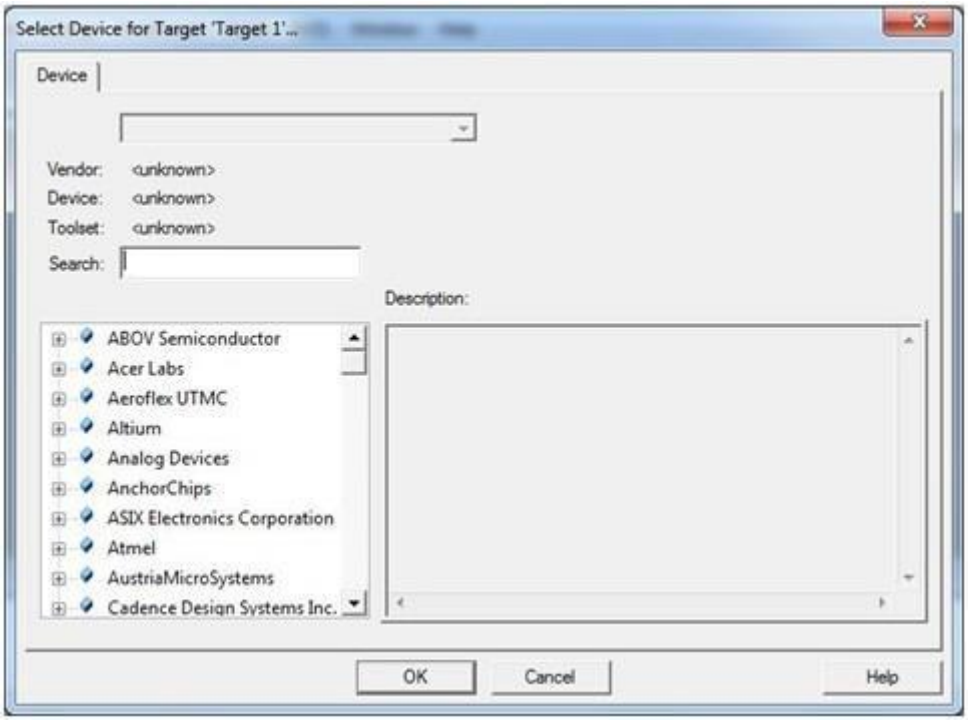
Step 2: To initiate the programming you must create a project using the keil Uvision IDE

The option to create a new project will be available under the project tab in the toolbar. Next, you have to store the project in a folder and give a suitable name to it.



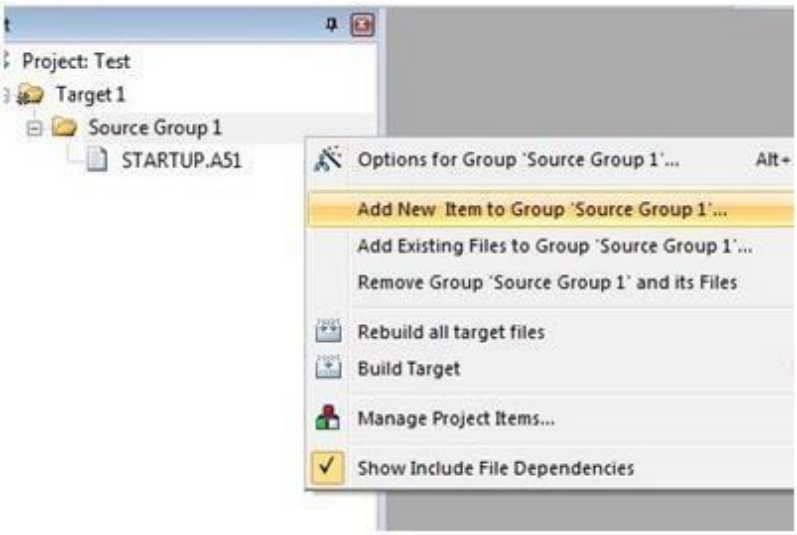
Step 3: Selecting the type of device you are working with

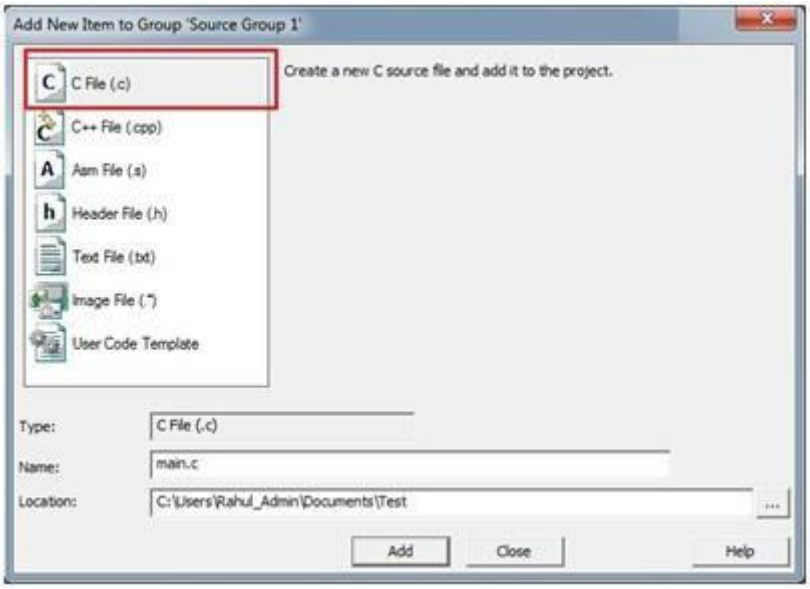
The device selection dialog provides you with the option to select the 8051 derivatives for which you want to develop the program. If you are not sure about your device you can refer to the description of the devices which is displayed on the left pane of the dialog. Accordingly, select your device and click OK to confirm.



Step 4: Adding C files to your project

You must add C file to your project before you begin coding. This can be done by right-clicking on your project from the project pane and selection "Add a new item to source group 1". In the next dialog box, you will be given with the choice on what type of file you want to add such as C, C++, header, text etc. Since here we are dealing with Embedded C programming, select C File (.c) option. Provide the necessary name, location and click on add.



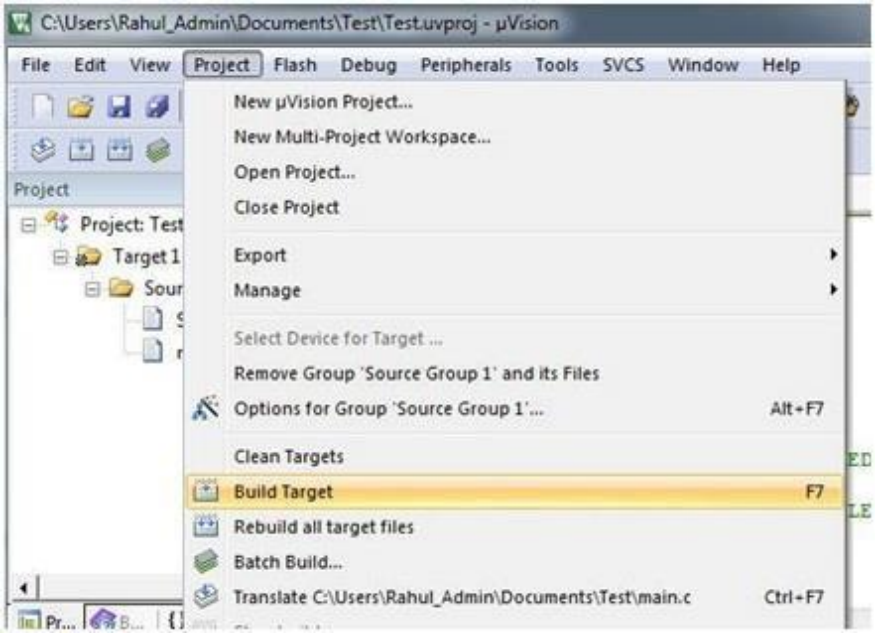


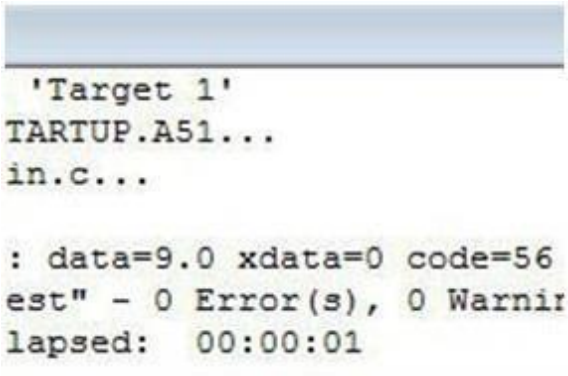
Step 5: Coding in C

The main part has finally arrived, so now you can go along with programming in C with your respective microcontroller.

Step 6 : Compiling and building the C project using Keil Uvision IDE

In order to build recently created C program go to Project tab and click on Build Target on the menu bar. An alternate way to do this is by pressing the F7 key. If the code that you have written is correct, the code will successfully compile without any errors. You can check your output in the Build Output pane.





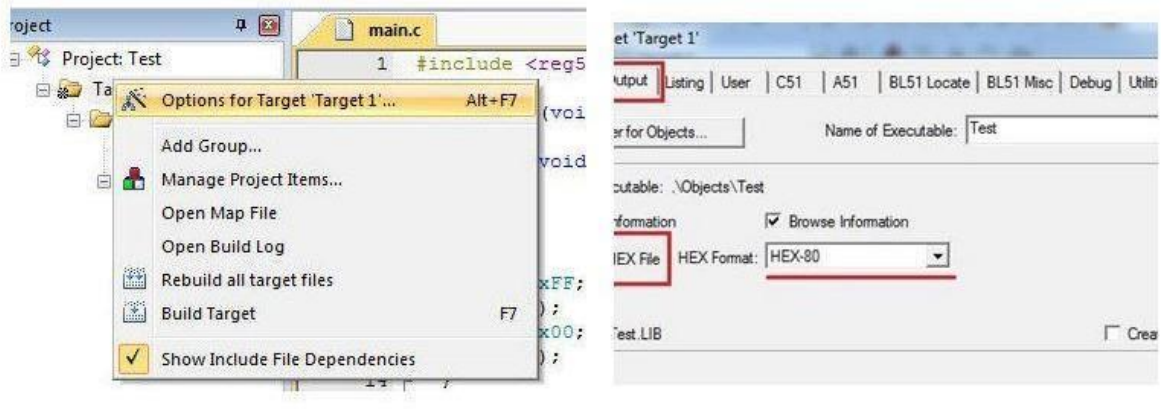
Step 7: Generating the hex file using Keil Uvision IDE

The code you compiled cannot be directly fed to the microcontroller, it is not possible. For that purpose, we have to generate the hex code for your respective file.

In order to generate the hex code, right click on the 'Target 1' folder and select options for target 'Target 1'. Select the Output tab in the target 'Target 1' dialog box. Make sure Create Hex File option is checked and the HEX format should be HEX-80. Click OK.

Again rebuild your project by pressing F7. Your required hex file would have been generated with the same as your project in the Objects folder.

If you wish you can also view your hex code by using a notepad.



Step 8: Burning the hex code into 8051 microcontroller

In order to burn the hex code to your microcontroller, there are two ways which are specific to the device you are working with. Some devices, for example, P89V51 they have their own built-in bootloader and you can burn the hex code directly through the serial port. Mostly you will require a specific programmer for your 8051 microcontroller through which you can easily upload your hex code by connecting the programmer via normal USB port.

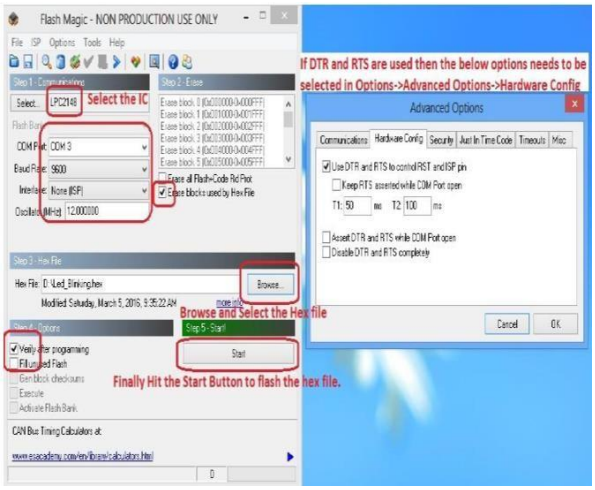
2. Flash Magic

Flash Magic is a PC tool for programming flash based microcontrollers from NXP using a serial or Ethernet protocol while in the target hardware.

Procedure for using Flash Magic:

1. Select the IC from Select Menu.
2. Select the COM Port. Check the device manager for detected Com port.
3. Select Baud rate from 9600-115200
4. Select None ISP Option.
5. Oscillator Freq 12.000000(12 MHz).
6. Check the Erase blocks used by Hex file option
7. Browse and Select the hex file.
8. Check the Verify After Programming Option.
9. If DTR and RTS are used then go to Options->Advanced Options-> Hardware Configuration and select the Use DTR and RTS Option.
10. Hit the Start Button to flash the hex file.

11. Once the hex file is flashed, Reset the board. Now the controller should run your application code.



CONCLUSION:

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 3

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

AIM/ OBJECTIVE: Write program for Interfacing Seven Segment Display to 8051 microcontroller.

HARDWARE/ SOFTWARE USED:

Keil software, SST flash, SST89E516RD2 microcontroller Trainer kit

Theory: Seven segment displays internally consist of 8 LEDs. In these LEDs, 7 LEDs are used to indicate the digit 0 to 9 and a single LED is used for indicating decimal point. Generally, seven segments are two types, one is a common cathode and the other is a common anode.

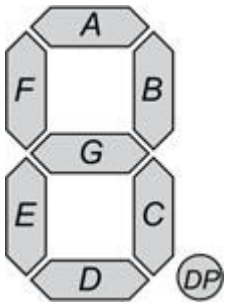


Fig.1 Seven segment

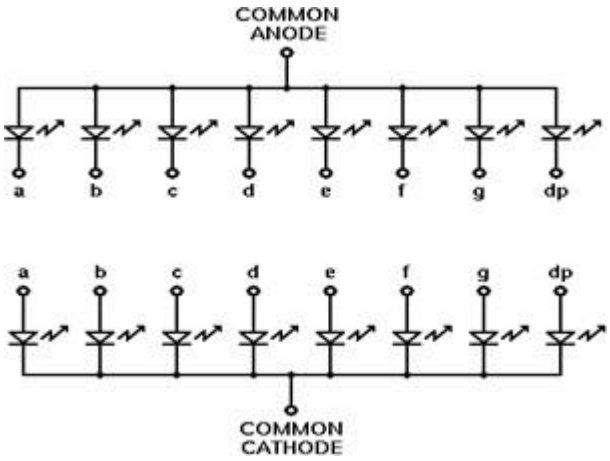


Fig.2 A common anode and Cathode LEDs

The common cathode, all the cathode of LEDs are tied together and labeled as com and the anode are left alone. In the common anode, seven-segment display all anodes are tied together and cathodes are left freely.

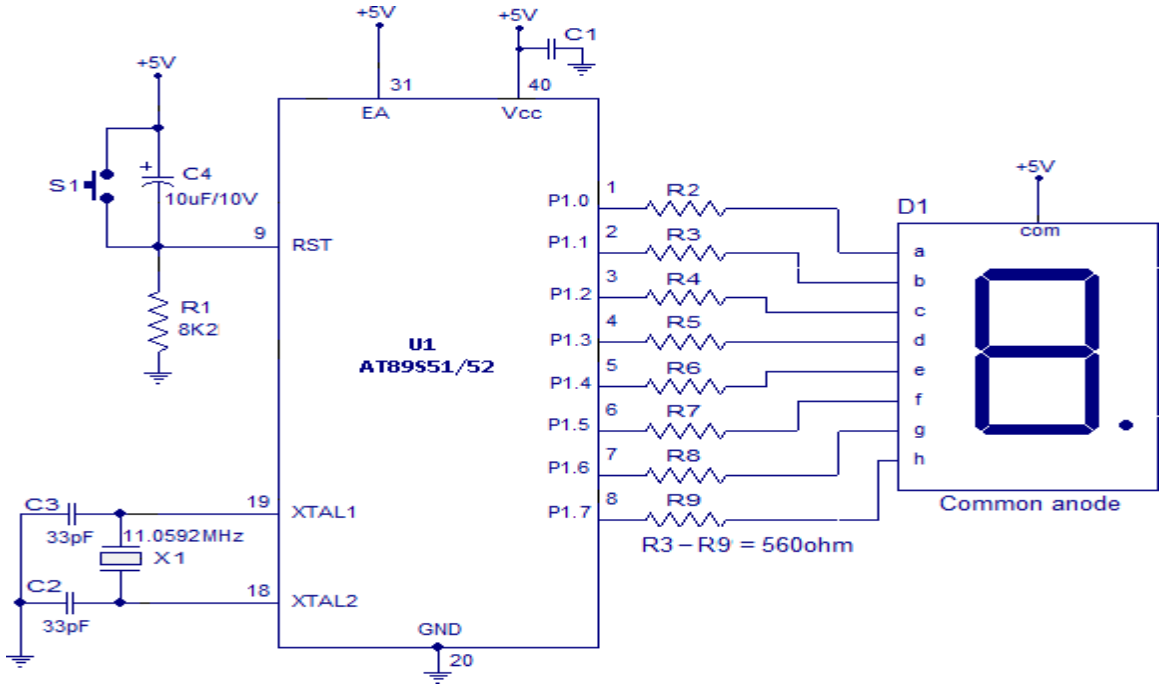
Seven Segment Truth Table for Common cathod:

Port P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
Digit	0	G	F	E	D	C	B	A	Hex form
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	1	1	1	1	6F

Seven Segment Truth Table for Common anode:

Port P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
Digit	0	G	F	E	D	C	B	A	Hex form
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	00	0	0	0	90

Interfacing Diagram:



Procedure:

1. Open the Keil µVision4 or 3 Software
2. Click on the project —> click on new µvision project —> type project name —> Click on Save
3. Select the Target from SST family controller —> click OK—> click YES on a copy STARTUP.A51 to Project
4. Click on File —> click on the NEW document —> Save this document with the filename.C extension.
5. Type C program and save using filename.c
6. After this in Project window Right-click on Source group 1—>click on Add existing file on source group 1—> click filename.asm—> Click on Add.

7. After this click on Project —> click Build Target(or pressF7 key) —> on Build the output window shows 0 errors and 0 warning then project is completely built if not then remove the error then build it.

8. Click on ‘Project’ —> Click on ‘option for target’ —> click on ‘Output’—>Tick on ‘create Hex file’—>click on OK —> click on build

9. The hex file store at the location of your project in the object folder.

10. Use SSTflash software to burn hex file in SST89E516RD2 microcontroller trainer kit.

CONCLUSION:

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 4

TITLE: Interfacing of Stepper motor to 8051- software delay using Timer

AIM/ OBJECTIVE: Interfacing of Stepper motor to 8051.

HARDWARE/ SOFTWARE USED:

Keil software, SST flash,, SST89E516RD2 Microcontroller Trainer kit, Stepper Motor

Theory: A stepper motor is a widely used device that translates electrical pulses into mechanical movement. In applications such as disk drivers, dot matrix printers, and robotics, the stepper motor is used for position control. Stepper motors commonly have a permanent magnet rotor (also called the shaft) surrounded by a stator. The most common stator motors have four stator windings that are paired with a center tapped common. The centre tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator. The stepper motor shaft moves in a fixed repeatable increment, which allows one to move it to a precise position. This repeatable fixed movement is possible as a result of basic magnetic theory where poles of the same polarity repel and opposite poles attract. The direction of the rotation is dictated by the stator poles. The stator poles are determined by the current sent through the wire coils. As the direction of the current is changed, the polarity is also changed causing the reverse motion of the rotor.

Table: Normal 4-step Sequence

Clockwise ↓	Step	Winding A	Winding B	Winding C	Winding D	Counter clockwise ↑
	1	1	0	0	1	
	2	1	1	0	0	
	3	0	1	1	0	
	4	0	0	1	1	

This table shows a 2-phase, 4-step stepping sequence

Step angle: -

The step angle is the minimum degree of rotation associated with a single step. Various motors have different step angles.

Step Angle	Steps per Revolution
0.72	500
1.8	200
2.0	180
2.5	144
5.0	72
7.5	48
15	24

Steps per second and rpm relation:-

$$\text{Steps per second} = \frac{(\text{rpm} \times \text{steps per revolution})}{60}$$

The four –step sequence and number of teeth on rotor:-

After completing every four steps, the rotor moves only one tooth pitch. Therefore, in a stepper motor with 200 steps per revolution, the rotor has 50 teeth since 4 x 50 = 200 steps are needed to complete one revolution. This leads to conclusion that 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.

For example a motor with a 2-degree step angle has following characteristics:

Step angle: 2 degrees Steps per revolution: 180

Number of rotor teeth: 180/4= 45 Movement per 4-step sequence: 8 degrees

To allow for finer resolution, all stepper motors allow an 8-step sequence which is also called half stepping.

Table: Half-step 8-step sequence

	Step	Winding A	Winding B	Winding C	Winding D	
Clockwise ↓	1	1	0	0	1	Counter clockwise ↑
	2	1	0	0	0	
	3	1	1	0	1	
	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	

Motor speed: The motor speed, measured in steps per second (steps/s), is a function of the switching rate.

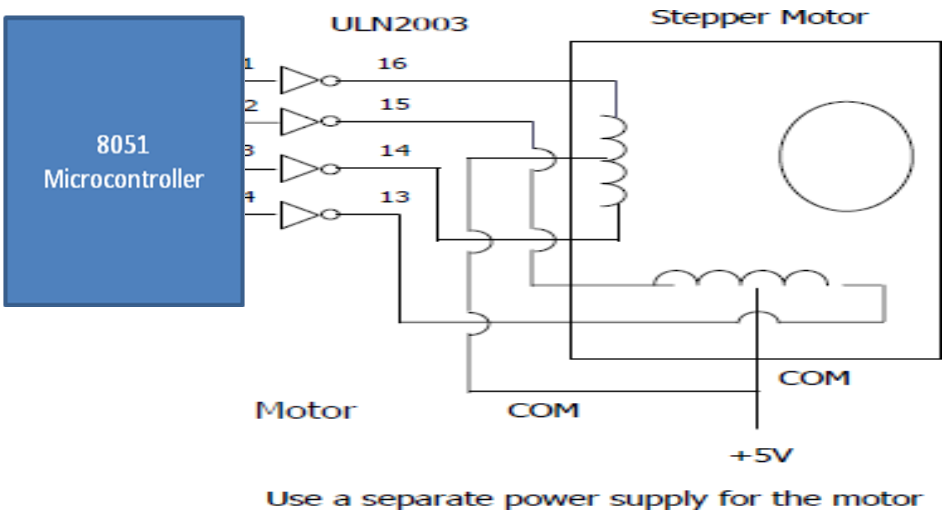
Holding torque: With the motor shaft at standstill or zero rpm condition, the amount of torque, from an external source required to break away the shaft from its holding position. This is measured with rated voltage and current applied to the motor.

Wave drives 4-step sequence: The 8-step sequence is the combination of the wave drive 4-step and normal 4-step sequences.

Unipolar versus bipolar stepper motor interface: There are three common types of stepper motor interfacing: universal, unipolar, and bipolar. They can be identified by the number of connections to the motor. A universal stepper motor has eight, while the unipolar has six and the bipolar has four. The universal stepper motor can be configured for all three modes, while the unipolar can be either unipolar or bipolar. Obviously the bipolar cannot be configured for universal nor unipolar mode.

8051 connection to the stepper motor:-

Interfacing diagram:-



The 8051 lacks sufficient current to drive the stepper motor windings, we must use a driver such as the ULN 2003 to energize the stator. However if the transistors are used as drivers, we must also use diodes to take care of inductive current generated when the coil is turned off. One reason that using the ULN2003 is preferable to the use of transistors as drivers is that ULN2003 has an internal diode to take care of back EMF.

CONCLUSION:

DPU

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 5

TITLE: Write a program for interfacing button, LED, relay & buzzer as follows

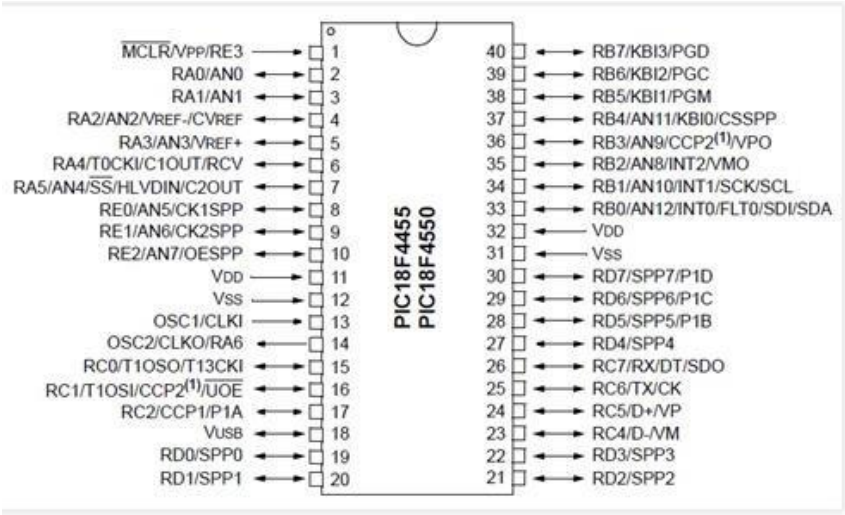
AIM/ OBJECTIVE : Write a program for interfacing button, LED, relay & buzzer with PIC18F4550 as follows:

- When button 1 is pressed, relay and buzzer is turned off and LED's chase in right direction.
- When button 2 is pressed, relay and the buzzer is turned On and LED's chase in left direction.

HARDWARE/ SOFTWARE USED: MPLAB IDE v7.51, PIC Loader, PIC18F4550 Trainer Kit

Theory: PIC is a family of Harvard architecture microcontrollers made by Microchip Technology. The PIC architecture is distinctively minimalist. It is characterized by the following features:

- Separate code and data spaces (Harvard architecture).
- A small number of fixed length instructions.
- Single-cycle execution (4 clock cycles).
- A single accumulator (W), the use of which (as source operand) is implied (i.e. is not encoded in the opcode).
- All RAM locations function as registers as both source and/or destination of math and other functions.
- 32 Kb Flash memories, 2K RAM, 256 bytes EEPROM.
- 2 comparators, 10 bit 13 channel ADC, 35 I/O Ports, 3 Timers, SPI and I2C supported.

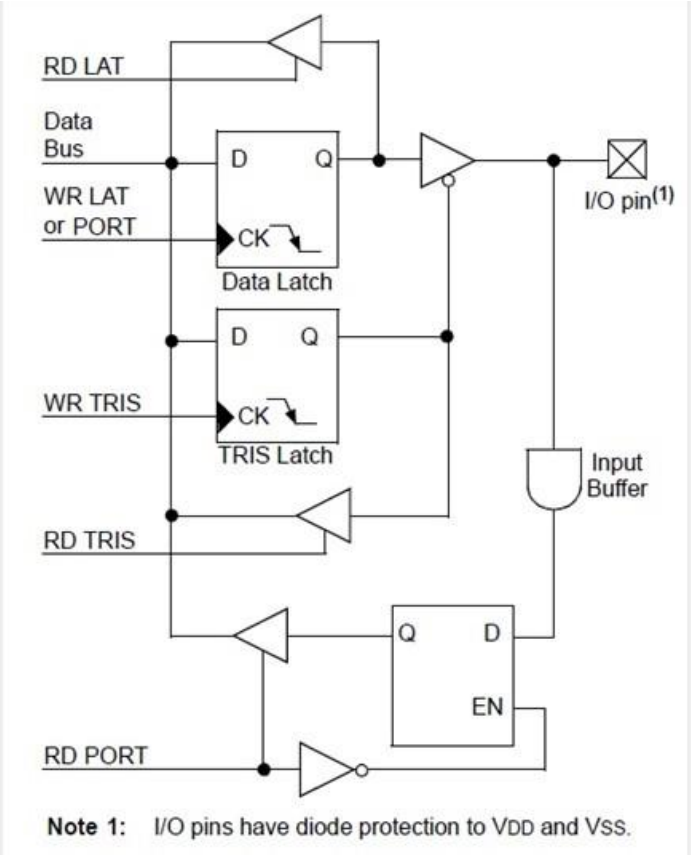


I/O Ports:

Depending on the device selected and features enabled, there are up to five ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

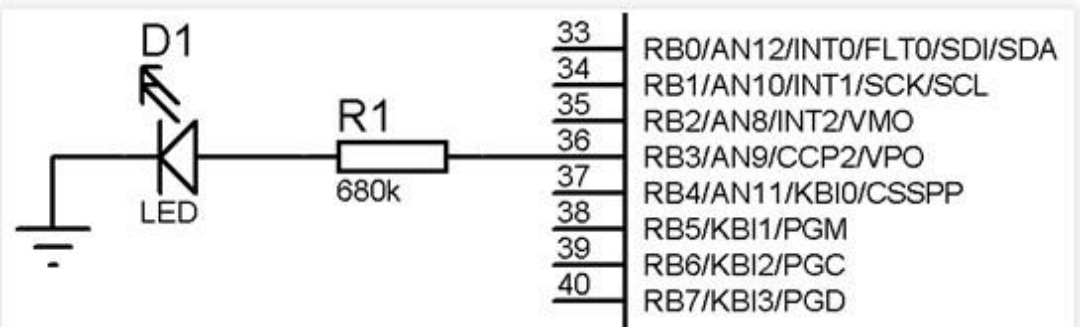
Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)
- The Data Latch register (LATA) is useful for read - modify-write operations on the value driven by the I/Opins.
- Reading the PORT A register reads the status of the pins; writing to it will write to the port latch.
- A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure.



- Writing a '1' to the TRIS register configures the pin as input; whereas writing a '0' configures the pin as output.
- When configured as output, data can be written on the I/O pin using the LAT or the PORT register.
- When configured as input, the status of the I/O pin can be read using the PORT register.

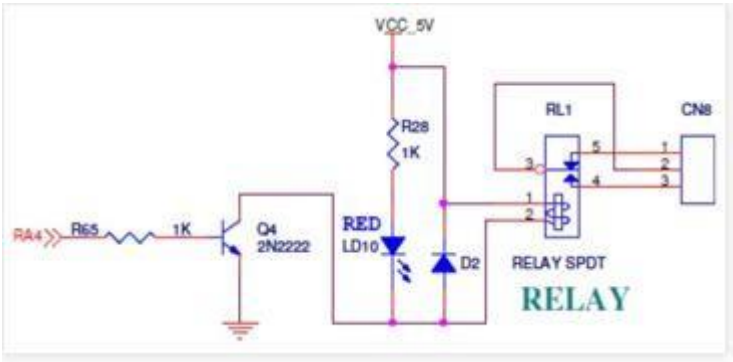
Interfacing LED with microcontroller:



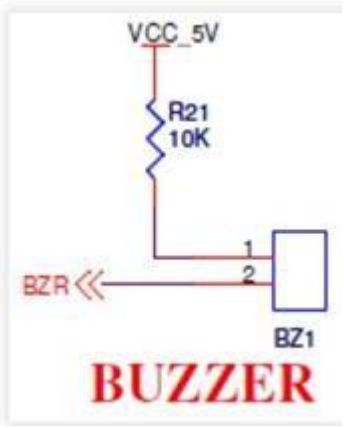
To drive the LED, following steps must be followed:

1. Configure the pin connected to the LED as output. This is done by writing 0 to the TRIS register. This can be done in 2 ways:
    a. Configuring entire PORTB as output:
        TRISB = 0b00000000; or 0x00
    b. Configuring only the RB3 pin as output:
        TRISBbits.TRISB3 = 0; other pins remain unaffected.
2. Writing the data on the PORT pin to drive the LED ON or OFF. To turn the LED ON, we can follow one of these steps:
    a. LATB = 0b00001000; or 0x08 or
    b. PORTB= 0b00001000; or 0x08.
    c. LATBbits.LATB3 = 1; or
    d. PORTBbits.RB3 = 1;
3. To turn the LED OFF, we can follow one of these steps:
    a. LATB = 0b00000000; or 0x00
    b. PORTB=0b00000000; or 0x00
    c. LATBbits.LATB3 = 0; or
    d. PORTBbits.RB3 = 0;

Interfacing Relay and Buzzer with microcontroller:



Relay Interfacing



Buzzer interfacing

To turn the relay and buzzer ON and OFF, follow the same steps as that of LED. However, from the figure, we can see that the logic of the the buzzer is inverted i.e. when we write '0' to the PORT pin connected to the buzzer, it is turned ON; whereas on writing a '1' turns the buzzer OFF.

Interfacing a BUTTON with microcontroller:

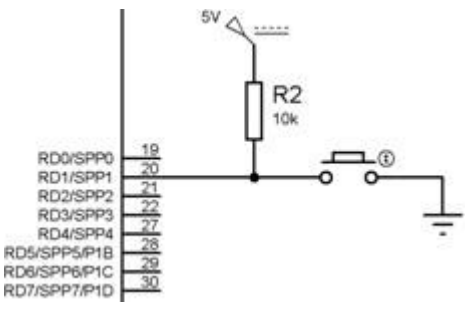
To interface a push-button with a pin of the microcontroller, one end of the button is grounded, while the other end is connected to the pin of the microcontroller.

The microcontroller pin is pulled HIGH with the help of a Pull-Up resistor or by setting the LAT register bit HIGH.

When the button is not pressed i.e. when it is open, the PORT pin receives logic '1' through the Pull-Up resistor or through the LAT register.

When the button is pressed, the PORT pin receives logic '0' since it is connected to the ground directly via the closed push-button.

The status or the logic received by the pin is read with the help of the PORT register.



Button Interfacing

Procedure:

1. Open MPLABX IDE and create a new project.
   File-> New Project->Microchip Embedded -> Standalone Project.
   Select device PIC18F4550.
   Select compiler XC8.
   Project name -> Finish
2. Create a C source file: Right click on the Source Files
   New -> C source file
   Give a name to the file.
3. Copy or type the source program in the C file created.
4. Save the file.
5. Right-click on the project name, and select “Clean and Build”.
6. Open PIC Loader.
7. Press F3
8. Press Reset
9. Press F4

CONCLUSION :

DPU

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 6

TITLE: Interfacing of LCD to PIC 18FXXXX

AIM/ OBJECTIVE: LCD Interfacing with PIC18F4550 and with 8051 microcontroller

HARDWARE/ SOFTWARE USED: MPLAB IDE v7.51, Pic loader, Pic 18F4550 Trainer kit, Keil software, SST flash ,SST89E516RD2 microcontroller Trainer kit.

Theory:

INTRODUCTION TO LCD:-

Liquid crystal displays 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. Hitachi 44780 LCD controller provides an easy solution to interface it with microcontrollers.12 microcontroller pins are needed for this interface.

LCD is finding widespread use replacing LEDs:-

- The declining prices of LCD
- The ability to display numbers, characters,and graphics
- Incorporation of a refreshing controller
- into the LCD, thereby relieving the CPU of the task of refreshing the LCD
- Ease of programming for characters and graphics

THE LCD MODULE:-

The LCD has 16 pins in general, but usually 14 pins are of importance, pin number's 15 and 16 are used for backlight control.

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. A standard LCD controller chip HD 44780U ca receives data from a microcontroller & communicates with the LCD. Figure shows the interfacing of LCD with microcontroller describing various pins whose description has been given in the table. There are 3 control lines & 8 (or 4 in case of 4-bit data) data lines. The three control signals are RS,E,R/W.

PIN DSICRIPTION OF LCD(16*2):

The most commonly used LCDs found in the market today are 1 Line, 2 Line or 4 Line LCDs which have only 1 controller and support at most of 80 characters, whereas LCD's supporting more than 80 characters make use of 2 HD44780 controllers.

Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections). Pin description is shown in the table below.

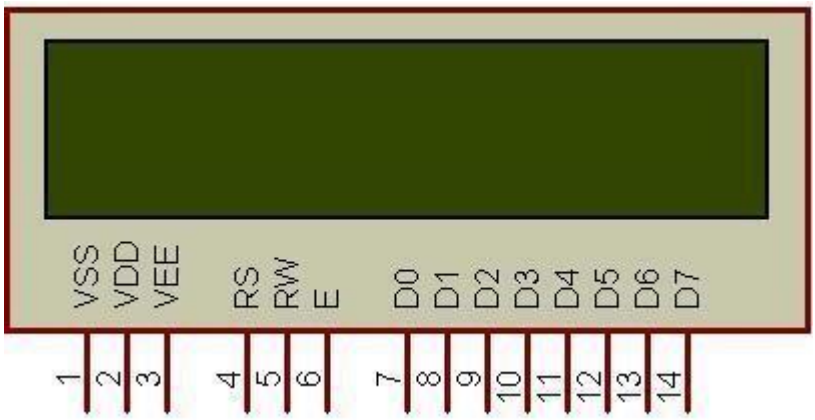
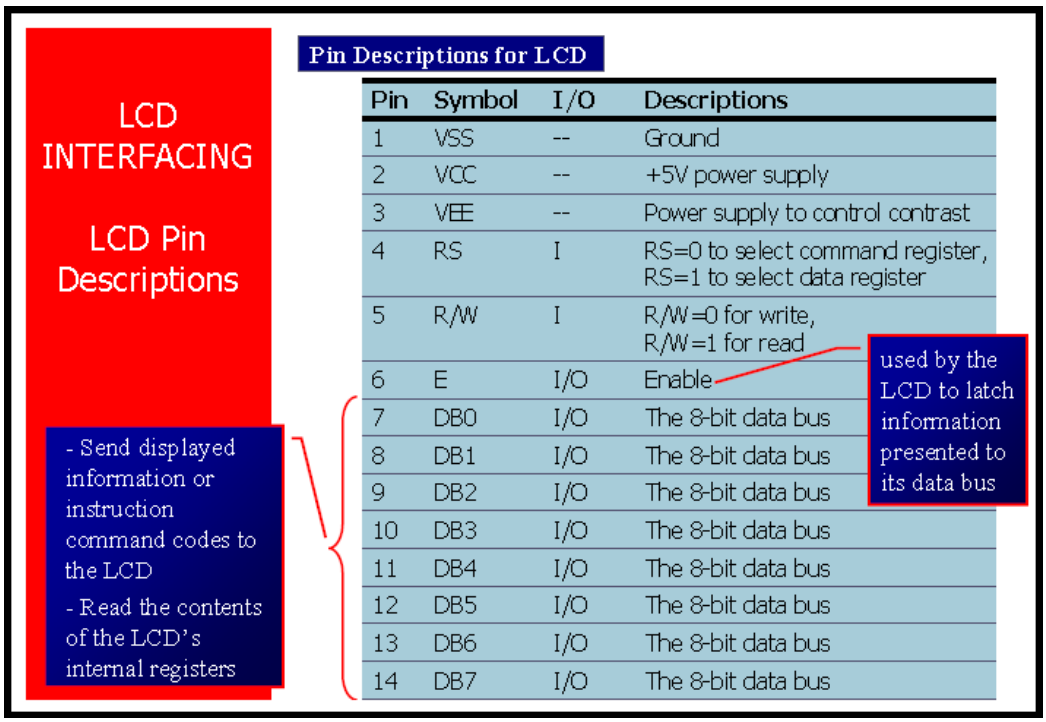


Figure 1: Character LCD type HD44780 Pin diagram



Pin Descriptions for LCD

Pin	Symbol	I/O	Descriptions
1	VSS	--	Ground
2	VCC	--	+5V power supply
3	VEE	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

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

'Enable-EN' 'register select-RS' & 'read/write-RW'.:-

EN line is to instruct the LCD that microcontroller is sending the data. This line is first made high by microcontroller, then the other control lines are RS &RW are defined & data is put on the data bus. After the data has been put on the data bus the EN is made low, means now the data is treated a command or instruction now to the data module. When RS line is high, the data is a text data to be displayed on the LCD.

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. The read/write line RW, when made low, the information on the data bus is written to the LCD & when the RW line is high, it means that the data is being read from the LCD. It is very important to note that EN line must go from high to low for LCD to execute the instruction. The 8 bit data pins,D0-D7 are used to send the information to LCD or read the content of the LCD internal register. There are also instruction command codes that can be sent to the LCD to clear the display or force the cursor to the home position or blink the cursor.The most commonly used LCDs found in the market today are 1 Line, 2 Line or 4 Line LCDs which have only 1 controller and support at most of 80 characters, whereas LCD's supporting more than 80 characters make use of 2 HD44780 controllers. Most LCDs

with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections). Pin description is shown in the table below.

Pin No.	Name	Description
Pin no. 1	**D7**	Data bus line 7 (MSB)
Pin no. 2	**D6**	Data bus line 6
Pin no. 3	**D5**	Data bus line 5
Pin no. 4	**D4**	Data bus line 4
Pin no. 5	**D3**	Data bus line 3
Pin no. 6	**D2**	Data bus line 2
Pin no. 7	**D1**	Data bus line 1
Pin no. 8	**D0**	Data bus line 0 (LSB)
Pin no. 9	**EN1**	Enable signal for row 0 and 1 (1^{st}controller)
Pin no. 10	**R/W**	0 = Write to LCD module 1 = Read from LCD module
Pin no. 11	**RS**	0 = Instruction input 1 = Data input
Pin no. 12	**VEE**	Contrast adjust
Pin no. 13	**VSS**	Power supply (GND)
Pin no. 14	**VCC**	Power supply (+5V)
Pin no. 15	**EN2**	Enable signal for row 2 and 3 (2^{nd}controller)
Pin no. 16	**NC**	Not Connected

LCD ADDRESSING for 16x2:-

16x2 LCD	80	81	82	83	84	85	86 through 8F
	C0	C1	C2	C3	C4	C5	C6 through CF

LCD COMMANDS CODES:-

LCD INTERFACING

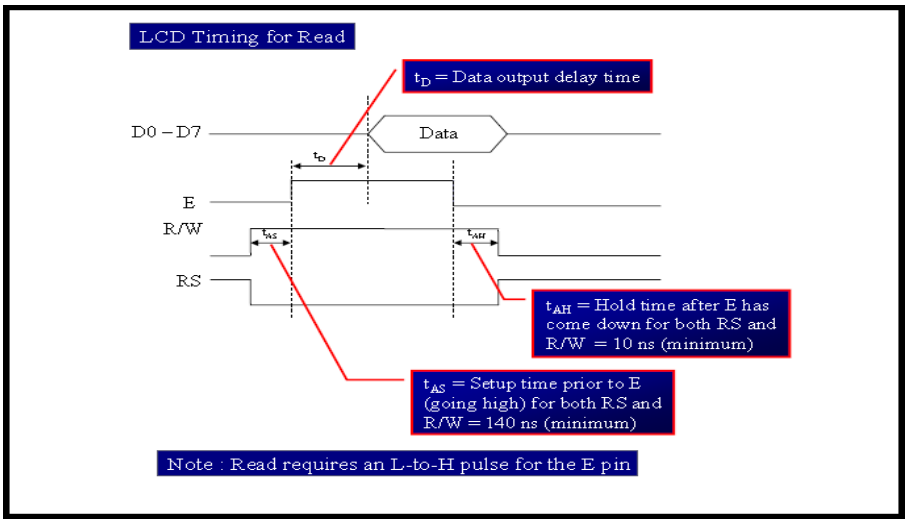
LCD Command Codes

LCD Command Codes

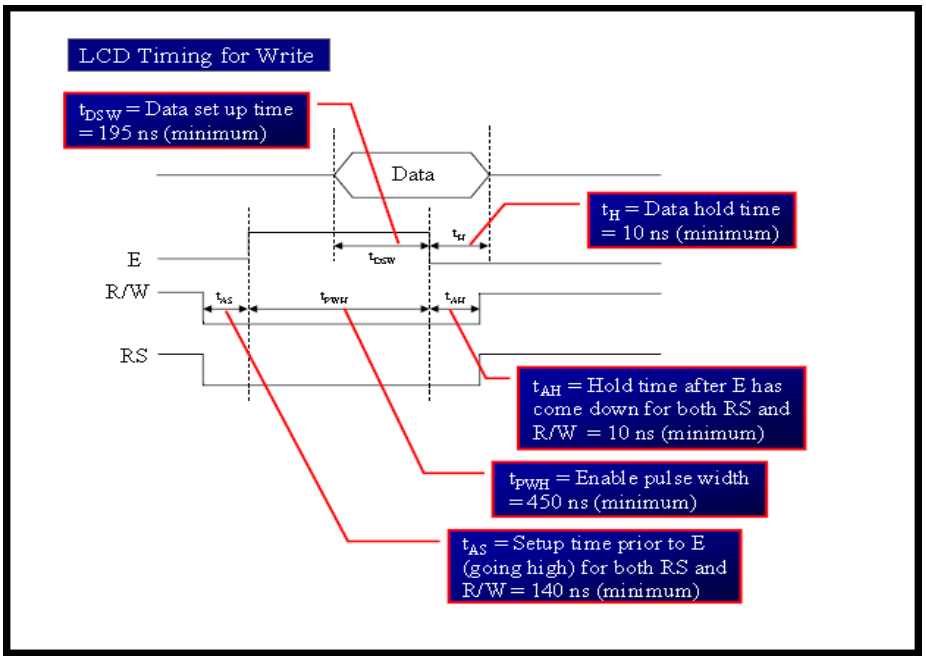
Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

NOTE:-Generally used commands by any user : 38H,0EH,01H,06H,84H.

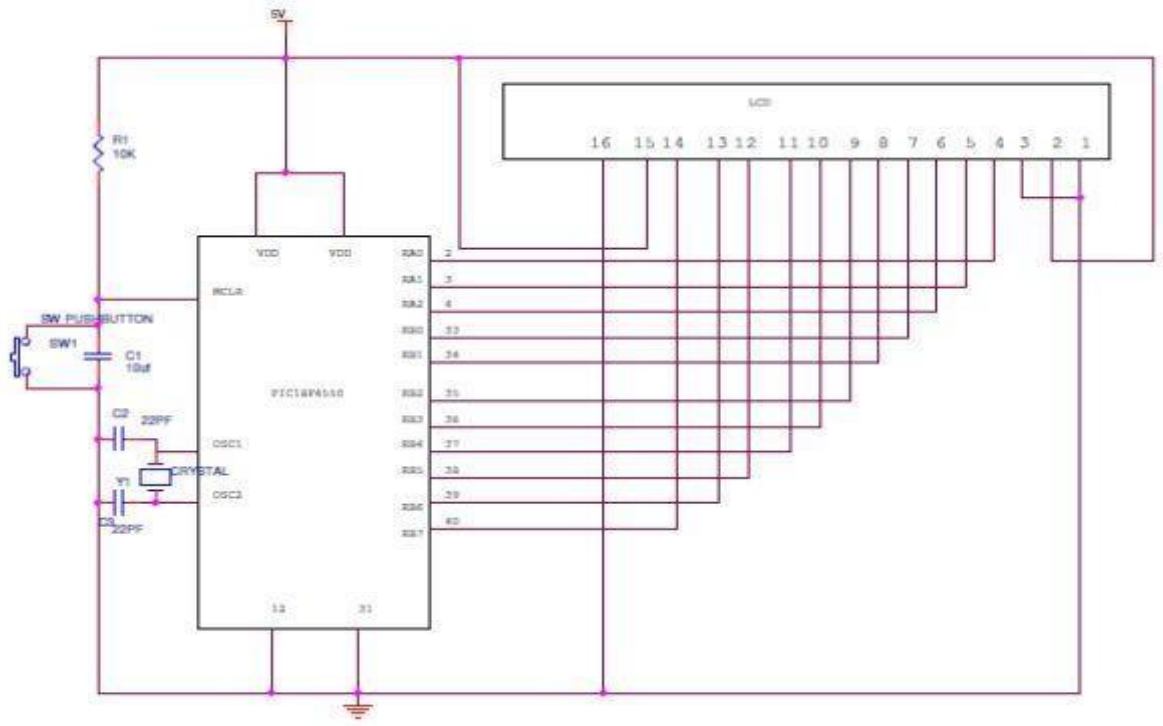
TIMING DIAGRAM FOR READ:-



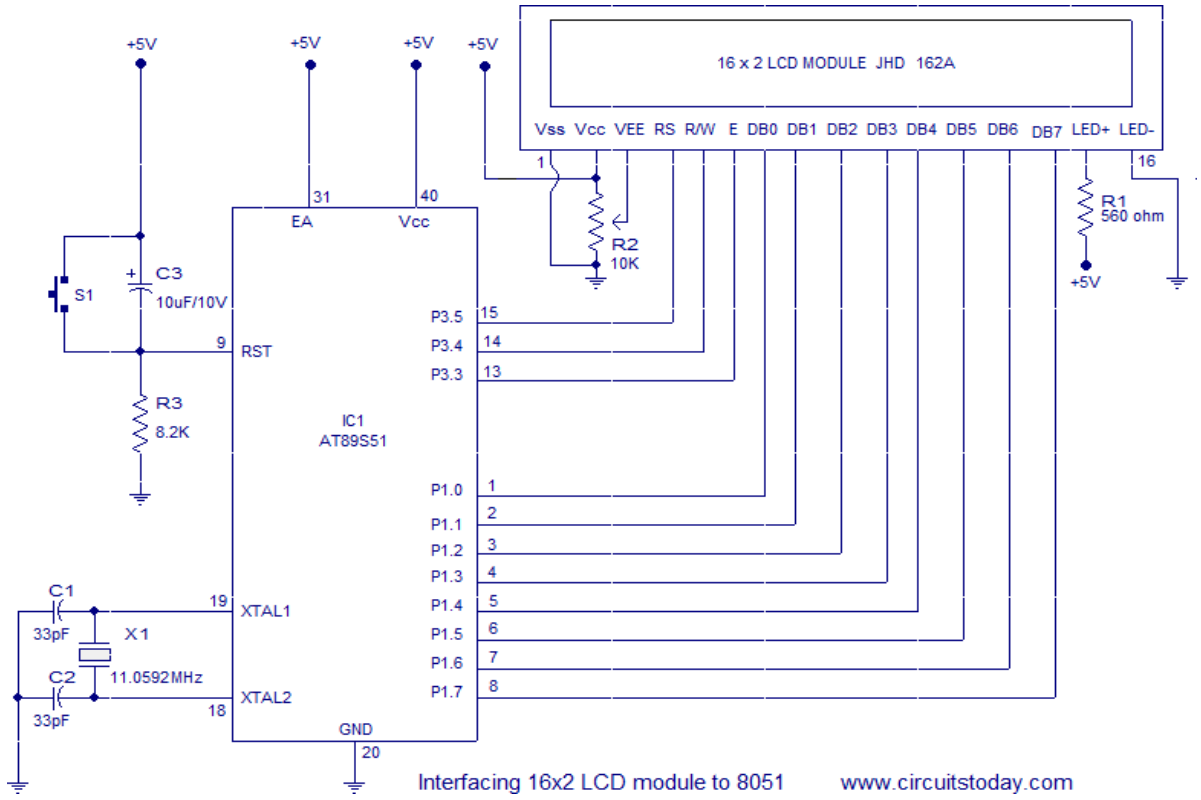
TIMING DIAGRAM FOR WRITE:-



Interfacing of LCD with PIC18F4550:



Interfacing of LCD with 8051 :



Programming Steps:

Before displaying anything on LCD, it needs to be configured with proper instructions. The following programming steps explain the procedure of configuring the LCD and display a character on it.

Step 1: Initialize the LCD.

The LCD must be initialized the by following pre-defined commands of character LCD.

- 0x38, to configure the LCD for 2-line, 5×7 font and 8-bit operation mode
- 0x0C, for Display On and Cursor Off
- 0x01, to Clear Display screen
- 0x06, to increment cursor
- 0x80, to set cursor position at first block of the first line of LCD.

The above set of commands is written in lcd_ini() function of the adjoining code.

Step 2: Send the commands to LCD.

- Send the command byte to the port connected to LCD data pins
- RS=0, to select command register of LCD
- RW=0, to set the LCD in writing mode
- EN=1, a high to low pulse to latch command instruction
- Delay of 1ms
- EN=0

The above set of commands is written in lcdcmd(unsigned char) function.

Step 3: Send data to LCD.

- Send data at the port which connected to LCD data pins
- RS=1, register select to select data register of LCD
- RW=0, this set the LCD in writing mode
- EN=1, a high to low pulse to latch data
- Delay of 1ms
- EN=0

The lcddata(unsigned char) function has the above set of instructions.

Step 4: Display character on LCD.

The functions lcdcmd() and lcddata() are user-defined functions. They are used to send a character (E in this case) to be displayed on LCD.

```
lcdcmd(0x38);          // send command 0x38 to LCD
lcddata('E');          // send character E to LCD
```

CONCLUSION :

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 7

TITLE: Generate square wave using timer.

AIM/ OBJECTIVE: Write embedded C language program to generate a square wave of 50Hz any port pin.

HARDWARE/ SOFTWARE USED:

1. Keil uVision2 Compiler / Assembler
2. SST Flash Software for downloading
3. 8051 Trainer kit.

Theory:

Time Delay Generation:

Most used subroutine is one that generates a programmable time delay. Time delays may be done by using software loops that essentially do nothing for some period or by using hardware timers that count internal clock pulses. The key to writing this program is to calculate the exact time each instruction will take at the clock frequency in use. Following terms are very helpful to write a subroutine which generates desired time delay.

T-state: T-state is defined as one subdivision of the operation performed in one clock period. The terms T-state and clock period are often used synonymously.

Machine Cycle: Machine cycle in 8051 is defined as 12 oscillator periods. The 8051, take one to four machine cycles to execute an instruction. To calculate the machine cycle for the 8051, we take 1/12 of crystal frequency, and then take its inverse.

Assume crystal frequency of 11.0592 MHz

M/C frequency = 11.0592 MHz/12 = 921.6 KHz

Machine Cycle = 1/921.6 KHz = 1.085 us (microseconds)

Instruction Cycle: Instruction cycle is defined as the time required for completing the execution of an instruction. One instruction cycle consists of one to four machine cycles.

e.g. 2 Machine cycles are required for instruction DJNZ R2, target to be executed. Then instruction cycle is calculated as follows.

Instruction cycle = No. Machine cycles x Machine cycle period

= 2 x 1.085 us

= 2.17 us

Timer /Counter in 8051:

8051 has 2 timer/ counter. They can be used a timers to generate a time delay or as counter to count events happening outside the microcontroller. Both Timer 0 and Timer 1 are 16 bits wide. Since the 8051 has an 8-bit architecture, each 16-bit timer register is accessed as two separate registers of low byte and high byte.

- Timer 0 can be accessed as –
- TL0 – Timer 0 lower byte
- TH0 – Timer 0 higher byte
- Timer 1 can be accessed as –
- TL1 - Timer 1 lower byte
- TH1 – Timer 1 higher byte

Both timer shares the Timer control (TCON) register, which controls the timer/ counter operation and Timer mode (TMOD) register, which is used to configure the timer for different operating modes.

TMOD (Timer Mode Register):

GATE	T/C	M1	M0	GATE	T/C	M1	M0
Timer/Counter1				Timer/Counter0			

Both the timers used the same 8 bit register to set various timer operation mode. TMOD is 8-bit register where lower 4 byte are set aside for timer 0 and higher 4 bytes for timer 1. Since, it is not bit addressable; the corresponding bit value is directly loaded into TMOD.

Gate:

8051 has both hardware and software controls to start and stop the timers. By the means of software controlling instruction timers are used to control to start timer or stop.

C/T:

This bit in TMOD is used to determine whether timer is to be used as delay generator or event counter.

If C/T = 0 – used as timer

If C/T = 1 – used as counter

M1 M0:

M1, M0 selects the timer mode.

M1	M0	Mode	Operation
0	0	0	13 bit counter, 8 bit C/T with THX and TLX as 5 bit Prescalar.
0	1	1	16 bit counter, 8 bit C/T with THX and TLX cascaded with no Prescalar.
1	0	2	8 bit auto reload, THX hold the value which is to be loaded into TLX after each overflow.
1	1	3	Split timer mode.

TCON (Timer Control Register):

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Timer run control bits TR0 and TR1 and timer overflow flags TF0 and TF1 are the part of 8 bit register called TCON. The upper 4 bits are used to store TR and TF flags of both timer- 0 and timer 1 while the lower 4 bits are set aside for interrupt. Timer run control bit TR0/TR1 is used to start the corresponding timer / counter. Timer overflow flag bit TF0/TF1 is set when corresponding timer/ counter is overflowed i.e. count value FFFF h to 0000 h.

Different Modes of timer/ counter –

1. Mode 0 (13 bit timer/counter):

- Mode 0 is exactly like mode 1 except it is 13 bit timer.

- Hence it can hold the values from 0000H to 1FFFH in TL and TH.
- When timer rolls over from 1FFFH to 0000H, the overflow flag i.e. TFX is set.

2. Mode 1 (16 bit Timer / counter):

- It is 16 bit timer. Hence allowed values from 0000H to FFFFH to be loaded in TLX and THX.
- After the corresponding 16 bit value is loaded, the timer is started by setting TRX flags.
- After starting, it counts up until it reaches the limit i.e. FFFFH. When it rolls over from FFFFH to 0000H, it sets timer overflow flag i.e. TFX flag.
- After the rolling over process, the operation in mode 1 can be repeated by loading the initial value in TLX & THX and clearing TFX bit.

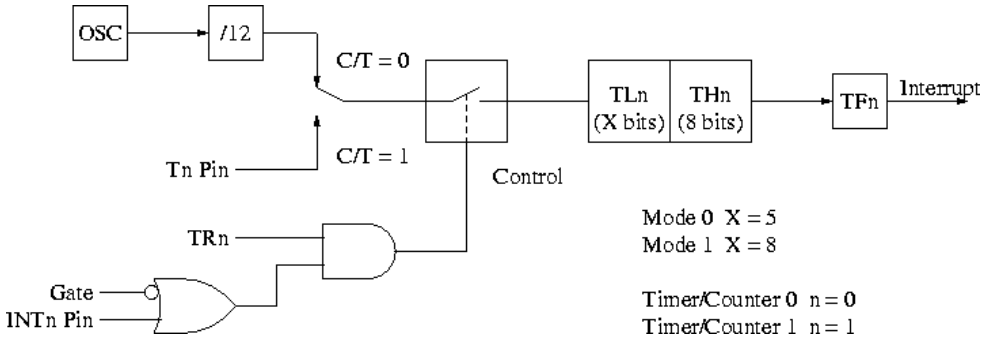


Figure 3.1: Timer/ Counter in Mode 0 and Mode 1

3. Mode 2 (8 bit auto reload):

- It is 8 bit timer. Hence it allows only values from 00H to FFH to be loaded in THX.
- When THX is loaded into 8 bit value, it sends a copy of it to corresponding TLX. Then timer must be started which is done by SETB TR1 for T1.
- After rolling over of TLX from FFH to 00H, overflow flag for corresponding timer i.e. TFX is set. TLX auto loaded by value present in THX.

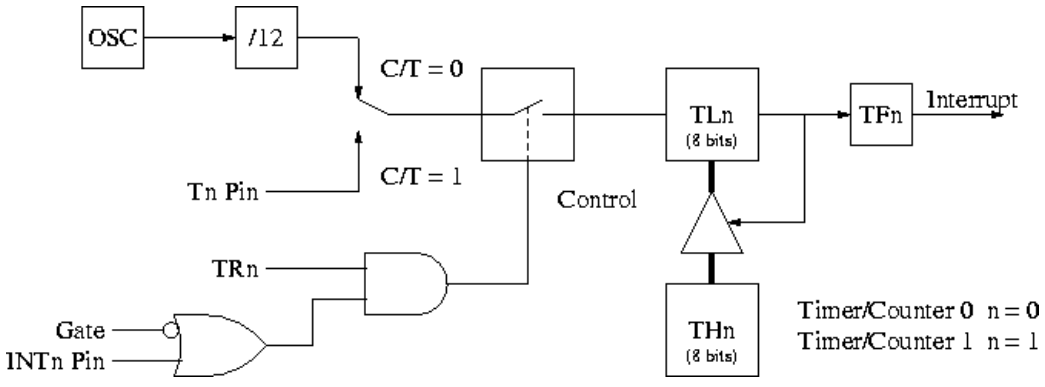


Figure 3.2: Timer/ Counter in Mode 2

4. Mode 3 (Split timer mode):

- In mode 3, timer0 worked as split timer i.e. two independent timer.
- TL0 uses the TR0 and TF0 bits of timer 0.
- TH0 uses the TR1 and TF1 bits of timer 1.

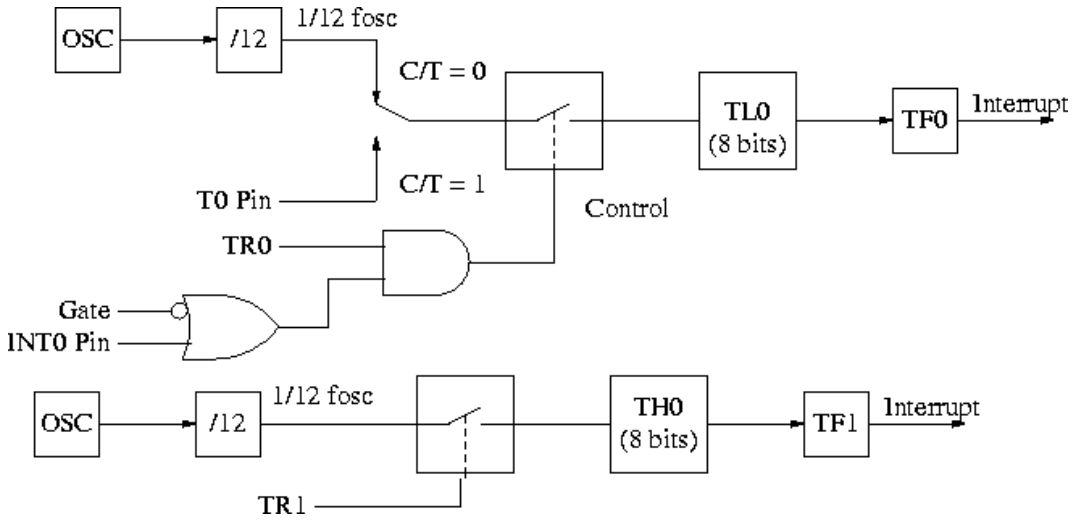


Figure 3.3: Timer/ Counter in Mode 3

Time Delay Generation using Timer:

Mode 1 and Mode 2 are widely used for most of the applications. Mode 1 is used for time delay generation and mode 2 is used to generate the baud rate in serial communication.

The following steps are taken to generate a time delay using the mode 1 and polling method.

1. Load the TMOD value indicating which timer is to be used and timer mode 1 is selected.
2. Load the registers TL and TH with initial count values. Delay generated is Depends upon the initial count value.
3. Start the timer.
4. Keep the monitoring the timer flag (TF)
5. Get out of the loop when TF becomes high.
6. Stop the timer.
7. Clear the TF flag for the next round.
8. Go back to the step 2 to load TH and TL values.

The size of the time delay depends on two factors, (a) the crystal frequency and (b) the timer's 16-bit register in mode 1. The largest delay is achieved by the making both TH and TL zero.

Formula for delay calculations using mode 1 of the timer for crystal frequency of XTAL = 11.0592 MHz, (TH, TL) = $(NNNNN)_{10}$ is as follows.

Time Delay (Td) = (65536 - NNNNN) x 1.085 us.

Therefore Maximum delay = (65536 – 0000) x 1.085 us = 71 ms.

Finding the Values to be loaded into timer for desired delay

Assume the 11.0592 MHz as crystal frequency for 8051.

i. Divide the desired time delay by 1.085us. (n = Td/1.085us)
ii. Perform 65536 – n. where n is the decimal value from step 1.
iii. Convert the result of step 2 to hexadecimal, where yyxx is the initial hex value to be loaded into the timer's registers.
iv. Set TL = xx and TH = yy.

To generate the 2 KHz square wave on port pin

A. Main Program

1. Load the value "10h" in TMOD register indicating Timer-1 is to be used and timer mode 1 is selected.
2. Load the registers TL and TH with initial count values i.e. DC00h. (TH = DCh, TL = 00h)
3. Start the timer by setting TR1 bit in TCON register.
4. Halt the program.

CALCULATIONS:

Frequency = 50 Hz

Machine Cycle = 1.085 us (microseconds)

Time period = T_P = 1/50Hz = 20 ms

Required duty cycle is 50%

Therefore, $T_{on} = T_{off} = 1/20 = 10$ ms

Desired time delay is $T_D = 10$ ms

Divide T_D by $1.085 \times 10^{-6} = n = 10 \times 10^{-3} / 1.085 \times 10^{-6} = 9216$

Subtract n from 65536 = 65536 – 9216 = 56320

Convert above decimal value in to Hex value = DC00Ah

Load this value into Timer Register. (TH = DCh , TL = 00h)

CONCLUSION :

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 8

TITLE: Interface analog voltage 0-5V to internal ADC and display value on LCD

AIM/ OBJECTIVE: Interface analog voltage 0-5 to internal ADC and display value on LCD

HARDWARE/ SOFTWARE USED: MPLAB IDE, PIC Loader, PIC18F4550 Trainer Kit

THEORY: PIC 18F4520 has on chip 10 bit ADC.

It has 4 registers for controlling and storage of result.

1. ADCON0 – ADC Control Register 0
2. ADCON1 – ADC Control Register 1
3. ADRESH – ADC Result High Register
4. ADRESL – ADC Result Low Register

A/D Control Register 0 (ADCON0)

The ADCON0 register, shown in the below image, controls the operation of the A/D module i.e. Used to Turn ON the ADC, Select the Sampling Freq, and also Start the conversion.

ADCON0 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

ADCS1-ADCS0: A/D Conversion Clock Select bits. These bits are based on ADCON1 Reister's ADCS2 bit.

CHS2-CHS0: Analog Channel Select bits

000 = Channel 0 (AN0)

001 = Channel 1 (AN1)

010 = Channel 2 (AN2)

011 = Channel 3 (AN3)

100 = Channel 4 (AN4)

101 = Channel 5 (AN5)

110 = Channel 6 (AN6)

111 = Channel 7 (AN7)

GO/DONE: A/D Conversion Status bit

When ADON = 1:

1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)

0 = A/D conversion not in progress

ADON: A/D On bit

1 = A/D converter module is powered up

0 = A/D converter module is shut-off and consumes no operating current

A/D Control Register 1 (ADCON1)

The ADCON1 register, shown below, configures the functions of the port pins i.e Used to configure the GPIO pins for ADC. The port pins can be configured as analog inputs (RA3 can also be the voltage reference) or as digital I/O.

ADCON1 REGISTER

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Steps :

1. Initialize the ADC Module: We have already learnt how to initialize an ADC so we just call this below function to initialize the ADC

The void ADC_Initialize() function is be as follows.

```
void ADC_Initialize()
{
  ADCON0 = 0b01000001; //ADC ON and Fosc/16 is selected
  ADCON1 = 0b11000000; // Internal reference voltage is selected
}
```

2. Select the analog channel: Now we have to select which channel we are going to use to read the ADC value. Lets make a function for this, so that it will be easy for us to shift between each channel inside the *while* loop.

```
unsigned int ADC_Read(unsigned char channel)
{
    //****Selecting the channel**///
  ADCON0 &= 0x11000101; //Clearing the Channel Selection Bits
  ADCON0 |= channel<<3; //Setting the required Bits
  //**Channel selection complete***///
}
```

Then channel to be selected is received inside the variable channel. In the line

```
ADCON0 &= 0x1100101;
```

The previous channel selection (if any) is cleared. This is done by using the bitwise and operator "&". The bits 3, 4 and 5 are forced to be 0 while the others are left to be in their previous values.

Then the desired channel is selected by left shifting the channel number thrice and setting the bits using the bitwise or operator "|".

```
ADCON0 |= channel<<3; //Setting the required Bits
```

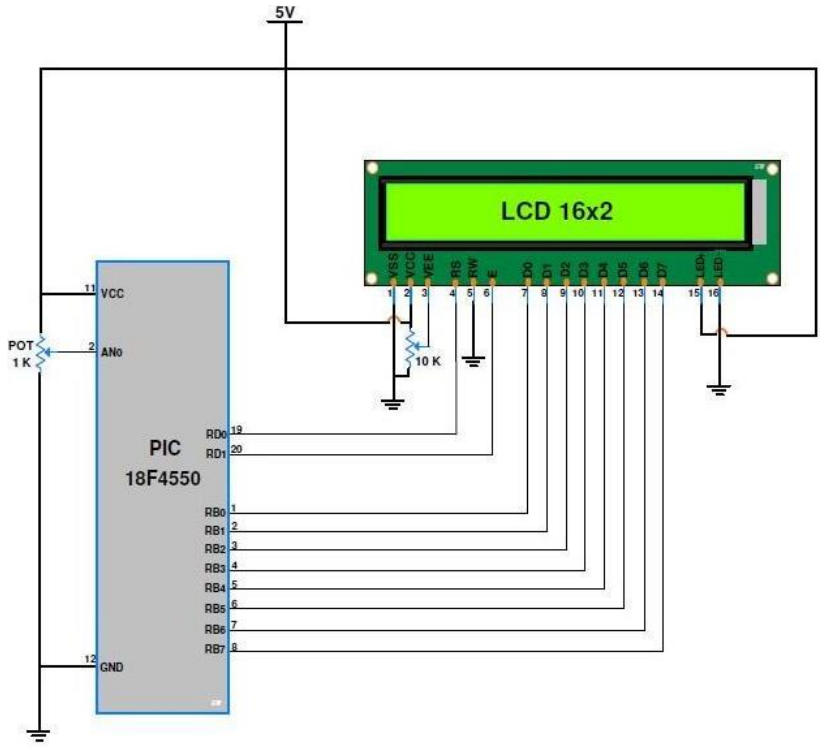
3. Start ADC by making Go/Done bit high: Once the channel is selected we have to start the ADC conversion simply by making the GO_DONE bit high:

```
GO_nDONE = 1; //Initializes A/D Conversion
```

4. Wait for the Go/DONE bit to get low: The GO/DONE bit will stay high until the ADC conversion has been completed, hence we have to wait till this bit goes low again. This can be done by using a *while* loop.

```
while(GO_nDONE); //Wait for A/D Conversion to complete
```

Circuit Diagram:



PROGRAM FOR ANALOG VOLATGE TO INTERNAL ADC & DISPLAY ON LCD:-

```
#include <p18f4550.h>
#include<stdio.h>
#define LCD_EN LATAbits.LA1
#define LCD_RS LATAbits.LA0
#define LCDPORT LATB

void lcd_delay(unsigned int time)
{
 unsigned int i , j ;

   for(i = 0; i < time; i++)
   {
        for(j=0;j<50;j++);
   }
}
void SendInstruction(unsigned char command)
{
    LCD_RS = 0;              // RS low : Instruction
    LCDPORT = command;
    LCD_EN = 1;              // EN High
    lcd_delay(10);
    LCD_EN = 0;              // EN Low; command sampled at EN falling edge
    lcd_delay(10);
}
void SendData(unsigned char lcddata)
{
    LCD_RS = 1;              // RS HIGH : DATA
    LCDPORT = lcddata;
    LCD_EN = 1;              // EN High
    lcd_delay(10);
    LCD_EN = 0;              // EN Low; data sampled at EN falling edge
```

```
    lcd_delay(10);
}

void InitLCD(void)
{
   ADCON1 = 0x0F;
   TRISB = 0x00; //set data port as output
   TRISAbits.RA0 = 0; //RS pin
   TRISAbits.RA1 = 0; // EN pin

   SendInstruction(0x38);      //8 bit mode, 2 line,5x7 dots
   SendInstruction(0x06);       // entry mode
   SendInstruction(0x0C);      //Display ON cursor OFF
   SendInstruction(0x01);      //Clear display
   SendInstruction(0x80);      //set address to 0
}

void ADCInit(void)
{
   TRISEbits.RE1 = 1;              //ADC channel 6 input
   TRISEbits.RE2 = 1;              //ADC channel 7 input

   ADCON1 = 0b00000111;                  //Ref voltages Vdd & Vss; AN0 - AN7 channels
Analog
   ADCON2 = 0b10101110;                //Right justified; Acquisition time 12T; Conversion
clock Fosc/64
}

unsigned short Read_ADC(unsigned char Ch)
{
   ADCON0 = 0b00000001 | (Ch<<2);      //ADC on; Select channel;
   GODONE = 1;            //Start Conversion

   while(GO_DONE == 1 );    //Wait till A/D conversion is complete
   return ADRES;                    //Return ADC result
```

```
}

void DisplayResult(unsigned short ADCVal)
{
 unsigned char i,text[16];
 unsigned short tempv;
 tempv = ADCVal;

 SendInstruction(0x80);              //set to 1st line
 for(i=0;i<10;i++)                   //Display the 10 bit ADC result on LCD
 {
  if(tempv & 0x200)
  {
   SendData('1');
  }
  else
  {
    SendData('0');
  }
  tempv = tempv<<1;
 }

 ADCVal = (5500/1024)*ADCVal;                    //Convert binary data to mV;   1 bit <=>
(5500/1024)mV
 sprintf(text,"ADC value=%4dmv",ADCVal);        //Convert integer data to string

 SendInstruction(0xC0);                 //set to 2nd line
 for(i=0;i<16;i++)                      //Display string on LCD
 {
  SendData(text[i]);
 }
}
void main()
{
   unsigned short Ch_result;
```

```
    TRISB = 0x00;                   //PORTB connected to LCD is output
    ADCInit();
    InitLCD();
 while(1)
    {
       Ch_result = Read_ADC(7);
       DisplayResult(Ch_result);
       lcd_delay(1000);
    }
}
```

CONCLUSION:

Dr. D Y Patil Institute of Technology, Pimpri Pune

Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 9

Aim: Generation of PWM signal for DC motor control.

Software and Hardware used: MPLAB IDE, PIC Loader, PIC18F4550 Trainer Kit

APPLICATION:

In Industries DC motors are using. In some application speed control is required. Here we have assembly of small DC motor with speed control**.**

THEORY:

The direct current (DC) motor is one of the first machines devised to convert electrical power into mechanical power. Permanent magnet (PM) direct current converts electrical energy into mechanical energy through the interaction of two magnetic fields. One field is produced by a permanent magnet assembly; the other field is produced by an electrical current flowing in the motor windings. These two fields result in a torque which tends to rotate the rotor. As the rotor turns, the current in the windings is commutated to produce a continuous torque output. The stationary electromagnetic field of the motor can also be wire-wound like the armature (called a wound-field motor) or can be made up of permanent magnets (called a permanent magnet motor). The purpose of a motor speed controller is to take a signal representing the demanded speed, and to drive a motor at that speed

PWM Signal

Pulse Width Modulation (PWM) is a digital signal which is most commonly used in control circuitry. This signal is set high (5v) and low (0v) in a predefined time and speed. The time during which the signal stays high is called the "on time" and the time during which the signal stays low is called the "off time". There are two important parameters for a PWM as discussed below:

Duty cycle of the PWM:

The percentage of time in which the PWM signal remains HIGH (on time) is called as duty cycle. If the signal is always ON it is in 100% duty cycle and if it is always off it is 0% duty cycle.

Duty Cycle =Turn ON time/ (Turn ON time + Turn OFF time)

PWM using PIC18F4550:

PWM signals can be generated in our PIC Microcontroller by using the **CCP (Compare Capture PWM)** module. The resolution of our PWM signal is 10-bit, that is for a value of 0 there will be a duty cycle of 0% and for a value of 1024 (2^10) there be a duty cycle of 100%. There are two CCP modules in our PIC MCU (CCP1 And CCP2), this means we can generate two PWM signals on two different pins (pin 17 and 16) simultaneously, in our tutorial we are using CCP1 to generate PWM signals on pin 17.

The following registers are used to generate PWM signals using our PIC MCU:

1. CCP1CON (CCP1 control Register)
2. T2CON (Timer 2 Control Register)
3. PR2 (Timer 2 modules Period Register)
4. CCPR1L (CCP Register 1 Low)

Programming PIC to generate PWM signals:

In our program we will read an Analog voltage of 0-5v from a potentiometer and map it to 0-1024 using our ADC module. Then we generate a PWM signal with frequency 5000Hz and vary its duty cycle based on the input Analog voltage. That is 0-1024 will be converted to 0%-100% Duty cycle. This tutorial assumes that you have already learnt to use ADC in PIC if not, read it from here, because we will skip details about it in this tutorial.

So, once the configuration bits are set and program is written to read an Analog value, we can proceed with PWM.

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.

2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the TRISC<2> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)
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, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

The timer module's prescaler is set by making the bit T2CKPS0 as high and T2CKPS1 as low the bit TMR2ON is set to start the timer.

T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits

00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

To set the Frequency of the PWM signal. The value of the frequency has to be written to the PR2 register. The desired frequency can be set by using the below formulae:

```
PWM Period = [(PR2) + 1] * 4 * TOSC * (TMR2 Prescale Value)
```

Rearranging these formulae to get PR2 will give

```
PR2 = (Period / (4 * Tosc * TMR2 Prescale )) - 1
```

Our PWM signal has 10-bit resolution hence this value cannot be stored in a single register since our PIC has only 8-bit data lines. So we have use to other two bits of CCP1CON<5:4> (CCP1X and CCP1Y) to store the last two LSB and then store the remaining 8 bits in the CCPR1L Register.

The PWM duty cycle time can be calculated by using the below formulae:

```
PWM Duty Cycle = (CCPRIL:CCP1CON<5:4>) * Tosc * (TMR2 Prescale Value)
```

Rearranging these formulae to get value of CCPR1L and CCP1CON will give:

```
CCPRIL:CCP1Con<5:4> = PWM Duty Cycle / (Tosc * TMR2 Prescale Value)
```

The value of our ADC will be 0-1024 we need that to be in 0%-100% hence, PWM Duty Cycle = duty/1023. Further to convert this duty cycle into a period of time we have to multiply it with the period (1/ PWM_freq)

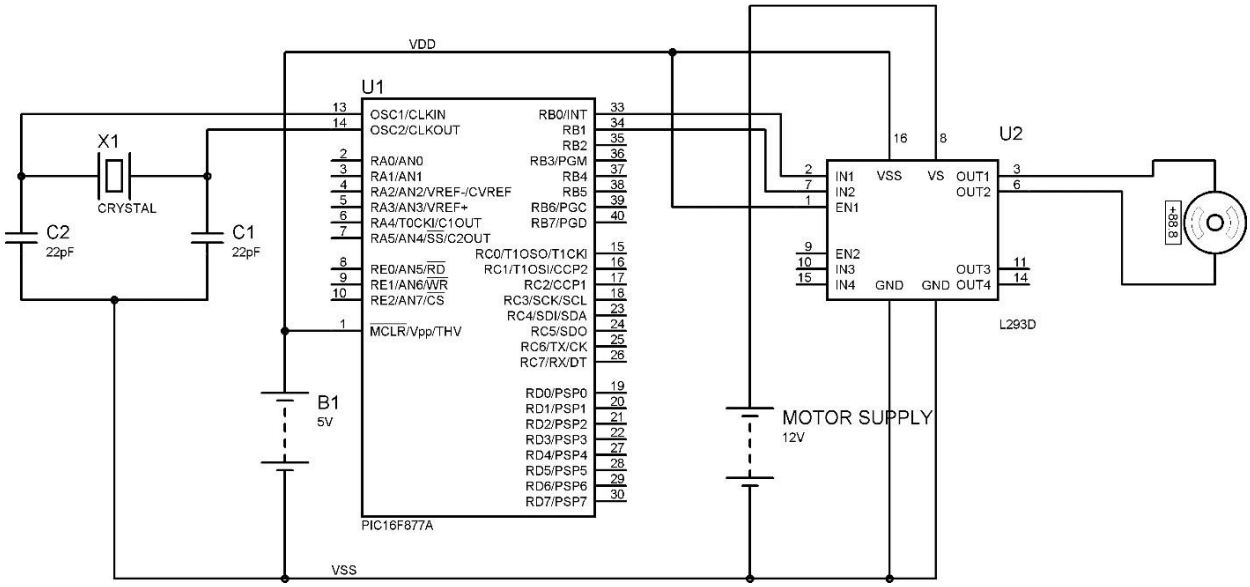
We also know that Tosc = (1/PWM_freq), hence..

```
Duty = ( ( (float)duty/1023) * (1/PWM_freq) ) / ( (1/_XTAL_FREQ) * TMR2PRESCALE) ;
```

Resolving the above equation will give us:

```
Duty = ( (float)duty/1023) * (_XTAL_FREQ / (PWM_freq*TMR2PRESCALE));
```

Circuit Diagram:



/* Calculations

* Fosc = 48MHz

* PWM Period = [(PR2) + 1] * 4 * TMR2 Prescale Value / Fosc

* PWM Period = 200us

* TMR2 Prescale = 16

* Hence, PR2 = 149 or 0x95

* Duty Cycle = 10% of 200us

* Duty Cycle = 20us

* Duty Cycle = (CCPR1L:CCP1CON<5:4>) * TMR2 Prescale Value / Fosc

* CCP1CON<5:4> = <1:1>

* Hence, CCPR1L = 15 or 0x0F

*/

Program:

```
#include<p18f4550.h>
unsigned  char  count=0;
bit TIMER,SPEED_UP;

void timer2Init(void)
{
   T2CON   =   0b00000010;          //Prescalar = 16; Timer2 OFF
   PR2     =   0x95;                //Period Register
}

void delay(unsigned int time)
{
   unsigned int i,j;
   for(i=0;i<time;i++)
      for(j=0;j<1000;j++);
}

void main(void)
{
   unsigned int i;
   TRISCbits.TRISC1    = 0;         //RC1 pin as output
   TRISCbits.TRISC2    = 0;         //CCP1 pin as output
   LATCbits.LATC1      = 0;
   CCP1CON  =   0b00111100;         //Select PWM mode; Duty cycle LSB
CCP1CON<4:5> = <1:1>
   CCPR1L   =   0x0F;               //Duty cycle 10%
   timer2Init();                    //Initialise Timer2
   //Interrupt_Init();              //Initialise interrupts
```

```
    //SPEED_UP = 1;
    TMR2ON = 1;                    //Timer2 ON
    while(1)                       //Loop forever
    {
       for(i=15;i<150;i++)
       {
          CCPR1L = i;
          delay(100);
       }
       for(i=150;i>15;i--)
       {
          CCPR1L = i;
          delay(100);
       }
    }
}
```

Conclusion:

Dr. D Y Patil Institute of Technology, Pimpri Pune
Department of Electronics & Telecommunication Engineering

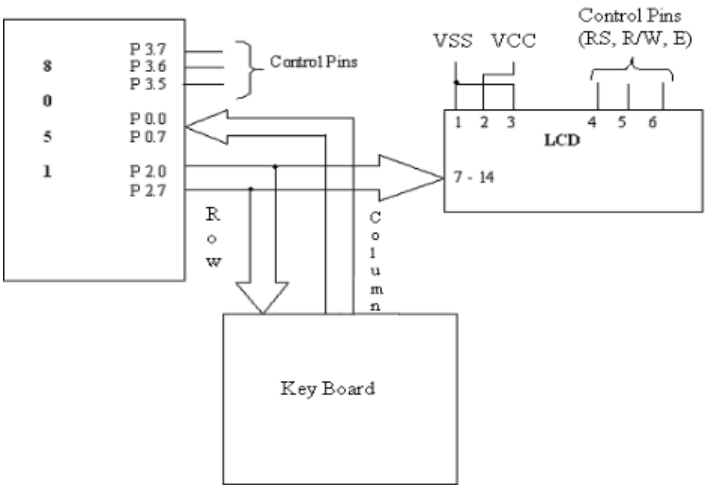
Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 1

LCD and keypad interfacing

Block Diagram:



HEX values of the keys:

LABEL ON THE KEYTOP	HEX CODE	LABEL ON THE KEYTOP	HEX CODE
0	0	-	0C
1	1	*	0D
2	2	/	0E
3	3	%	0F
4	4	AC	10
5	5	CE	11
6	6	CHK	12
7	7	=	13
8	8	MC	14
9	9	MR	15
.	0A	M	16
+	0B	M+	17

Program to interface LCD and KEYPAD :

```
#include <REG51xD2.H>
#include "lcd.h"
unsigned char getkey();
void delay(unsigned int);
main()
{
unsigned char key,tmp;
InitLcd(); //Initialise LCD
WriteString("Key Pressed="); // Display msg on LCD
while(1)
{
GotoXY(12,0); //Set Cursor Position
key = getkey(); //Call Getkey method
}
}
unsigned char getkey()
{
unsigned char i,j,k,indx,t;
P2 = 0x00; //P2 as Output port
indx = 0x00;//Index for storing the 1st value of scanline
for(i=1;i<=4;i<<=1) //for 3 scanlines
{
P2 = 0x0f & ~i; //write data to scanline
t = P0; //Read readlines connected to P0
t = ~t;
if(t>0) //If key press is true
{
delay(6000); //Delay for bouncing
for(j=0;j<=7;j++) //Check for 8 lines
{
t >>=1;
if(t==0) //if get pressed key
{
```

```
k = indx+j; //Display that by converting to Ascii
t = k>>4;
t +=0x30;
WriteChar(t); //Write upper nibble
t = k & 0x0f;
if(t > 9)
t+=0x37;
else
t+=0x30;
WriteChar(t); //write lower nibble
return(indx+j); //Return index of the key pressed
}
}
}
indx += 8; //If no key pressed increment index
}
}
void delay(unsigned int x) //Delay routine
{ for(;x>0;x--); }
```

Dr. D Y Patil Institute of Technology, Pimpri Pune
Department of Electronics & Telecommunication Engineering

Course – Microcontroller

Academic Year 2022-23 - Semester 1

EXPERIMENT NO. - 2

C program to interface Temperature Sensor

Program for temperature sensor

```
# include <at89c51xd2.h>
#include<intrins.h>
#include "lcd.h"
unsigned int Adc;
unsigned char Low_adc,High_adc,relay;
read_adc()
{
unsigned char status;
P2_3 = 1 ; // Start conversion of ADC
status = P1; //Read status of ADC
while((status & 0x01) != 0x01)
{
status = P1;
}
P2_2 = 0; // Enable outputs
P2_0 = 0; // Activate B1 to B8 outputs
Low_adc = P0; // Read lower byte of ADC and place in R0
P2_0 = 1; // Deactivate B1 to B8 outputs
P2_1 = 0; // Activate B9 to B12 and POL, over range outputs
High_adc = P0; // Read higher byte of ADC
High_adc = High_adc & 0x0F;
P2_1 = 1; // deactivate B9 to B12 and
```

```
POL, over range outputs
P2_2 = 1; // Disable outputs
P2_3 = 0; // Stop conversion of ADC
}
main()
{ float Temp,Vol,Res;
unsigned char Temp1;
unsigned char Temp2,Temp3;
P0 = 0xFF ; // Make port 0 as input
P2 = 0xFF ; // Make port 2 as high now the relay is on.
P1_1 = 0 ; // switch OFF relay
P2_3 = 0 ; // STOP conversion of ADC
relay = 10;
while(1)
{
Microcontrollers Lab-10ESL47 2015-16
Dept. of EEE, C.I.T., Gubbi 39
read_adc(); //Read ADC
Adc = High_adc;
Adc <<= 8;
Adc = Adc | Low_adc;
if( (Adc > 0x656) && (relay != 0)) //IF greater than 0x0656 Switch OFF
relay
{
ClrLcd();
WriteString("RELAY OFF");
P1_1 = 0 ;
relay = 0;
}
else if ( (Adc < 0x5b9) && (relay!= 1)) //IF less than 0x05B9 Switch ON
relay
{
ClrLcd();
WriteString("RELAY ON");
P1_1 = 1 ;
```

```
relay = 1;
}
Vol =-((Adc/10)*0.000488); //voltage before amplifier
Res =((100*(1.8-Vol)-100*Vol)*100) /(100*Vol + 100*(1.8+Vol));//
Resistance Value
Res = Res - 100;
Temp = Res/ 0.384;
Temp1 = Temp;
Temp2 = 0x30 + (Temp1 / 0x0A);
Temp3 = 0x30 + (Temp1 % 0x0A);
GotoXY(0,1);
WriteString("Temperature ");
WriteChar(Temp2);
WriteChar(Temp3);
WriteString("'C");
}
}
```