MICROCONTROLLER LABORATORY MANUAL
IV Semester (21EE43)

Name of the Student:	
Semester/Section	:
USN	:
Batch	:

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute Affiliated to VTU, Belagavi)

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' Grade

&

(ISO 9001:2015 Certified)

ELECTRICAL & ELECTRONICS ENGINEERING DEPARTMENT

SHAVIGE MALLESWARA HILLS, KUMARASWAMY LAYOUT

Vision of the Institute

To impart quality technical education with a focus on Research and Innovation emphasizing on Development of Sustainable and Inclusive Technology for the benefit of society.

Mission of the Institute

- To provide an environment that enhances creativity and Innovation in pursuit of Excellence.
- To nurture teamwork in order to transform individuals as responsible leaders and entrepreneurs.
- To train the students to the changing technical scenario and make them to understand the importance of Sustainable and Inclusive technologies.

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute Affiliated to VTU, Belagavi)

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING
BENGALURU-560078

VISION OF THE DEPARTMENT

To meet the challenging needs of society by innovation, problem solving and to develop an exciting and supportive learning environment that transforms our students and inspires them to make a real difference in their career and society.

MISSION OF THE DEPARTMENT

- ✓ **To provide enduring learning environment that facilitates the students to pursue their higher education.**
- ✓ **To train students with diverse skills to work professionally in several fields through innovative teaching and learning process.**
- ✓ **To provide value based and behavioral training programs that helps students in developing their overall professional competence and social awareness.**

PROGRAMME EDUCATIONAL OBJECTIVES [PEOs]

PEO-1: Graduates will have the ability to apply the knowledge of Electrical & Electronics Engineering to excel in their career path.

PEO-2: Graduates will be confident to work in flexible and diverse professional fields.

PEO-3: Graduates will have commitment and awareness to thrive in their ethical and social responsibilities.

PEO-4: Graduates will have the ability to pursue higher education.

PROGRAMME SPECIFIC OUTCOMES [PSOs]

PSO-1:The students will be able to apply the knowledge of mathematics, and applied science principles to solve diverse problems in the field of Power systems and Control of Electric Drives catering to Industrial, Research, Service and allied areas.

PSO-2:The students would be competent in identifying, analyzing and exhibiting the skills in providing solutions to problems related to control electronic systems using various modern tools.

PSO-3: The students will be able to demonstrate proficiency in design and development of different electrical and electronic systems as per specified standards.

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute Affiliated to VTU, Belagavi)

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING
BENGALURU-560078

MICROCONTROLLER LABORATORY MANUAL **(SYLLABUS)**

IV SEMESTER B. E (E & EE)

Sub. Code: 21EE43	**IA Marks: 50**
Hrs/Week: 2	**Exams Hrs: 2**
Total Hrs: 32	**Exam Marks: 50**

Course Objective:

1. To provide familiarity to microcontroller architecture, memory and I/O ports and assembly and C language programming.
2. Demonstrate the usage of instruction set, timers and counters of 8051microcontroller to carry out programming in both polling and interrupt driven environments.
3. To provide the knowledge about the microcontroller interfacing with common peripheral devices.

Course Outcomes:

1. Apply the knowledge to identify the various features of 8051 controller.
2. Analyze problems and illustrate various solutions using 8051 controller.
3. Evaluate various requirements for solving a problem and determine suitable solutions using 8051 controller.
4. Create an embedded system to solve practical problem.

Syllabus

Experiment No.	Contents of the Module	Hours	COs
1	Write an Embedded C program to make LED blink at a given rate (using delay.h as a header file).	02	4
2	Write an Embedded C program to make multiple LEDs blink at a given rate (using delay.h as a header file).	02	4
3	Write an Embedded C program to make LED blink on push of a button (using delay.h as a header file).	02	4
4	Write an Embedded C program to make multiple LEDs blink in different patterns on push of a button (using delay.h as a header file).	02	4
5	Write an Embedded C program to display a character on the Seven Segment Display (using delay.h as a header file).	02	4
6	Write an Embedded C program to control display of a character on the Seven Segment Display using a switch (using delay.h as a header file).	02	4
7	Write an Assembly Language program to make multiple LEDs blink at a given rate using interrupt method.	02	4
8	Write an Assembly Language program using interrupt method to make multiple LEDs blink in different patterns on push of a button.	02	4
9	Write an Assembly Language program to display a character on the Seven Segment Display ((decade UP counter).	02	4
10	Rewrite Embedded C program for Exp. No. 1 to 4 without using header file.	02	4
11	Two LEDs are connected to P2.0 and P2.1 respectively. Make the LED P2.0 blink at a rate of 1 second and the LED connected to P2.1 at the rate of 2 seconds. Write the program in Embedded C using interrupt method.	02	4
12	Interface an LCD with an 8051 and write an Embedded C program to display the message "Hello world."	02	4

DAYANANDA SAGAR COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING
BENGALURU – 560078

MICROCONTROLLER LABORATORY (IV-SEM)

DO's

- DO WEAR UNIFORM & GET YOUR MANNUAL, RECORDS AND ABSERVATION TO ALL THE SESSIONS.
- SAVE ALL THE PROGRAMS IN THE APPROPRIATE DIRECTORY BEFORE SHUTDOWN THE SYSTEM.
- COME PREPARED FOR VIVA & DISCUSS THE PROBLEMS WITH THE STAFF.
- EXECUTE THE PROGRAMS AND NOTE DOWN THE RESULTS IN THE OBSERVATION BOOK.
- RETURN ALL THE INTERFACING KITS SAFELY BEFORE LEAVING THE LAB.
- BEFORE LEAVING THE LAB GET YOUR OBSERVATION BOOK CORRECTED AND SHUTDOWN & COVER THE SYSTEMS.

DONT's

- DO NOT SWITCH ON THE SUPPLY WITHOUT THE PERMISSION OF THE FACULITY.
- DO NOT OPEN THE FILES WHICH ARE NOT RELVENT TO YOUR FILES.
- DO NOT BRING MOBILE PHONES, MUSIC GADGETS TO THE LAB.
- DO NOT COME LATE TO THE LAB MAINTAIN PUNCHUALITY & DISCIPLINE.

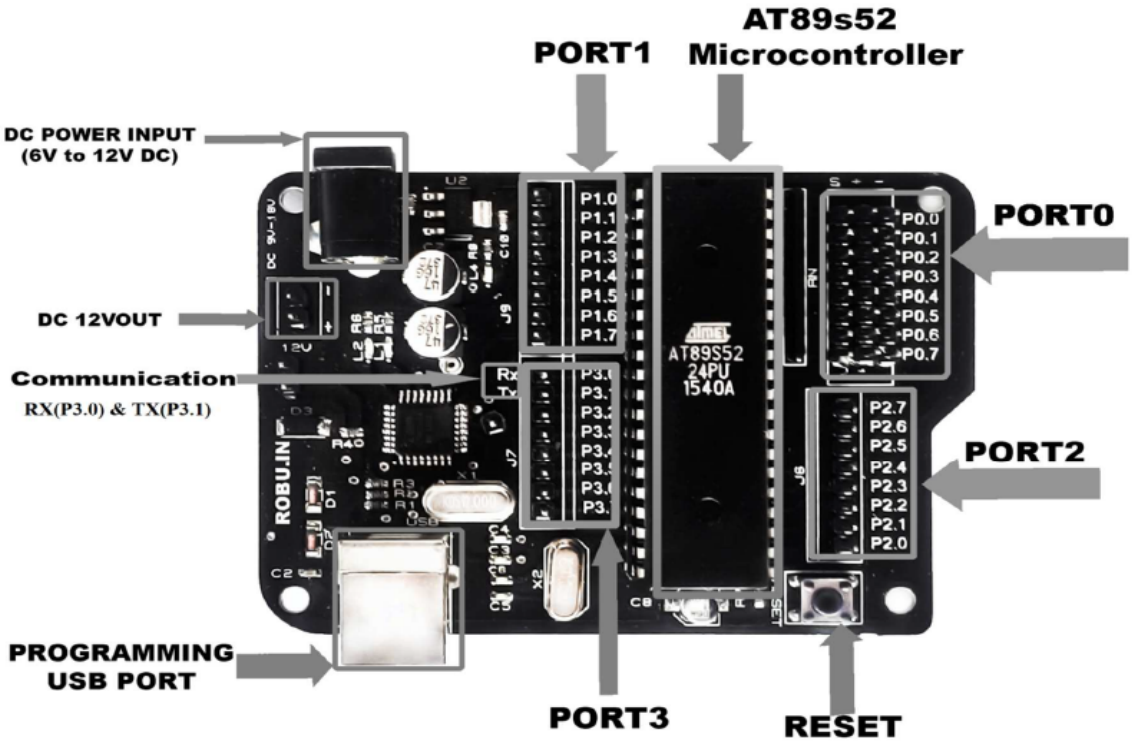
DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute Affiliated to VTU, Belagavi)

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

<u>MICROCONTROLLER LABORATORY (21EE43)</u>

ARYABHATTA 8051 DEVELOPMENT BOARD WITH ON-BOARD USB PROGRAMMER



1. Descriptions:

With this board you can develop and prototype 40 pin 89S52 and 89S51microcontrollers. The on board programmer allows easy connection with PC using USBtype B cable for Programming. The Operating Voltage is 9V to 15V DC.

2. Features:

a) Quartz crystal 11.0592 MHz
b) On board programmer.
c) Reset button.
d) Power plug-in jack.
e) GND bus.
f) VCC bus.
g) On board 5V voltage regulator.
h) Power Indicating LED.
i) On board Regulated Power Supply 5V, 12V, GND.
j) High quality PCB FR4.
k) External pull-up resistors for Port 0.
l) Port extensions for all ports.

3. Specifications:

a) Size: 85 x 62 mm.
b) Supported Microcontroller: AT89SXXXX series.

4. Hardware Details:

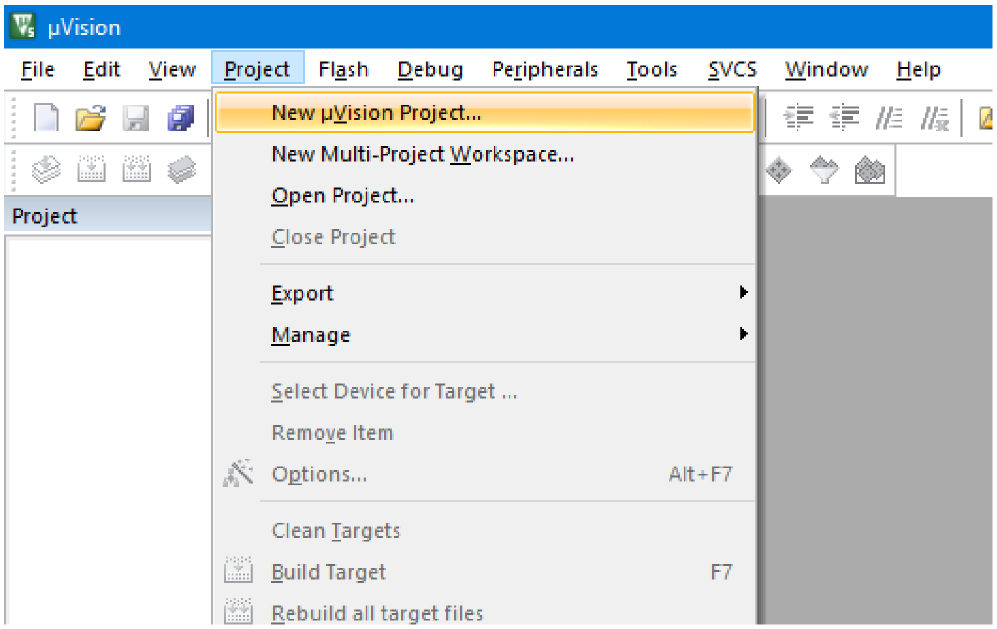
PORT 0(P0.0-P0.7): 8 bit Bidirectional I/O port with external pull-up and having multiplexed low-order address/data bus.

PORT 1(P1.0-P1.7): 8-bit bidirectional I/O port with internal pull-ups.

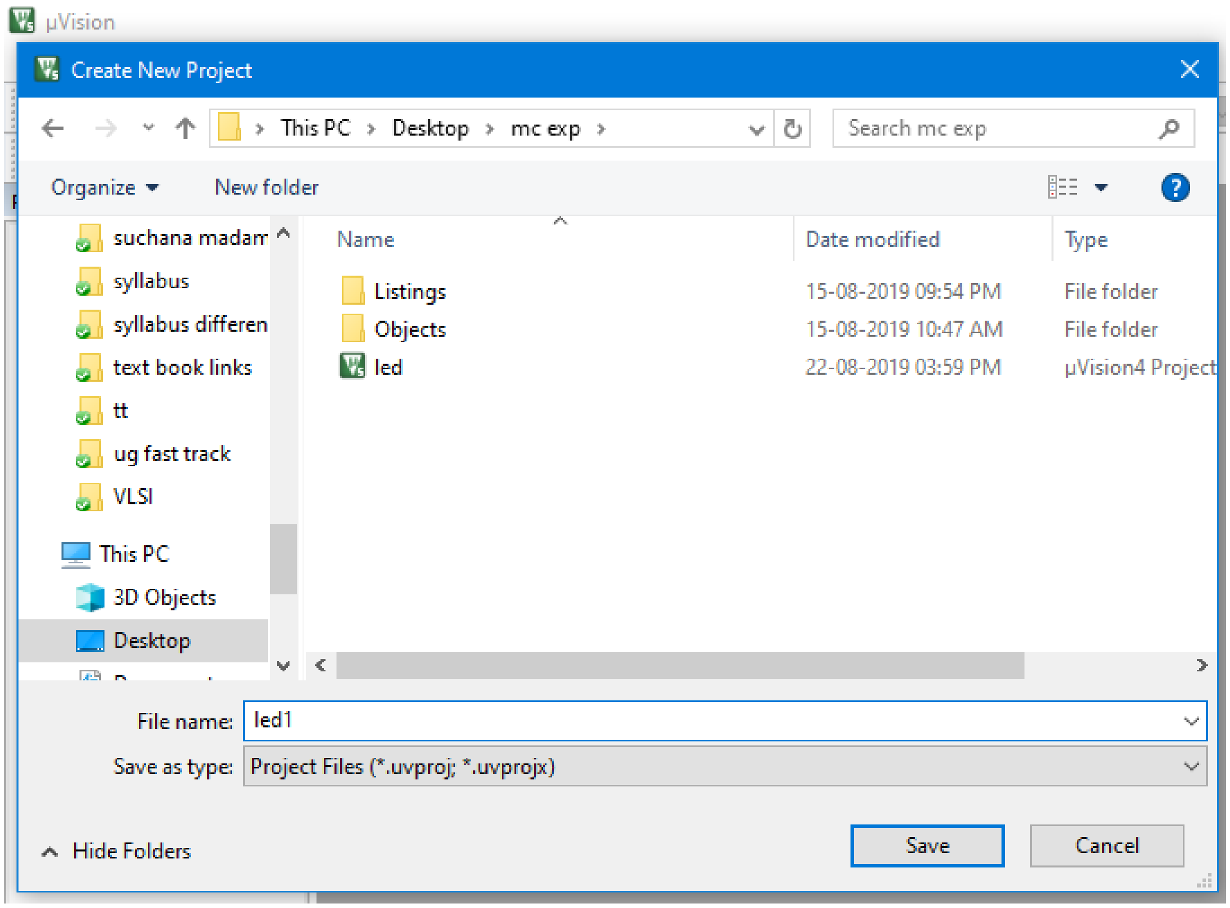
PORT 2(P2.0-P2.7):8 bit Bidirectional I/O port with internal pull-ups and having multiplexed Higher-order address/data bus.

Steps involved in programming Aryabhatta Board:

1. Launch KEIL compiler, create new project:



2. Enter the name of the project without dot extension:

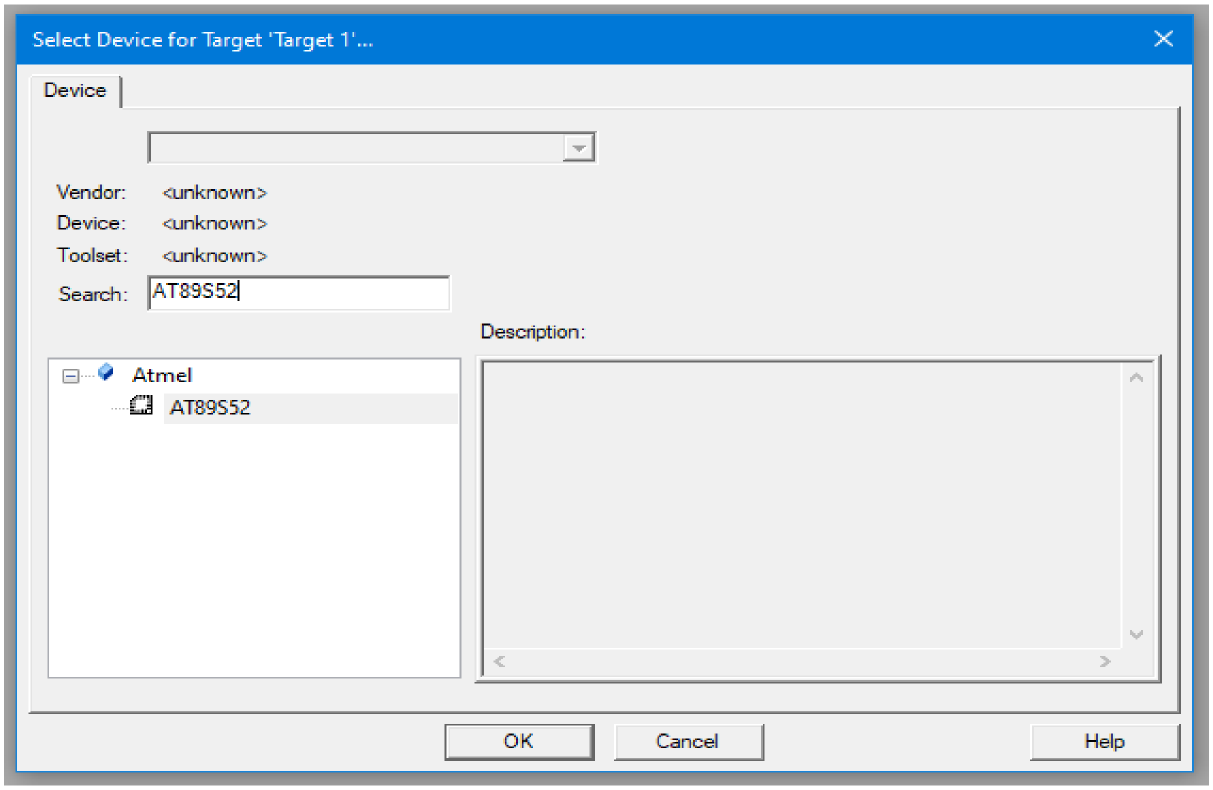


Name	Date modified	Type
Listings	15-08-2019 09:54 PM	File folder
Objects	15-08-2019 10:47 AM	File folder
led	22-08-2019 03:59 PM	µVision4 Project

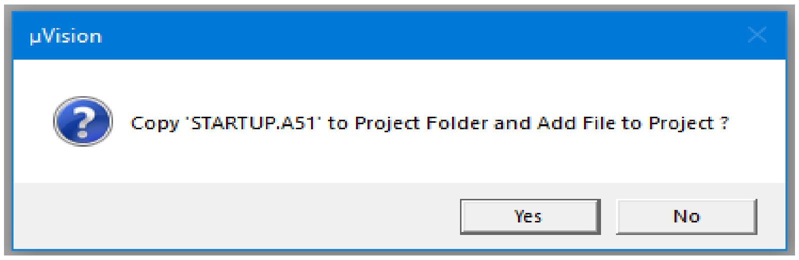
File name: led1

Save as type: Project Files (*.uvproj; *.uvprojx)

3. Select the target device AT89S52:

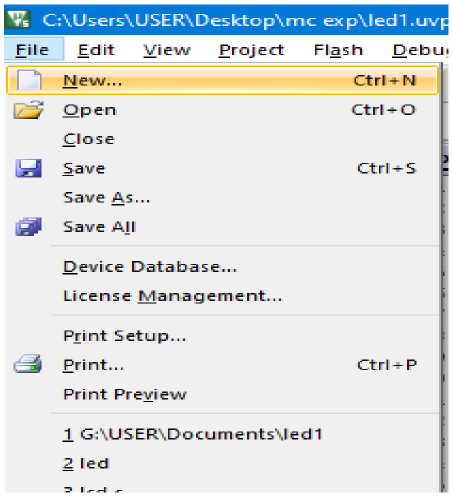


4. Click "No" on the pop up window:

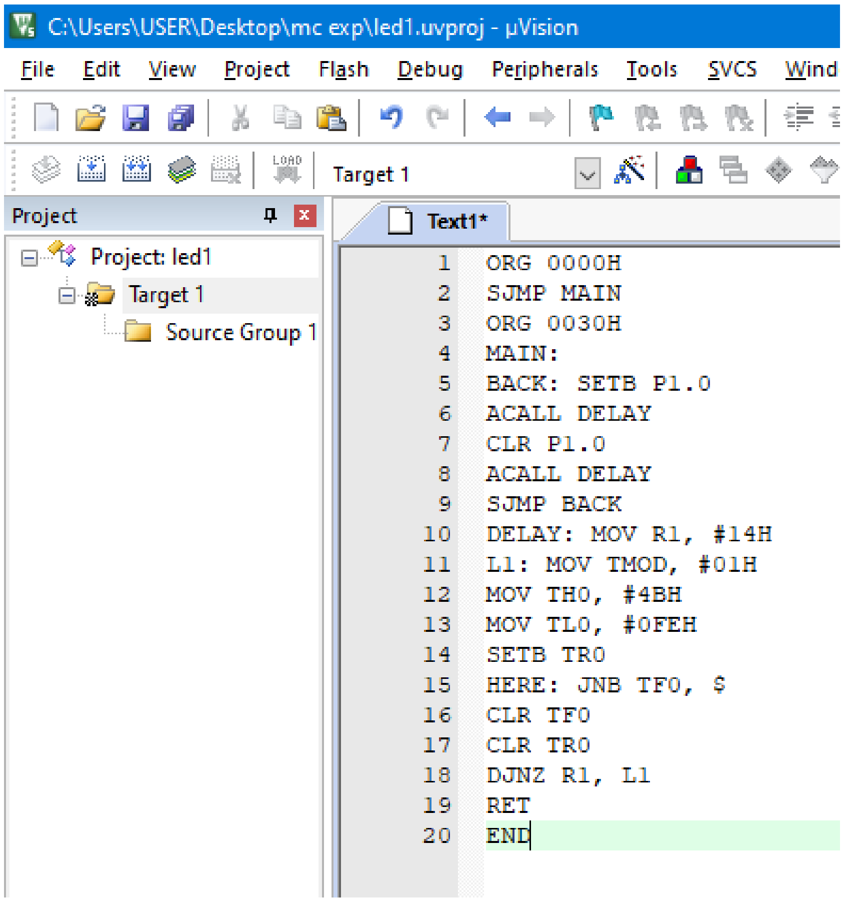


Copy 'STARTUP.A51' to Project Folder and Add File to Project ?

5. Create a new source code:

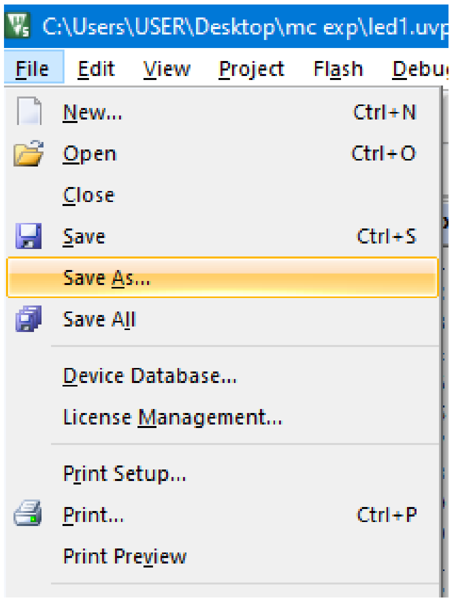


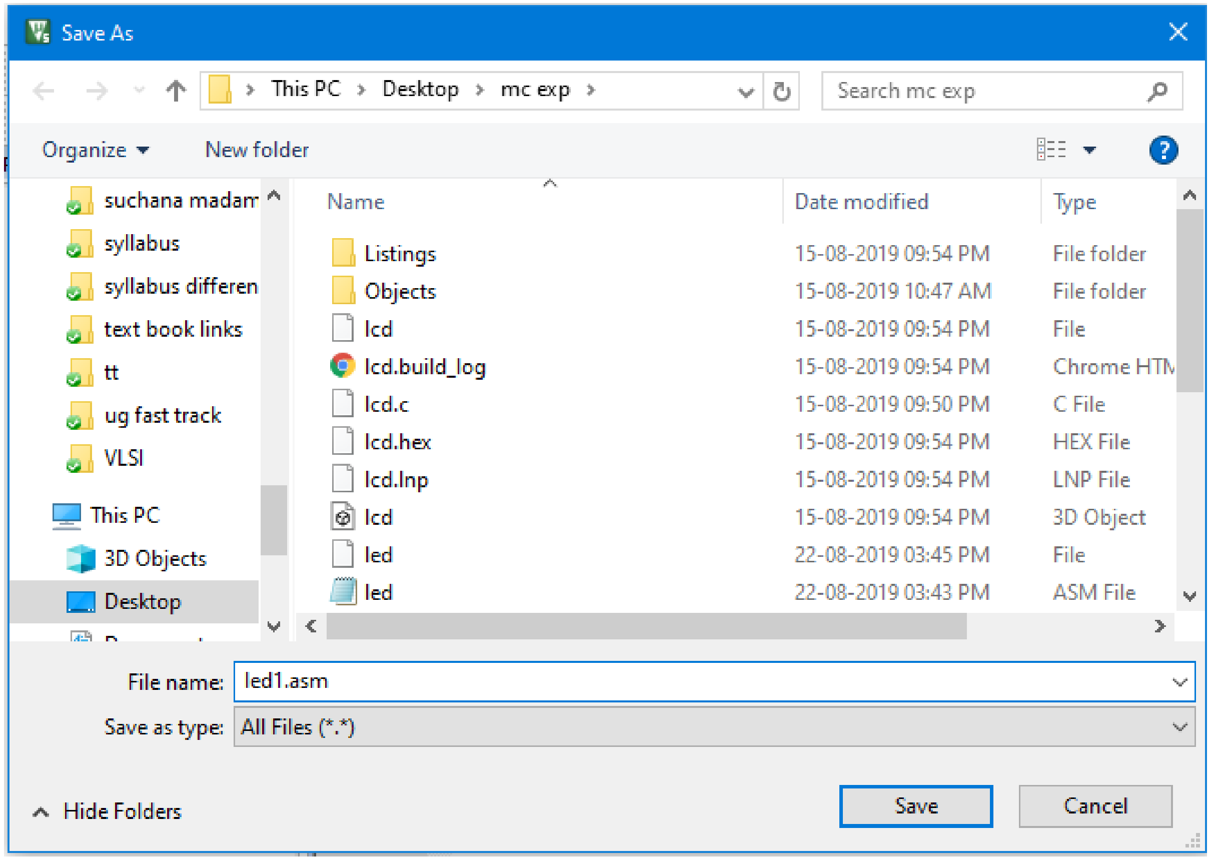
6. Type the program in either assembly language or C language:


C:\Users\USER\Desktop\mc exp\led1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Wind
Target 1
Project
Project: led1
Target 1
Source Group 1
Text1*

```
ORG 0000H
SJMP MAIN
ORG 0030H
MAIN:
BACK: SETB P1.0
ACALL DELAY
CLR P1.0
ACALL DELAY
SJMP BACK
DELAY: MOV R1, #14H
L1: MOV TMOD, #01H
MOV TH0, #4BH
MOV TL0, #0FEH
SETB TR0
HERE: JNB TF0, $
CLR TF0
CLR TR0
DJNZ R1, L1
RET
END
```

7. Save the source code as filename.asm if the program written in assembly or filename.c if the program is written in C language:

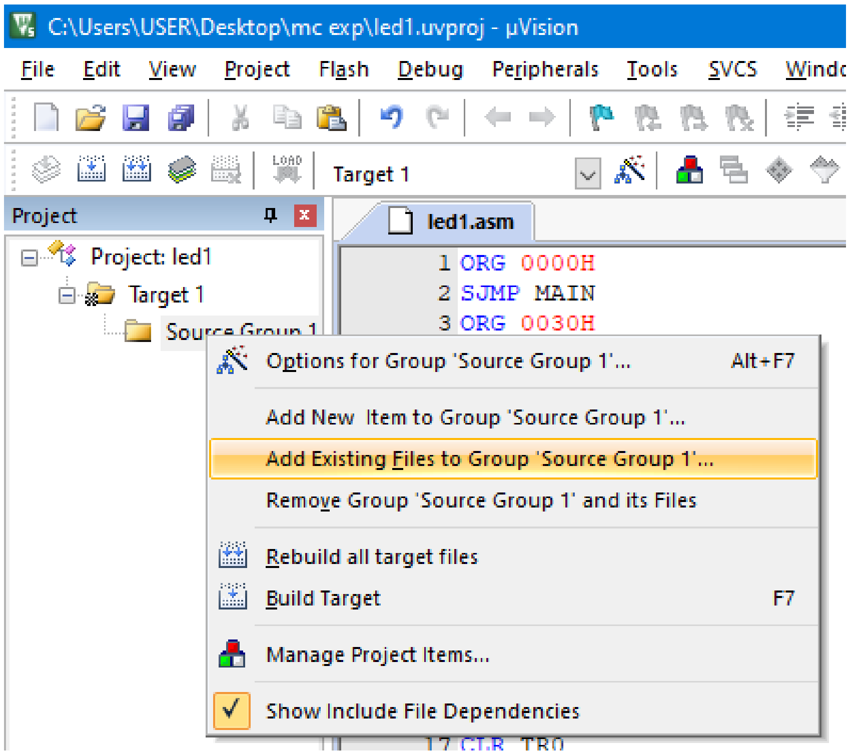




Name	Date modified	Type
Listings	15-08-2019 09:54 PM	File folder
Objects	15-08-2019 10:47 AM	File folder
lcd	15-08-2019 09:54 PM	File
lcd.build_log	15-08-2019 09:54 PM	Chrome HTM
lcd.c	15-08-2019 09:50 PM	C File
lcd.hex	15-08-2019 09:54 PM	HEX File
lcd.lnp	15-08-2019 09:54 PM	LNP File
lcd	15-08-2019 09:54 PM	3D Object
led	22-08-2019 03:45 PM	File
led	22-08-2019 03:43 PM	ASM File

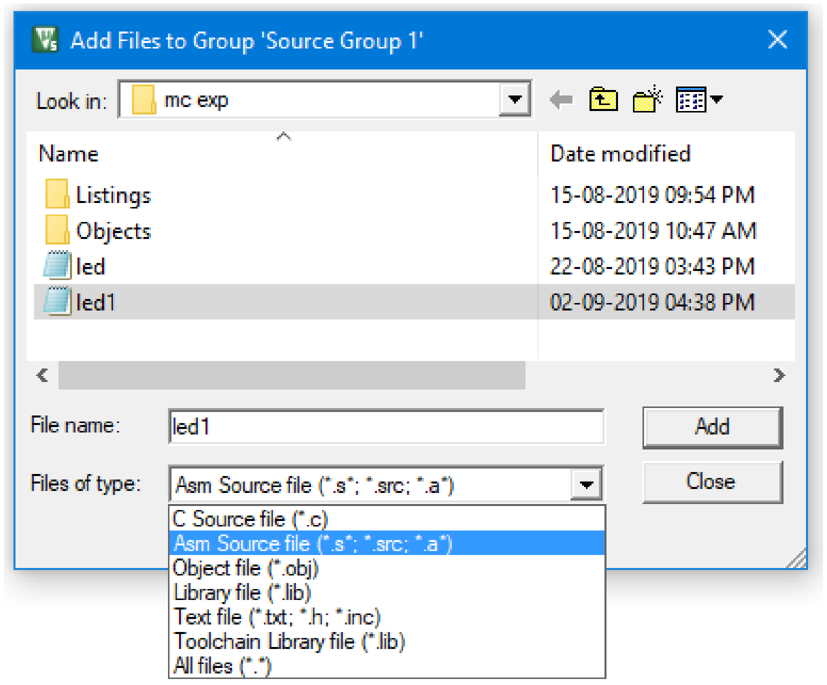
File name: led1.asm
Save as type: All Files (*.*)

8. Add the source code to the project by right clicking on source group 1:

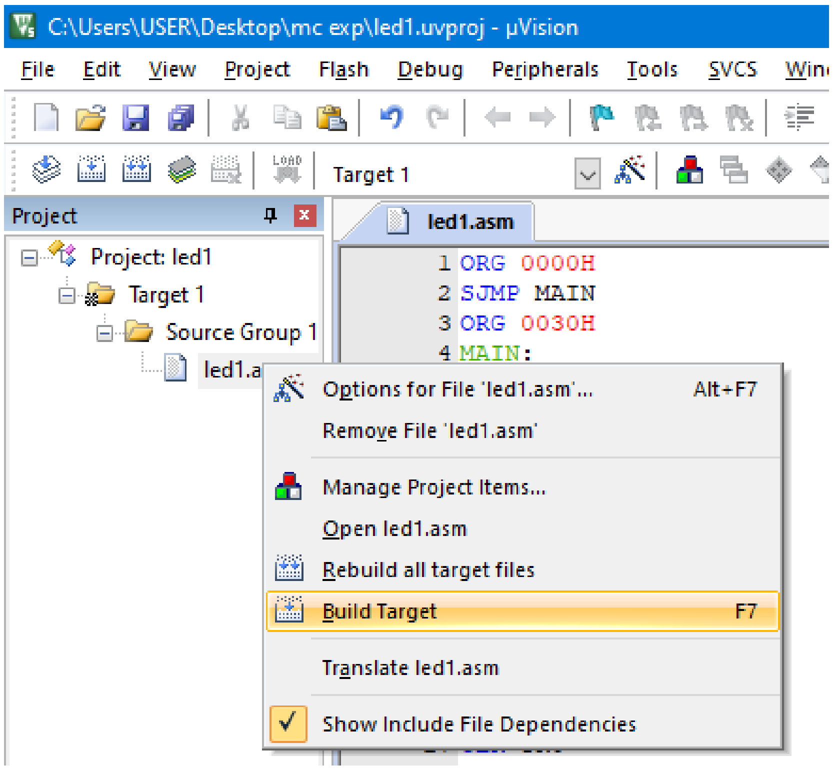

C:\Users\USER\Desktop\mc exp\led1.uvproj - µVision
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Windo
Target 1
Project
Project: led1
Target 1
Source Group 1
led1.asm
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Rebuild all target files
Build Target F7
Manage Project Items...
Show Include File Dependencies

```
ORG 0000H
SJMP MAIN
ORG 0030H
CLR TRO
```

9. Change the files of type to Asm source or C source based on whether the code is written in assembly or C:

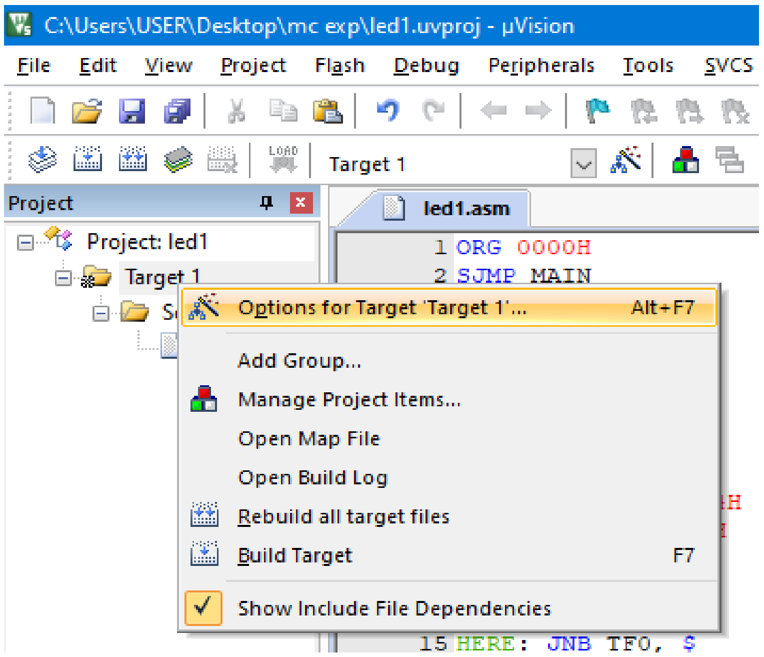

Add Files to Group 'Source Group 1'
Look in: mc exp
Name
Date modified
Listings 15-08-2019 09:54 PM
Objects 15-08-2019 10:47 AM
led 22-08-2019 03:43 PM
led1 02-09-2019 04:38 PM
File name: led1
Files of type: Asm Source file (*.s*; *.src; *.a*)
Add
Close
C Source file (*.c)
Asm Source file (*.s*; *.src; *.a*)
Object file (*.obj)
Library file (*.lib)
Text file (*.txt; *.h; *.inc)
Toolchain Library file (*.lib)
All files (*.*)

10. Right click on source code, click build target and check for errors:



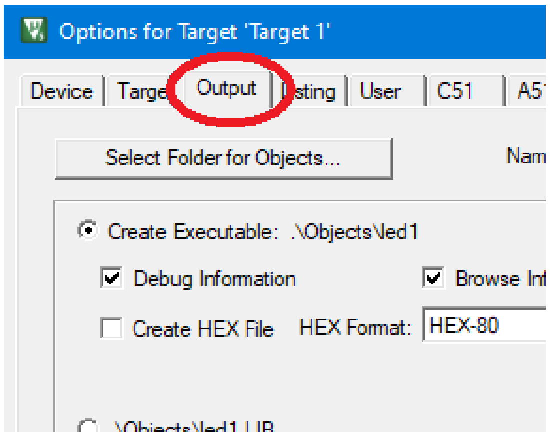
```
ORG 0000H
SJMP MAIN
ORG 0030H
MAIN:
```

11. Right click on Target 1, click on options for target:

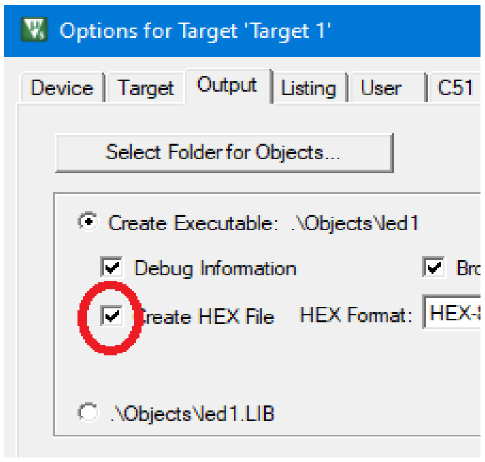

C:\Users\USER\Desktop\mc exp\led1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS
Target 1
Project
Project: led1
Target 1
led1.asm
Options for Target 'Target 1'... Alt+F7
Add Group...
Manage Project Items...
Open Map File
Open Build Log
Rebuild all target files
Build Target F7
Show Include File Dependencies

```
1 ORG 0000H
2 SJMP MAIN
15 HERE: JNB TF0, $
```

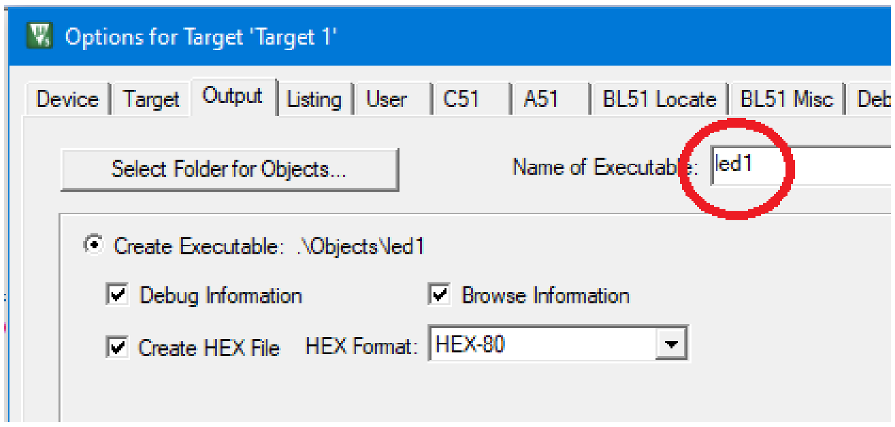
12. Select “Output” tab:



13. Enable Create Hex File:



14. Enter the name of the hex file:



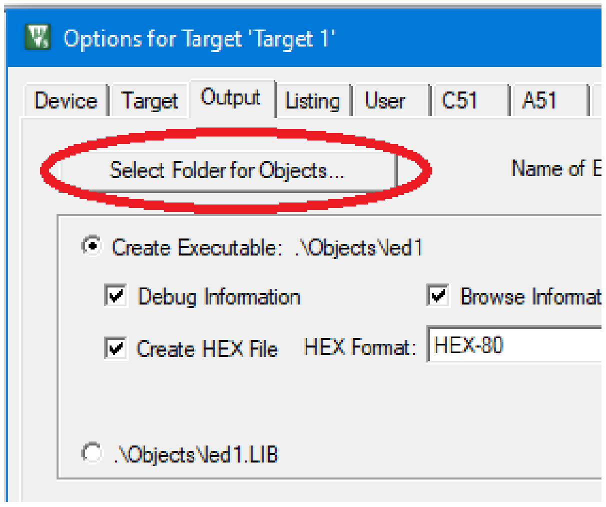
Name of Executable: led1

Create Executable: .\Objects\led1

☑ Debug Information ☑ Browse Information

☑ Create HEX File HEX Format: HEX-80

15. Select the location where the hex file has to be saved by clicking on "Select Folder for Objects":



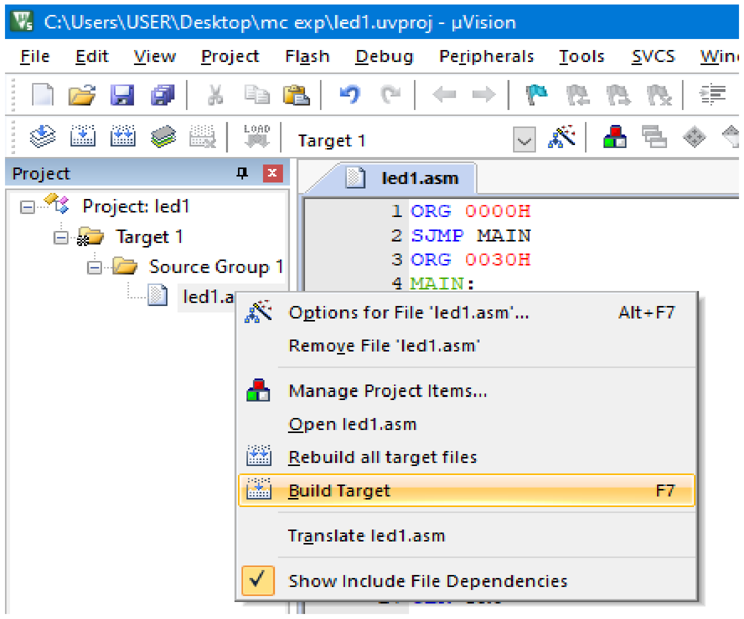
Create Executable: .\Objects\led1

☑ Debug Information ☑ Browse Informat

☑ Create HEX File HEX Format: HEX-80

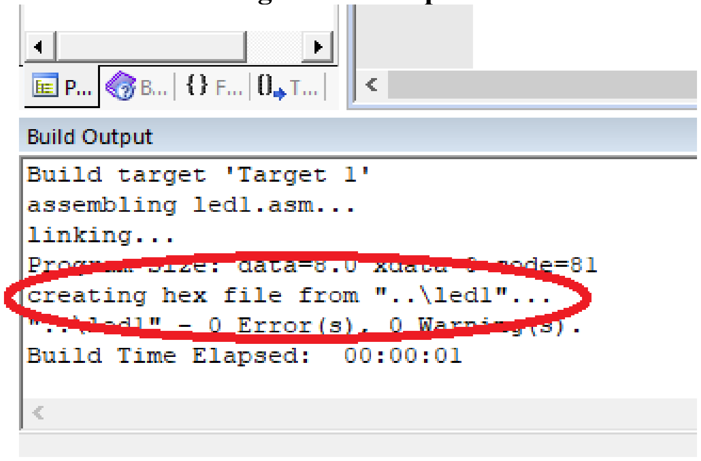
.\Objects\led1.LIB

16. Right click on source code, click build target and check for errors:



```
ORG 0000H
SJMP MAIN
ORG 0030H
MAIN:
```

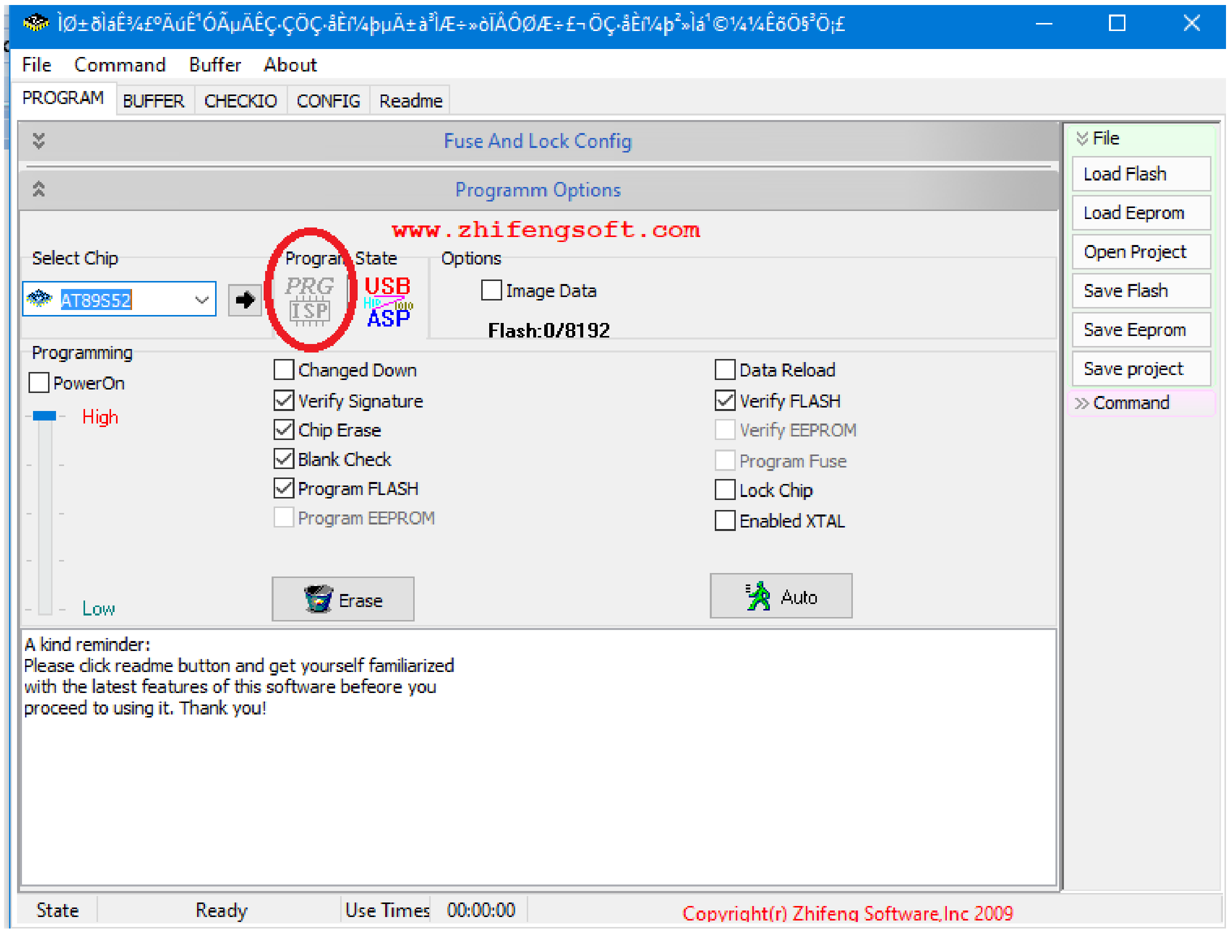
17. Compiler generate hex file at the target location specified:



```
Build target 'Target 1'
assembling led1.asm...
linking...
Program Size: data=8.0 xdata=0 code=81
creating hex file from "..\led1"...
"..\led1" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:01
```

18. Connect Aryabhatta board to system, Launch "progisp" software:

19. Make sure the driver software is installed for Aryabhatta board. If the driver software is not installed, PRGISP will be grey in color:



Select Chip: AT89S52

Program State: PRG ISP, USB ASP

Options: Image Data

Flash:0/8192

Programming: PowerOn, High, Low

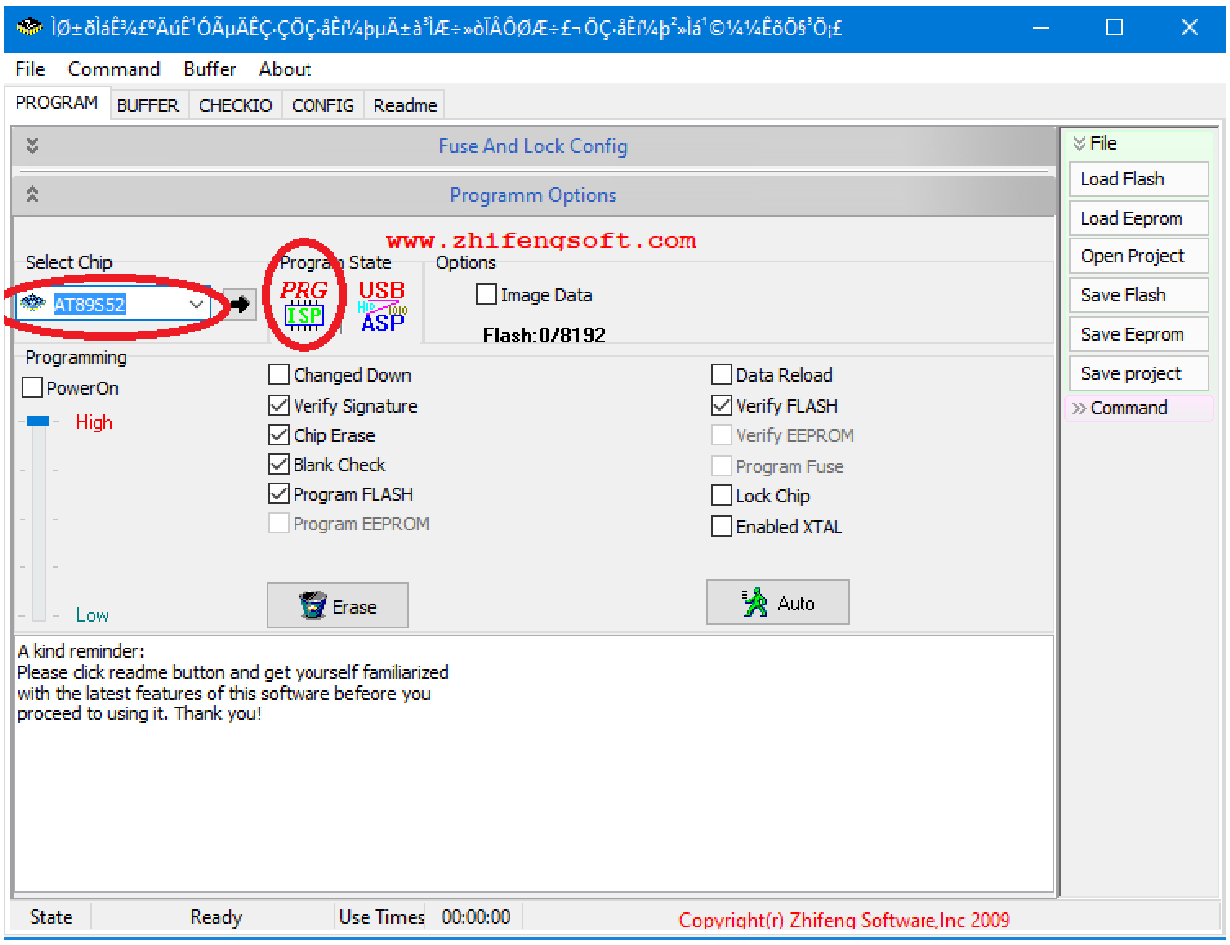
- [ ] Changed Down
- [✓] Verify Signature
- [✓] Chip Erase
- [✓] Blank Check
- [✓] Program FLASH
- [ ] Program EEPROM
- [ ] Data Reload
- [✓] Verify FLASH
- [ ] Verify EEPROM
- [ ] Program Fuse
- [ ] Lock Chip
- [ ] Enabled XTAL

A kind reminder:
Please click readme button and get yourself familiarized with the latest features of this software befeore you proceed to using it. Thank you!

State | Ready | Use Times 00:00:00 | Copyright(r) Zhifeng Software,Inc 2009

Once the driver software is installed the PRGISP turns to green in color and make sure that the chip selected is AT89S52:

:

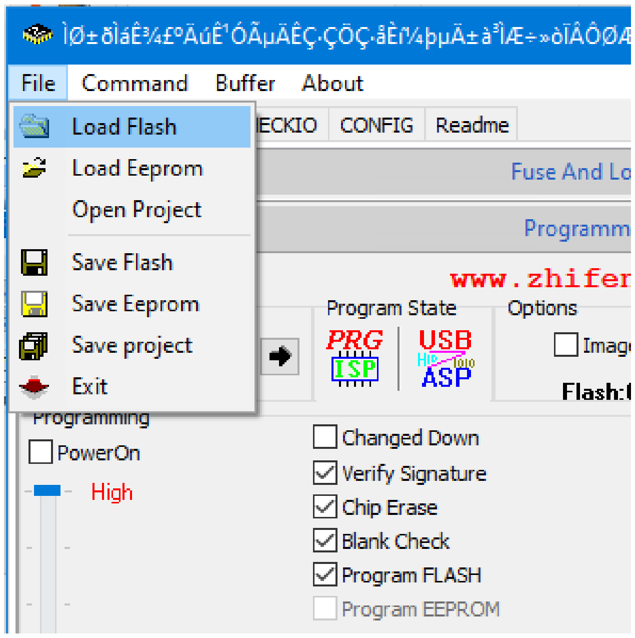


Selected chip: AT89S52

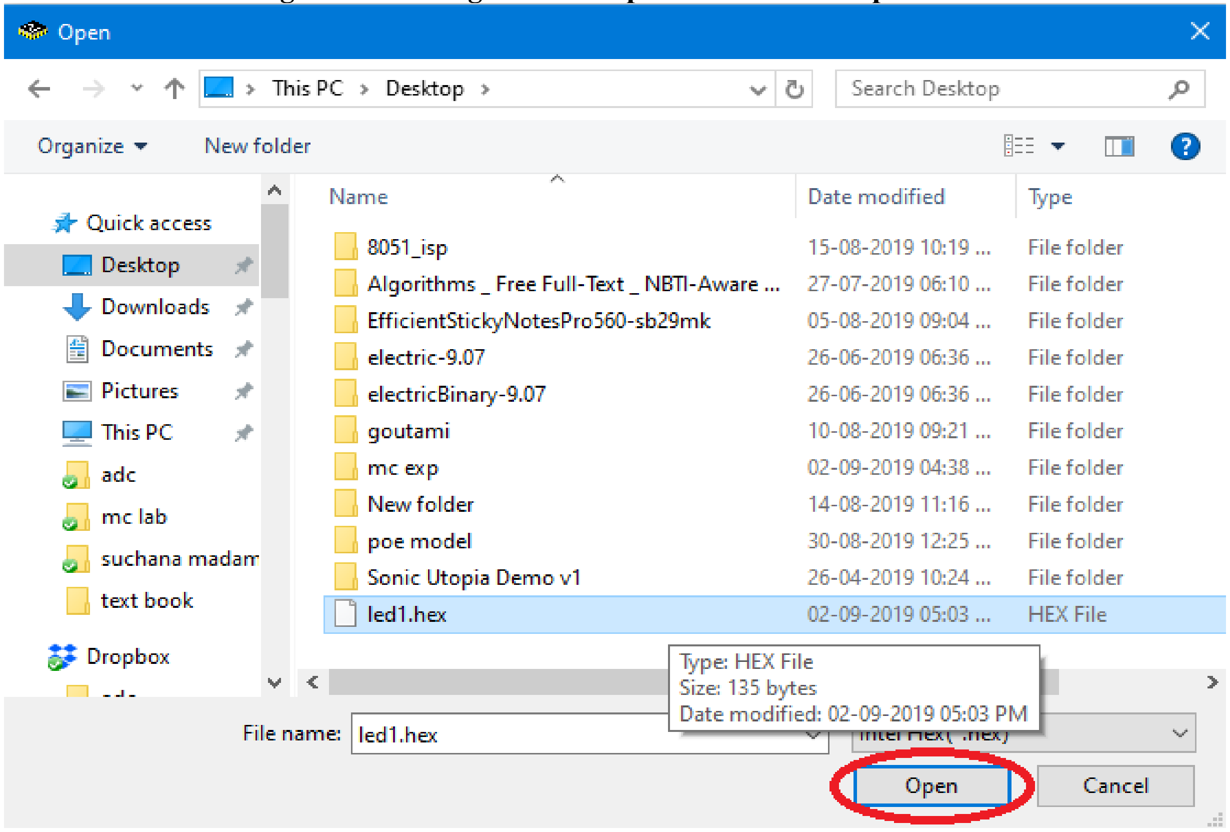
Checked options: Verify Signature, Chip Erase, Blank Check, Program FLASH, Verify FLASH

A kind reminder:
Please click readme button and get yourself familiarized with the latest features of this software beefore you proceed to using it. Thank you!

20. Click on File and click on Load Flash:



21. Select the hex file generated using KEIL compiler and click on open:

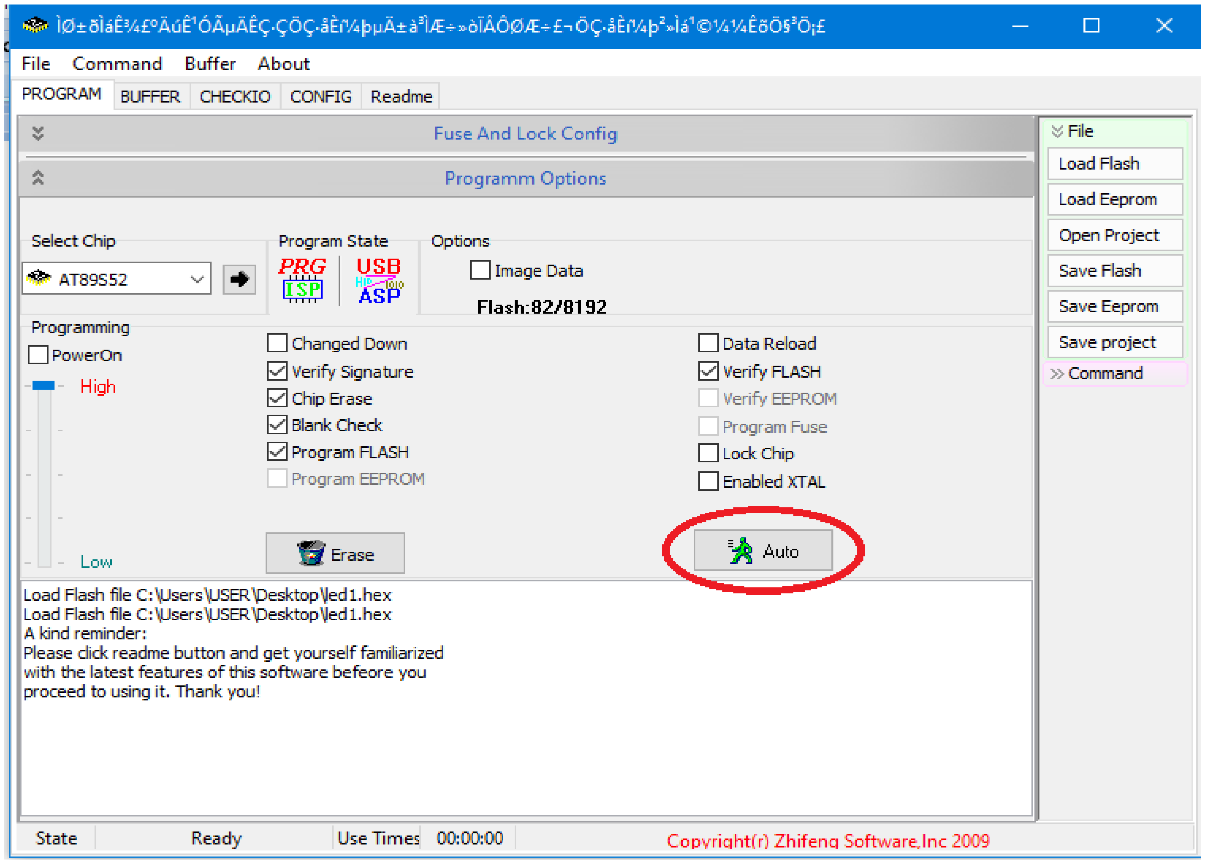


Name	Date modified	Type
8051_isp	15-08-2019 10:19 ...	File folder
Algorithms _ Free Full-Text _ NBTI-Aware ...	27-07-2019 06:10 ...	File folder
EfficientStickyNotesPro560-sb29mk	05-08-2019 09:04 ...	File folder
electric-9.07	26-06-2019 06:36 ...	File folder
electricBinary-9.07	26-06-2019 06:36 ...	File folder
goutami	10-08-2019 09:21 ...	File folder
mc exp	02-09-2019 04:38 ...	File folder
New folder	14-08-2019 11:16 ...	File folder
poe model	30-08-2019 12:25 ...	File folder
Sonic Utopia Demo v1	26-04-2019 10:24 ...	File folder
led1.hex	02-09-2019 05:03 ...	HEX File

Type: HEX File
Size: 135 bytes
Date modified: 02-09-2019 05:03 PM

File name: led1.hex

22. Click on "Auto":



Select Chip: AT89S52

Program State

Options: Image Data

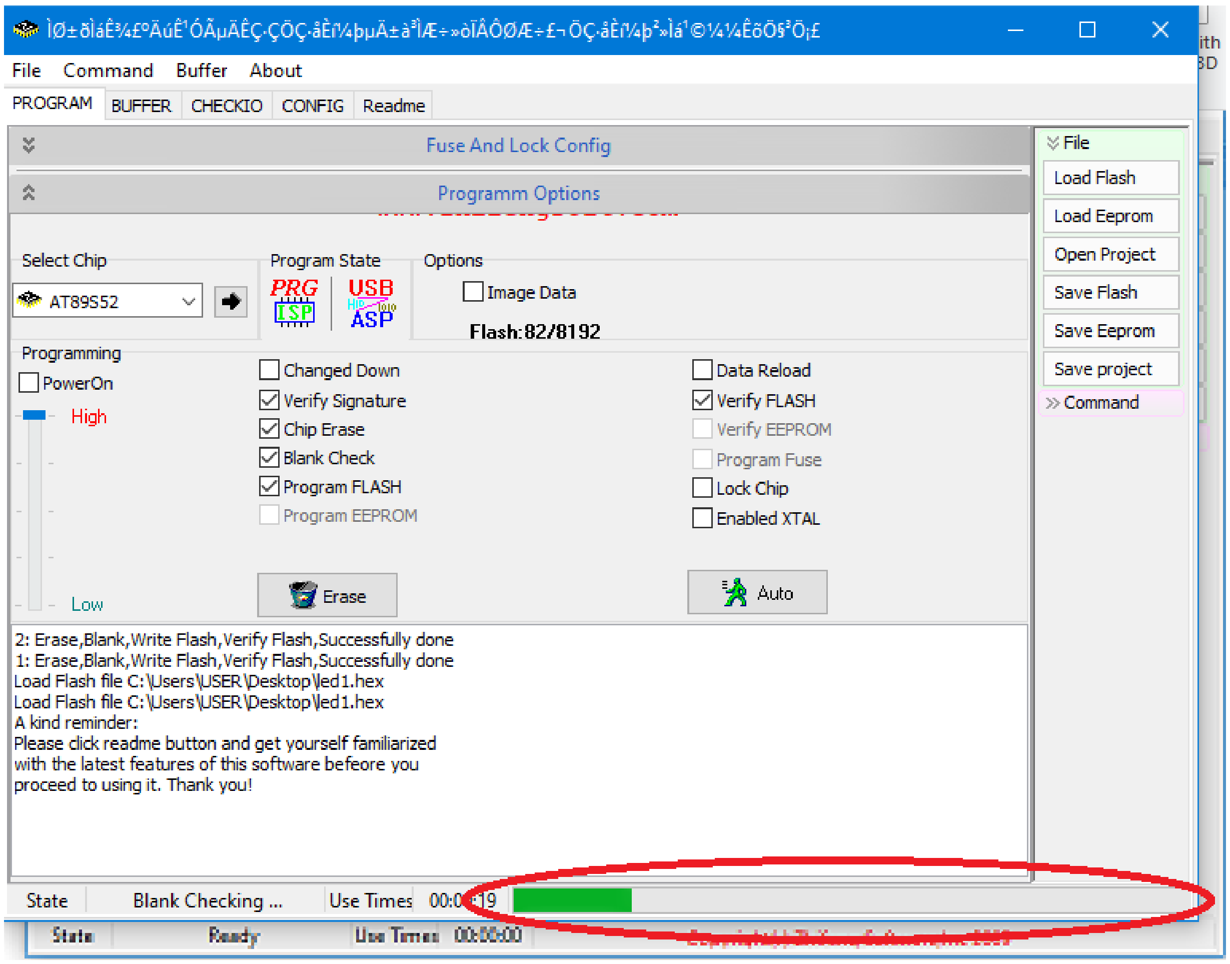
Flash:82/8192

Programming
- [ ] PowerOn
- [ ] Changed Down
- [✓] Verify Signature
- [✓] Chip Erase
- [✓] Blank Check
- [✓] Program FLASH
- [ ] Program EEPROM
- [ ] Data Reload
- [✓] Verify FLASH
- [ ] Verify EEPROM
- [ ] Program Fuse
- [ ] Lock Chip
- [ ] Enabled XTAL

```
Load Flash file C:\Users\USER\Desktop\led1.hex
Load Flash file C:\Users\USER\Desktop\led1.hex
A kind reminder:
Please click readme button and get yourself familiarized
with the latest features of this software beforee you
proceed to using it. Thank you!
```

State Ready Use Times 00:00:00 Copyright(r) Zhifeng Software,Inc 2009

23. Green color progress bar indicates the status of hex code burnt onto the flash memory of microcontroller:



```
2: Erase,Blank,Write Flash,Verify Flash,Successfully done
1: Erase,Blank,Write Flash,Verify Flash,Successfully done
Load Flash file C:\Users\USER\Desktop\led1.hex
Load Flash file C:\Users\USER\Desktop\led1.hex
A kind reminder:
Please click readme button and get yourself familiarized
with the latest features of this software beforre you
proceed to using it. Thank you!
```

Content of Header Files:

Reg51.h:

```
#ifndef __REG51_H__
#define __REG51_H__

/*  SFR BYTE Register  */
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;

/*  PSW   */
sbit CY   = 0xD7;
sbit AC   = 0xD6;
sbit F0   = 0xD5;
sbit RS1  = 0xD4;
sbit RS0  = 0xD3;
sbit OV   = 0xD2;
sbit P    = 0xD0;

/*  TCON  */
sbit TF1  = 0x8F;
sbit TR1  = 0x8E;
sbit TF0  = 0x8D;
sbit TR0  = 0x8C;
sbit IE1  = 0x8B;
sbit IT1  = 0x8A;
sbit IE0  = 0x89;
sbit IT0  = 0x88;

/*  IE   */
sbit EA   = 0xAF;
sbit ES   = 0xAC;
sbit ET1  = 0xAB;
sbit EX1  = 0xAA;
sbit ET0  = 0xA9;
sbit EX0  = 0xA8;

/*  IP   */
sbit PS   = 0xBC;
sbit PT1  = 0xBB;
sbit PX1  = 0xBA;
sbit PT0  = 0xB9;
sbit PX0  = 0xB8;

/*  P3  */
sbit RD   = 0xB7;
sbit WR   = 0xB6;
sbit T1   = 0xB5;
sbit T0   = 0xB4;
sbit INT1 = 0xB3;
sbit INT0 = 0xB2;
sbit TXD  = 0xB1;
sbit RXD  = 0xB0;

/*  SCON  */
sbit SM0  = 0x9F;
sbit SM1  = 0x9E;
sbit SM2  = 0x9D;
sbit REN  = 0x9C;
sbit TB8  = 0x9B;
sbit RB8  = 0x9A;
sbit TI   = 0x99;
sbit RI   = 0x98;

#endif
```

delay.h:

```
void delay(int seconds)
{
unsigned char i=0;
TMOD=0X01;
TH0=0X00;
TL0=0X00;
for(i=0;i<14*seconds;i++)
{
TR0=1;
while(TF0!=1);
TF0=0;
TR0=0;
}
}
```

Experiment No. 1

1. Write an Embedded C program to make LEDs blink at a given rate (using delay.h as a header file).

Circuit diagram:

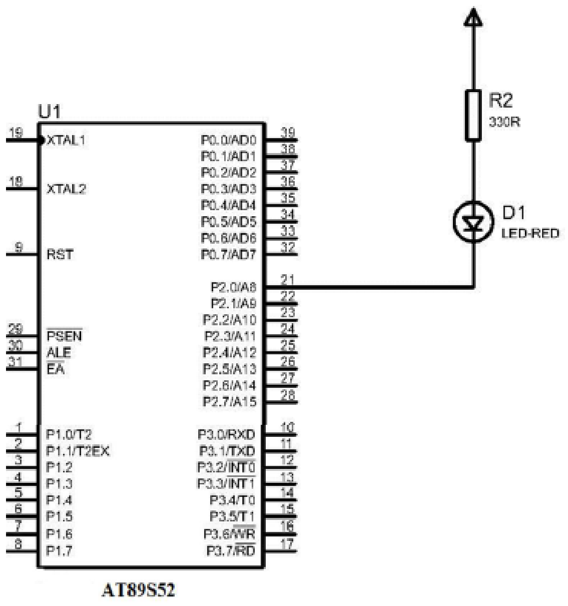


Fig 1. Single LED interfaced to pin P2.0

In Fig 1 the LED's cathode is connected to pin P2.0 and its anode is connected to resistor which in turn is connected to VCC. If P2.0 is reset to ZERO, the LED will be in the ON condition and if the pin is set to ONE, the LED will be in the OFF condition.

Program:

```
#include <REG51.H>
#include <delay.h>
sbit  LED1=P2^0;
void main (void)
{
        while (1)
        {                                           /*Loop forever*/
                LED1 = 0;
                delay(1);
                LED1 = 1;
                delay(1);
        }
}
```

Experiment No. 2

2. Write an Embedded C program to make multiple LEDs blink at a given rate (using delay.h as a header file).

Circuit diagram:

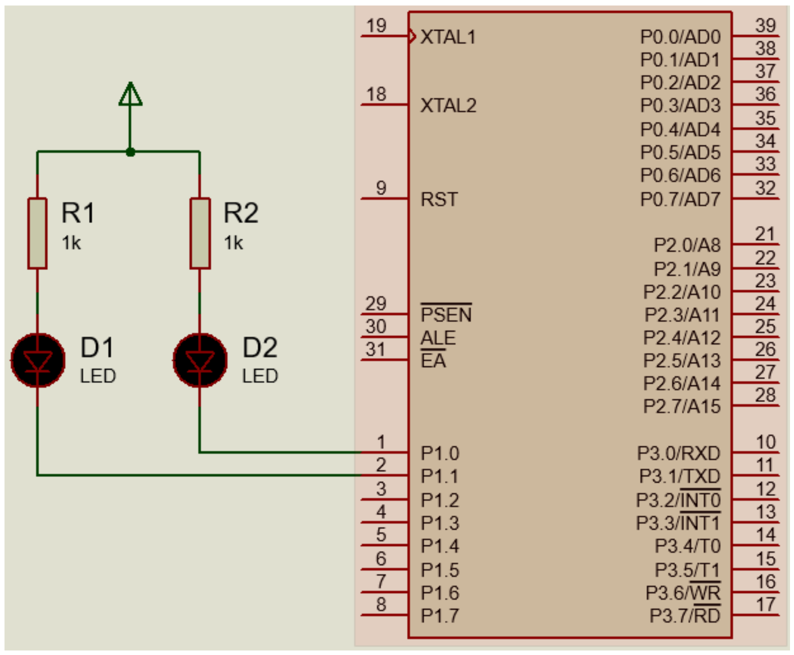


Fig 2. Two LEDs interfaced to pin P1.0 and P1.1

In Fig 2, two LEDs are connected to P1.0 and P1.1. Similar to first experiment, if the pin condition is ZERO, LEDs are in ON condition; otherwise LEDs are in OFF condition.

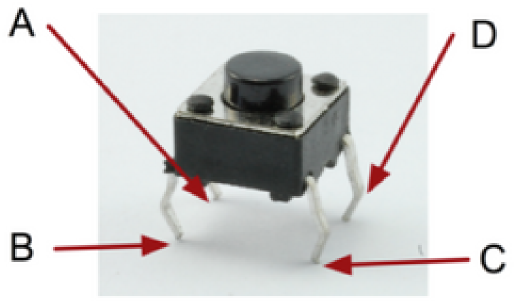
Program:

```
#include <REG51.H>
#include <delay.h>
sbit  LED1=P1^0;
sbit LED2=P1^1;
void main (void)
{
        while (1)
        {                                        /*Loop forever*/
                LED1 = 0;
                LED2=0;
                delay(2);
                LED1 = 1;
                LED2=1;
                delay(2);
        }
}
```

Experiment No. 3

3. Write an Embedded C program to make LED blink on push of a button (using delay.h as a header file).

Push Button Switch and its internal structure:
The Push-button switch as shown in Fig 3 is a basic input device in the embedded system. It is used to control the operation of any output device using the microcontroller or control unit. It basically breaks the electrical circuit and interrupts the flow of current. The push-button is basic mechanical on-off buttons that act as control devices. It short circuits the line when it is pressed and opens when it is not pressed.



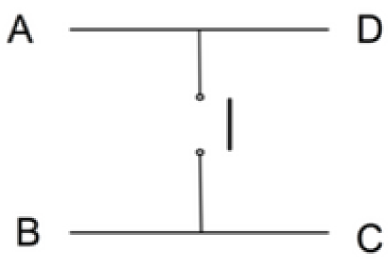
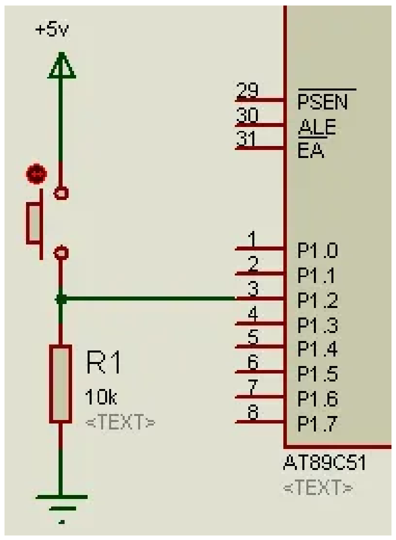


Fig 3. Push button switch

Connection of push-button:
In-circuit Pull-up and Pull-down resistor use to convert infinite or zero resistance into the digital signal. On the basis of the pull-up and pull-down resistor, we can interface the switch in two-way, but the most important point need to remember that the value of pull-up and pull-down resistor depends on the microcontroller.

Positive Logic: In this type of connection (Fig 4), pull-down resistor connected to the ground. When switch is pressed then logic asserts high and when released the switch logic assert low.



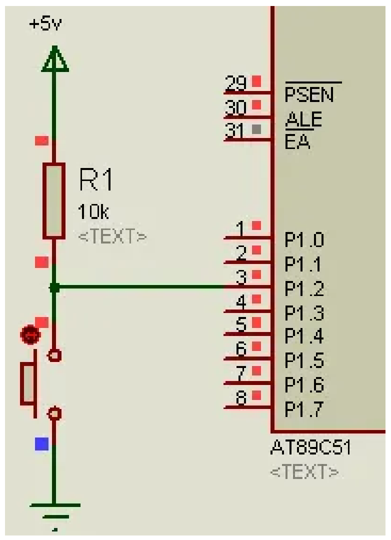


Fig 4. Positive logic connection **Fig 5. Negative logic connection**

Negative Logic: In this type of connection (Fig 5), pull-up resistor connected to VCC. When switch is pressed then logic asserts low and when released the switch logic assert high.

Circuit Diagram:

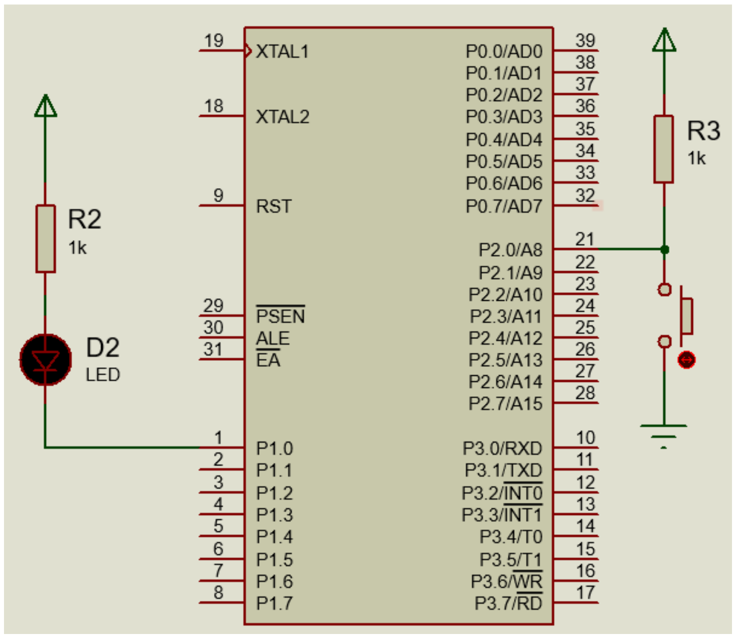


Fig 6. LED and push button switch (negative logic) connected to 8051

In Fig 6 LED is connected to P1.0 and switch is connected to P2.0 in negative logic type. When switch pressed, P2.0 is set to logic ZERO otherwise it is set to logic ONE. The LED will blink as long as switch is pressed and if the switch is released the LED stops blinking.

Program:

```
#include <REG51.H>
#include <delay.h>
sbit  SW1=P2^0;
sbit  LED1=P1^0;
void main (void)
{
        SW1=1;
        LED1=0;                                   /*LED var*/
        while (1)
        {                                         /*Loop forever*/
                if (SW1==0)
                {
                        LED1 = 1;                 /*Output to LED Port*/
                        delay(1);
                        LED1 = 0;
                        delay(1);
                }
        }
}
```

Experiment No. 4

4. Write an Embedded C program to make multiple LEDs blink in different patterns on push of a button (using delay.h as a header file).

Circuit Diagram:

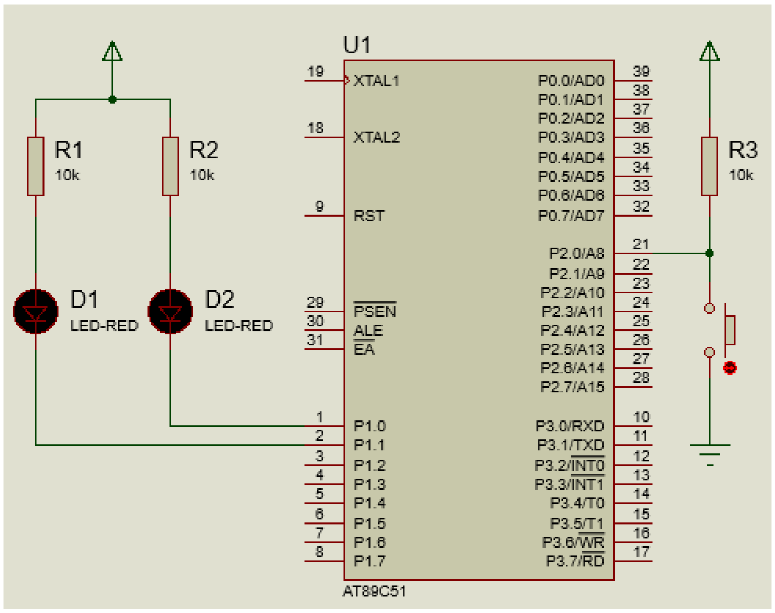


Fig 7. LEDs and push button switch (negative logic) connected to 8051

In Fig 7 two LEDs are connected to P1.0 and P1.1 and switch is connected to P2.0 in negative logic type. When switch pressed, P2.0 is set to logic ZERO and the both the LEDs will blink continuously as long as the switch is pressed. If switch is not pressed, P2.0 is set to logic ONE and LEDs will continue to blink alternatively as long as the switch is not pressed.

Program:

```
#include <REG51.H>
#include <delay.h>
sbit  SW1=P2^0;
sbit  LED1=P1^0;
sbit  LED2=P1^1;
void main (void)
{
        SW1=1;
        LED2=0;                                     /*LED var*/
        while (1)
        {                                           /*Loop forever*/
                if (SW1 == 0)
                {
                        LED1 = 1;                   /*Output to LED Port*/
                        LED2=1;
                        delay(1);
                        LED1 = 0;
                        LED2 = 0;
                        delay(1);
                }
                else
                {
                        LED1 = 1;                               /*Output to LED Port*/
                        LED2=0;
                        delay(1);
                        LED1 = 0;
                        LED2 = 1;
                        delay(1);

                }
        }
}
```

Experiment No. 5

5. Write an Embedded C program to display a character on the Seven Segment Display (using delay.h as a header file).

7-Segment Display:

The 7 segment LED display can display digits from 0 to 9 and quite a few characters like A, b, C, ., H, E, e, F, n, o, t, u, y, etc. A seven segment display consists of seven LEDs arranged in the form of a squarish **'8'** slightly inclined to the right and a single LED as the dot character. Different characters can be displayed by selectively glowing the required LED segments.

Pin Diagram of Seven Segment Display:

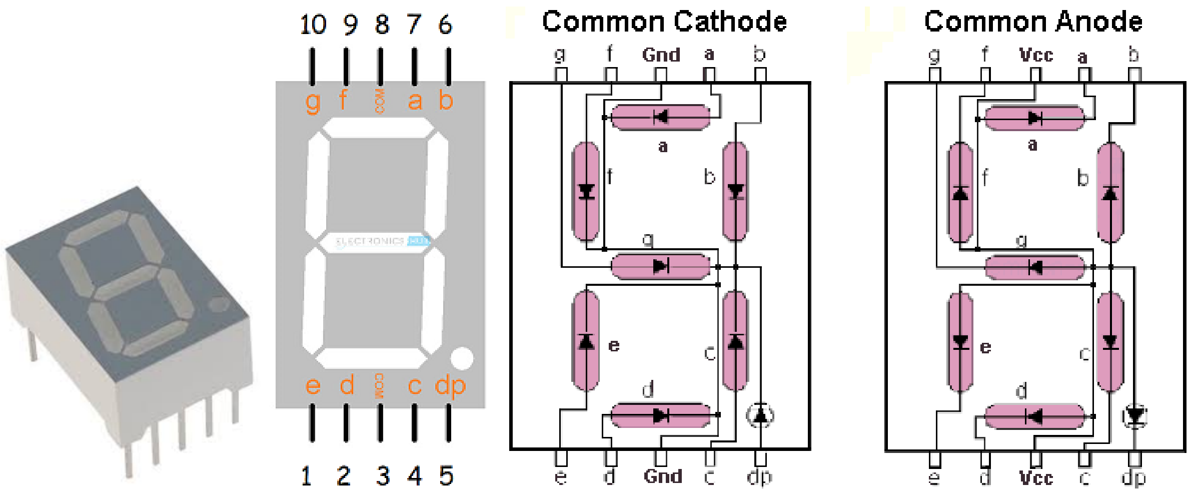


Fig 8. 7-Segment display

Generally, seven segment displays are available in 10 pin package. The pin diagram of seven segment display is shown in Fig 8. Seven segment displays is an electronic circuit consisting of 10 pins.

Out of 10 pins 8 are LED pins and these are left freely. 2 pins in middle are common pins and these are internally shorted. Depending on either the common pin is cathode or anode seven segment displays can be either named as common cathode or common anode display respectively. These are available from different vendors. They have shape of rectangular box similar to that of IC but in large size.

Top and Bottom View of Seven Segment Display:

From top view 8 segments can be seen (7 display segments and one decimal point) in the form of numeral '8'.
Here, the 7 LED's called segments are assigned with an alphabet from A to G. Forward biasing the particular segment or LED will emit the light energy thus illuminating a part of numeral. There is another segment assigned as H, used for displaying dot.
The decimal or dot point is used for representing the decimal point in a numeral. For example, to display 2.5, dot is used to represent the decimal point in this numeral.

Generally, in LED package either all the cathodes or all anodes of the segments are combined to form a common pin. Thus each seven segment display will have seven pins used for displaying the digits, one common pin and another pin for decimal/dot point

Bottom view of the seven segment display is shown below. The bottom view of the segment shows 10 pins of the segment. These are cathode or anode pins of the LEDs present in the seven segment. Seven segment is illuminated using these pins.

Internal Structure of Seven Segment Display:

The internal structure of display is very hard. Internally, the device will have SMD LEDs. This can be divided into two parts i.e. internal circuit and the display. The internal circuit will have LEDs arranged in the rectangular form. These two parts are surrounded by glass, ceramics and plastic in order to protect them.

Working of Seven Segment Display:

Seven segment display works, by glowing the required respective LEDS in the numeral. The display is controlled using pins that are left freely. Forward biasing of these pins in a sequence will display the particular numeral or alphabet. Depending on the type of seven segment the segment pins are applied with logic high or logic zero and in the similar way to the common pins also.

For example, to display numeral '1' segments b and c are to be switched on and the remaining segments are required to be switched off as shown in Fig 9. In order to display two digits two seven segments are used.

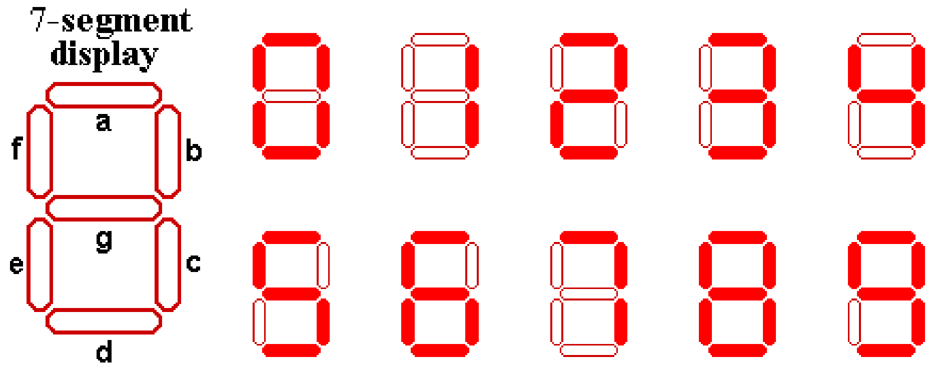


Fig 9. Number display on 7-Segment display

Depending on either the common pin is anode or cathode, seven segments are divided into following types.

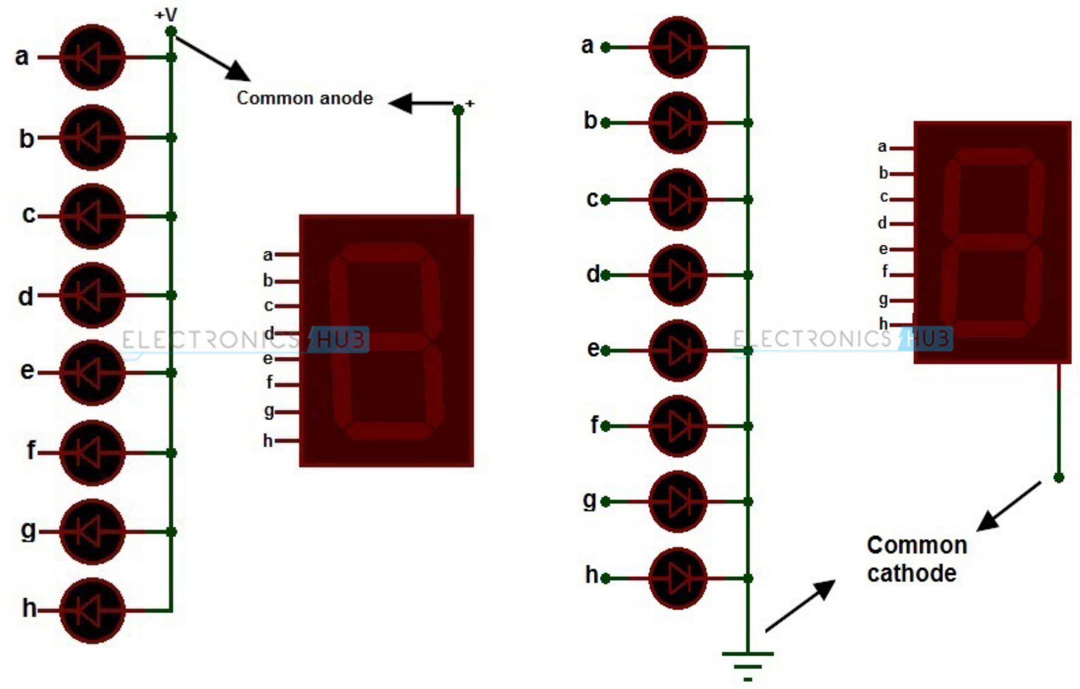
Types of Seven Segment Displays:

The following are the types of seven segments.

1. Common Anode (CA)
2. Common Cathode (CC)

1. Common Anode Seven Segment Display:

In common anode type, all the anodes of 8 LED's are connected to the common terminal and cathodes are left free as shown in Fig 10.a. Thus, in order to glow the LED, these cathodes have to be connected to the logic '0' and anode to the logic '1'.

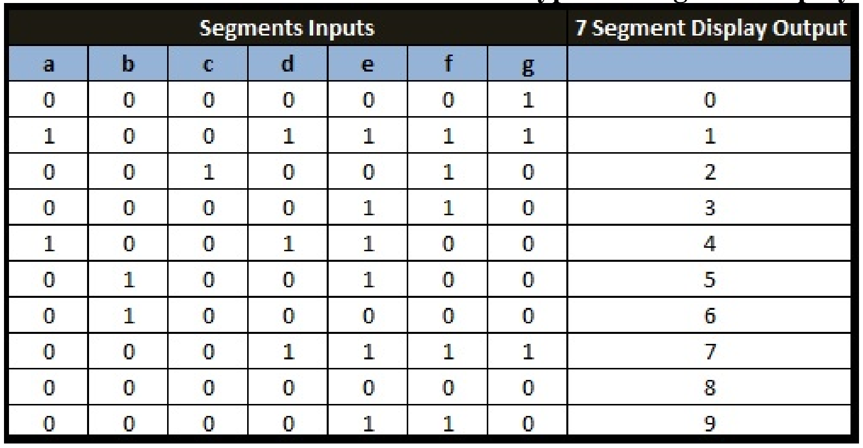


(a) (b)

Fig 10. (a) Common anode type of 7-Segment display, (b) Common cathod type of 7-Segment display

Below truth table, Table 1, gives the information required for driving the common anode type of seven segment display.

Table 1. Truth table for common anode type of 7-segment display



Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	
0	0	0	0	0	0	1	0
1	0	0	1	1	1	1	1
0	0	1	0	0	1	0	2
0	0	0	0	1	1	0	3
1	0	0	1	1	0	0	4
0	1	0	0	1	0	0	5
0	1	0	0	0	0	0	6
0	0	0	1	1	1	1	7
0	0	0	0	0	0	0	8
0	0	0	0	1	1	0	9

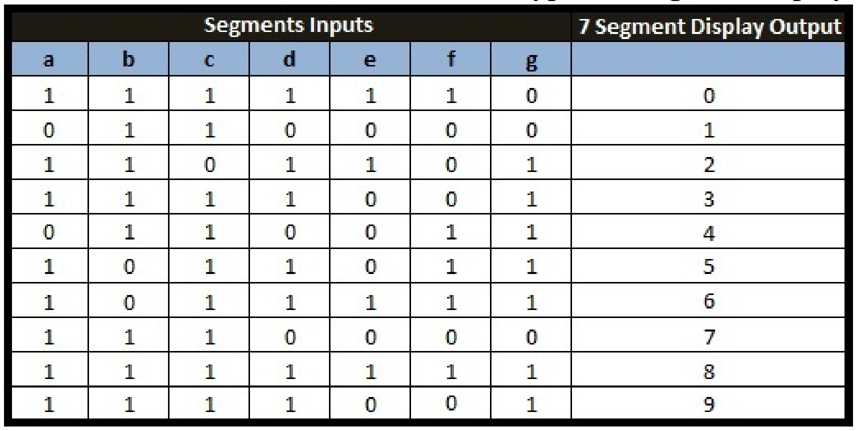
In order to display zero on this segment one should enable logic high on a, b, c, d, e and f segments and logic low on segment 'g'. Thus, the above table provides data on seven segments for displaying numerals from 0-9.

2. Common Cathode Seven Segment Display:

As the name indicates cathode is the common pin for this type of seven segments and remaining 8 pins are left free. Fig 10.b shows common cathod type of 7-Segment display. Here, logic low is applied to the common pin and logic high to the remaining pins. Table 2. shows the truth table of seven segment display. It shows the data to be applied to the seven segments to display the digits.

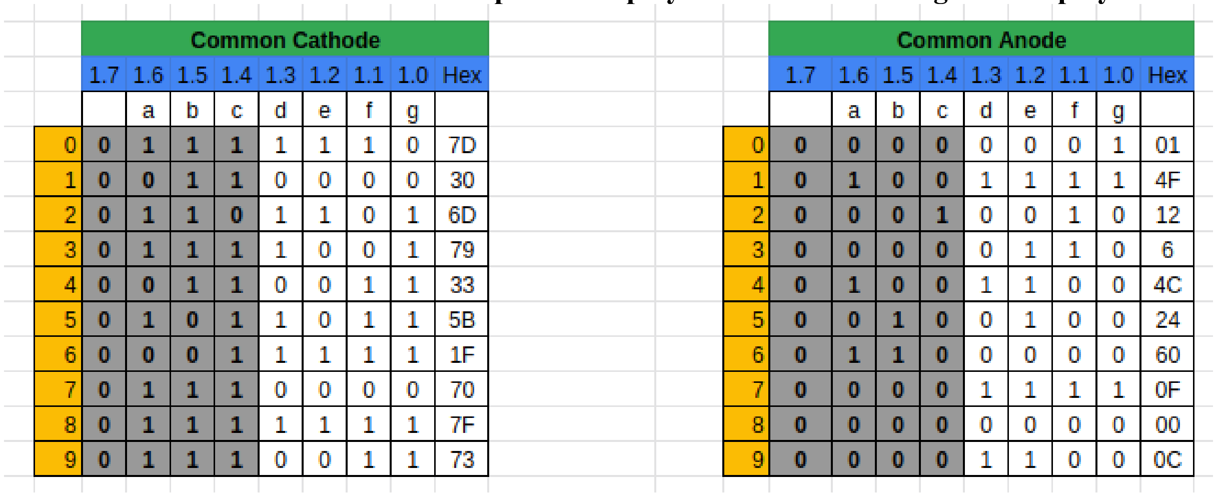
In order to display digit ‘0’ on seven segment, segments a, b, c, d, e and f are applied with logic high and segment g is applied with logic low.

Table 2. Truth table for common anode type of 7-segment display



Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	
1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	2
1	1	1	1	0	0	1	3
0	1	1	0	0	1	1	4
1	0	1	1	0	1	1	5
1	0	1	1	1	1	1	6
1	1	1	0	0	0	0	7
1	1	1	1	1	1	1	8
1	1	1	1	0	0	1	9

Table 3. Value to be loaded to port to display a character on 7-segment display



	Common Cathode								
	1.7	1.6	1.5	1.4	1.3	1.2	1.1	1.0	Hex
		a	b	c	d	e	f	g	
0	0	1	1	1	1	1	1	0	7D
1	0	0	1	1	0	0	0	0	30
2	0	1	1	0	1	1	0	1	6D
3	0	1	1	1	1	0	0	1	79
4	0	0	1	1	0	0	1	1	33
5	0	1	0	1	1	0	1	1	5B
6	0	0	0	1	1	1	1	1	1F
7	0	1	1	1	0	0	0	0	70
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	1	0	0	1	1	73

	Common Anode								
	1.7	1.6	1.5	1.4	1.3	1.2	1.1	1.0	Hex
		a	b	c	d	e	f	g	
0	0	0	0	0	0	0	0	1	01
1	0	1	0	0	1	1	1	1	4F
2	0	0	0	1	0	0	1	0	12
3	0	0	0	0	0	1	1	0	6
4	0	1	0	0	1	1	0	0	4C
5	0	0	1	0	0	1	0	0	24
6	0	1	1	0	0	0	0	0	60
7	0	0	0	0	1	1	1	1	0F
8	0	0	0	0	0	0	0	0	00
9	0	0	0	0	1	1	0	0	0C

Table 3 shows the value to be loaded to the ports which are connected to 7-segment display for both Common Anode type as well as common Cathode type.

<u>7-segment Display Input Value Generation (Common Anode):</u>

(8th Bit)	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	Input Value	Display Character
	g	f	e	d	c	b	a		
0	1	0	0	0	0	0	0	40H	0
0	1	1	1	1	0	0	1	79H	1
0	0	1	0	0	1	0	0	24H	2
0	0	1	1	0	0	0	0	30H	3
0	0	0	1	1	0	0	1	19H	4
0	0	0	1	0	0	1	0	12H	5
0	0	0	0	0	0	1	1	03H	6
0	1	1	1	1	0	0	0	78H	7
0	0	0	0	0	0	0	0	00H	8
0	0	0	1	1	0	0	0	18H	9

Circuit Diagram:

Interfacing seven segment display (Common Anode) to 8051:

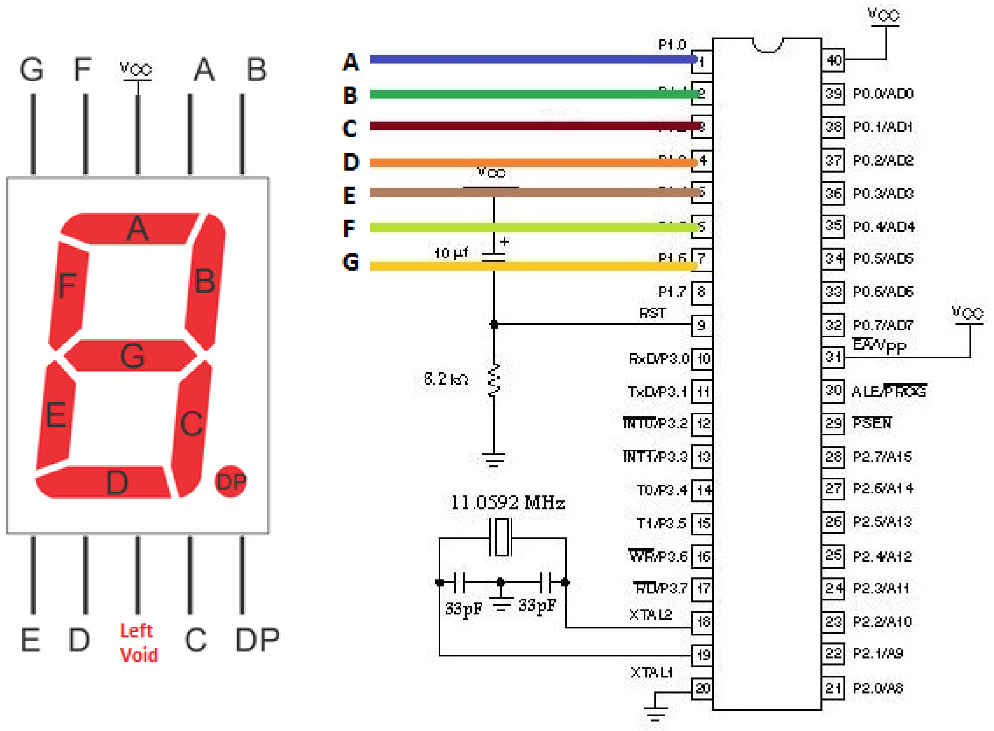


Fig 11. 7-Segment display interface to 8051 microcontroller

Program:

```
#include <REG51.H>
#include <delay.h>
void main (void)
{
        unsigned int i;                                    /*Delay var*/
        unsigned char j=0;
        unsigned int digit[10]={0x40,0x79,0x24,0x30,0x19,0x12,0x03,0x78,
        0x00,0x18};                                              /*LED var*/
        P1=0x00;
        while (1)
        {
                for(j=0;j<10;j++)
                {                                          /*Loop forever*/
                        P1=digit[j];
                        for (i=0; i<10000; i++)
                        {                        /*Delay for 10000 Counts*/
                                delay(1);         /*call wait function*/
                        }
                        for (i=0; i<10000; i++)
                        {                        /*Delay for 10000 Counts*/
                                delay(1);         /*call wait function*/
                        }
                }
        }
}
```

Experiment No. 6

6. Write an Embedded C program to control display of a character on the Seven Segment Display using a switch (using delay.h as a header file).

Circuit Diagram:

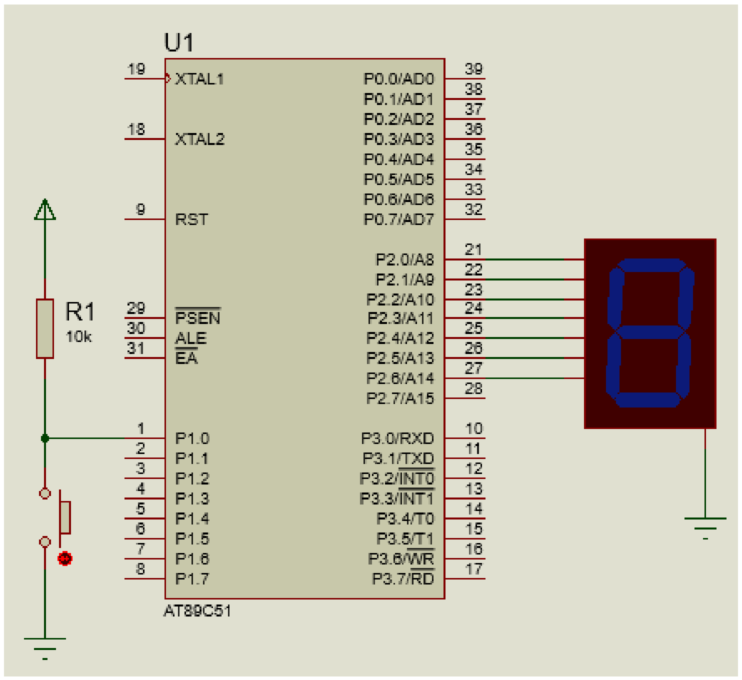


Fig 12. 7-Segment display and push button interfaced to 8051 microcontroller

As shown in Fig 12, a 7-segment display is connected to Port 2 (P2.0 to P2.6) and a push button switch is connected to P1.0. As long as switch is pressed, P1.0 is set to ZERO, 7-segment will display numbers from ‘0’ to ‘9’. If the switch is released, the 7-segment will stop displaying the characters. The 7-segment display shown in Fig 12 is of common cathode type of display.

Program:

```
#include <REG51.H>
#include <deay.h>
sbit  sw1=P1^0;
void main (void)
{
        unsigned char j=0;
        unsigned int digit[10]={0x40,0x79,0x24,0x30,0x19,0x12,0x03,0x78,
        0x00,0x18};                                          /*LED var*/
        P2=0x00;
        while (1)
        {
                for(j=0;j<10;j++)
                {
                        while(sw1!=0) ;
                        P2=digit[j];
                        delay(1);               /*call wait function*/
                }
        }
}
```

Experiment No. 7

7. Write an Assembly Language program to make multiple LEDs blink at a given rate using interrupt method.

Circuit Diagram:

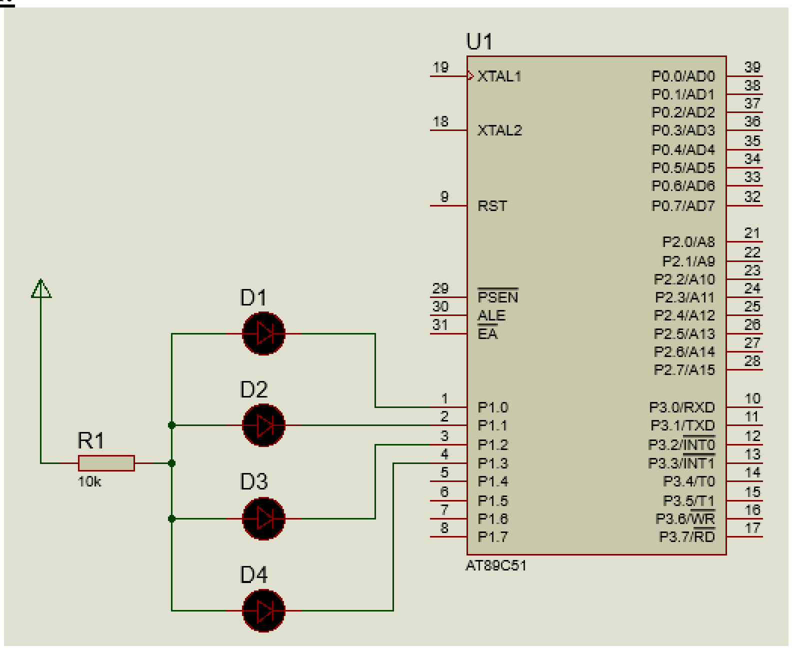


Fig 13. Four LEDs are connected to P1.0 to P1.3 of 8051 microcontroller

In Fig 13, four LEDs are connected to VCC through resistor. The LEDs can be triggered in any pattern and at any rate by modifying the program.

Program:

```
            ORG 0000H
            LJMP MAIN
            ORG 001BH
            DJNZ R0, L1
            CLR TF1
            CLR TR1
            CPL P1.0
            CPL P1.1
            CPL P1.2
            CPL P1.3
            MOV TL1,#00H
            MOV TH1,#00H
            SETB TR1
            MOV R0,#14
L1:         RETI
            ORG 0050H
MAIN:       MOV TMOD,#10H
            MOV IE,#88H
            MOV R0,#14
            MOV P1,#0FFH
            SETB TR1
HERE:       SJMP $
            END
```

Experiment No. 8

8. Write an Assembly Language program using interrupt method to make multiple LEDs blink in different patterns on push of a button.

Circuit Diagram:

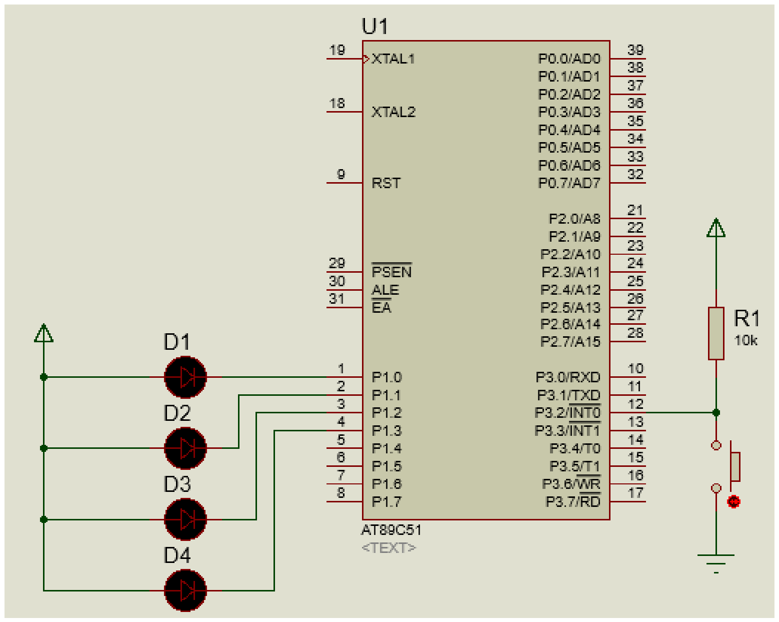


Fig 14. Four LEDs are connected to P1.0 to P1.3 of 8051 microcontroller

In this experiment, the LEDs blink rate and pattern can be controlled using a push button switch as shown in Fig 14. When the button is pressed, the pin P3.2 is set to ZERO and the LEDs starts blinking in a particular rate and pattern. If the button is not pressed, the LEDs remain in the OFF state.

Program:

```
        ORG 0000H
        LJMP MAIN
        ORG 0003H
        CPL P1.0
        CPL P1.1
        CPL P1.2
        CPL P1.3
        ACALL DELAY
        RETI
        ORG 0050H
MAIN:   MOV TMOD,#01H
        MOV R0,#14
        MOV TL0,#00H
        MOV TH0,#00H
        MOV IE,#81H
        MOV P1,#0FFH
DELAY:
BACK:   SETB TR0
```

```
HERE:       JNB TF0, $
            CLR TF0
            CLR TR0
            DJNZ R0, BACK
            MOV R0,#14
            RET
REPEAT:     SJMP $
            END
```

Experiment No. 9

9. Write an Assembly Language program to display a character on the Seven Segment Display (decade UP counter).

Circuit Diagram: Refer Fig 11.

Program:

```
                MOV DPTR,#80H
                MOV R2,#10
BACK:           CLR A
                MOVC A,@A+DPTR
                MOV P1,A
                ACALL DELAY
                INC DPTR
                DJNZ R2,BACK
                MOV R2,#10
                MOV DPTR,#80H
                SJMP BACK
DELAY:          MOV R1, #14H
L1:             MOV TMOD, #01H
                MOV TH0, #00H
                MOV TL0, #00H
                SETB TR0
HERE:           JNB TF0, $
                CLR TF0
                CLR TR0
                DJNZ R1, L1
                RET
                ORG 0080H
DATA1:          DB 7DH,30H,6Dh,79h,33h,5bh,1fh,70h,7fh,73h
                END
```

Experiment No. 10

10 a. Write an Embedded C program to make LEDs blink at a given rate using a delay function.

Circuit Diagram: Refer Fig 1.

Program:

```
#include <REG51.H>
sbit LED1=P1^0;
void delay(void)
{
        unsigned char i=0;
        for(i=0;i<14;i++)
        {
                TR0=1;
                while(TF0!=1);
                TF0=0;
                TR0=0;
        }
}
void main (void)
{
        TMOD=0X01;
        TH0=0x00;
        TL0= 0x00;
        while (1)
        {
                LED1=0;
                delay();
                LED1=1;
                delay();
        }
}
```

10 b. Write an Embedded C program to make multiple LEDs blink at a given rate using a delay function.

Circuit Diagram: Refer Fig 2.

Program:

```
#include <REG51.H>
sbit LED1=P1^0;
sbit LED2=P1^1;
void delay(void)
{
        unsigned char i=0;
        for(i=0;i<14;i++)
        {
                TR0=1;
                while(TF0!=1);
                TF0=0;
                TR0=0;
        }
}
void main (void)
{
        unsigned int i;
        unsigned char j;
        TMOD=0X01;
        TH0=0x00;
        TL0=0x00
       while (1)
       {
                LED1 = 1;
                LED2 = 1;
                delay();
                LED1 = 0;
                LED2 = 0;
             delay();
        }
}
```

10 c. Write an Embedded C program to make LED blink on push of a button using delay function.

Circuit Diagram: Refer Fig 6.

Program:

```
#include <REG51.H>
sbit SW1=P2^0;
sbit LED1=P1^0;
void delay(void)
{
        unsigned char i=0;
        for(i=0;i<14;i++)
        {
                TR0=1;
                while(TF0!=1);
                TF0=0;
                TR0=0;
        }
}
void main (void)
{
        SW1=1;
        LED1=0; /*LED var*/
        while (1)
        { /*Loop forever*/
                if (SW1 == 0)
                {
                        LED1=1; /*Output to LED Port*/
                        delay();
                        LED1=0;
                        delay();
                    }
            }
    }
```

10 d. Write an Embedded C program to make multiple LEDs blink in different patterns on push of a button using a delay function.

<u>Circuit Diagram:</u> Refer Fig 7.

<u>Program:</u>

```
#include <REG51.H>
sbit SW1=P2^0;
sbit LED1=1^0;
void delay(void)
{
        unsigned char i=0;
        for(i=0;i<14;i++)
        {
                TR0=1;
                while(TF0!=1);
                TF0=0;
                TR0=0;
        }
}
void main (void)
{
        SW1=1;
        LED1=0; /*LED var*/
        while (1)                       /*Loop forever*/
        {
                if (SW1 == 0)
                {
                        LED1=1; /*Output to LED Port*/
                        delay();
                        LED1=0;
                        delay();
                }
        }
}
```

Experiment No. 11

11. Two LEDs are connected to P2.0 and P2.1 respectively. Make the LED P2.0 blink at a rate of 1 second and the LED connected to P2.1 at the rate of 2 seconds. Write the program in Embedded C using interrupt method.

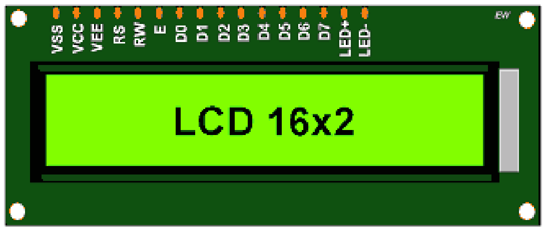
Program:

```
#include <REG51.H>
sbit  LED1=P2^0;
sbit  LED2=P2^1;
unsigned char count=28;
void delay(void)
{       unsigned char i=0;
        for(i=0;i<14;i++)
        {
                TR0=1;
                while(TF0!=1);
                TF0=0;
                TR0=0;
        }
}
void timer2seconds() interrupt 3
{
        if(count!=0)
        {
                count=count-1;
        }
        else
        {
                TF1=0;
                TR1=0;
                LED2=~LED2;
                TH1=0X00;
                TL1=0X00;
                TR1=1;
                count=28;
        }
}
void main (void)
{       unsigned int i;
        unsigned char j;
        TMOD=0X11; //To in Mode 1 and T1 in Mode 1
        IE=0X88; // Enable interrupts
        LED2=0;
        LED1=0;
        TR1=1;
        while (1)
        {
                LED1 = 0;
                delay() ;
                LED1 = 1;
                delay();
        }
}
```

Experiment No. 12

12. Interface an LCD with an 8051 and write an Embedded C program to display the message "Hello world."

16x2 LCD Display Pin Description:



No.	PIN	Function
1	VSS	Ground
2	VCC	+5 Volt
3	VEE	Contrast control 0 Volt: High contrast.

No.	PIN	Function
4	RS	Register Select 0: Commannd Reg. 1: Data Reg.
5	RW	Read / write 0: Write 1: Read
6	E	Enable H-L pulse
7-14	D0 - D7	Data Pins D7: Busy Flag Pin
15	LED+	+5 Volt
16	LED-	Ground

Fig 15. 16X2 LCD pin diagram and pin description

16x2 LCD Display Commands:

No	HEX Value	COMMAND TO LCD
1	0x01	Clear Display Screen
2	0x30	Function Set: 8-bit, 1 Line, 5x7 Dots
3	0x38	Function Set: 8-bit, 2 Line, 5x7 Dots
4	0x20	Function Set: 4-bit, 1 Line, 5x7 Dots
5	0x28	Function Set: 4-bit, 2 Line, 5x7 Dots
6	0x06	Entry Mode
7	0x08	Display off, Cursor off
8	0x0E	Display on, Cursor on
9	0x0C	Display on, Cursor off
10	0x0F	Display on, Cursor blinking
11	0x18	Shift entire display left
12	0x1C	Shift entire display right
13	0x10	Move cursor left by one character
14	0x14	Move cursor right by one character
15	0x80	Force cursor to beginning of 1st row
16	0xC0	Force cursor to beginning of 2nd row

First two pins of LCD 16x2 are used for ground and supply (+5 V).
Pin 3 - VEE pin: This pin is used for adjusting the contrast of the display. Voltage on this pin defines contrast on display, lower the voltage, higher the contrast. We can connect 4.7 k pot for contrast adjustment or simply connect this pin to ground to get maximum contrast.
Pin 4 –RS: Register Select pin
RS = 0: Data on the D0 to D7 pins is considered as a command**.**
RS = 1: Data on the D0 to D7 pins is considered as data to display on LCD16x2.

Pin 5 – RW: Read / Write pin
RW = 0: Write data to the LCD
RW = 1: Read data from the LCD
Pin 6 –E: Enable
This pin is used to latch the data present on the data pins D0 to D7. High to low pulse with a minimum width of 450 ns is required to latch the data to the display.
Pins 7:14 - DATA pins D0 to D7
Data pins are used to send data/command to the LCD16x2 as parallel 8 data bits.
Pin 15:16 - LED + and LED -
Liquid Crystal Displays don't have their own light like seven segment displays. Therefore, the module has a backlight LED. Supply to this LED is provided through these pins.
Fig 16 show the interfacing of 16x2 LCD with 8051 microcontroller.

Circuit Diagram:

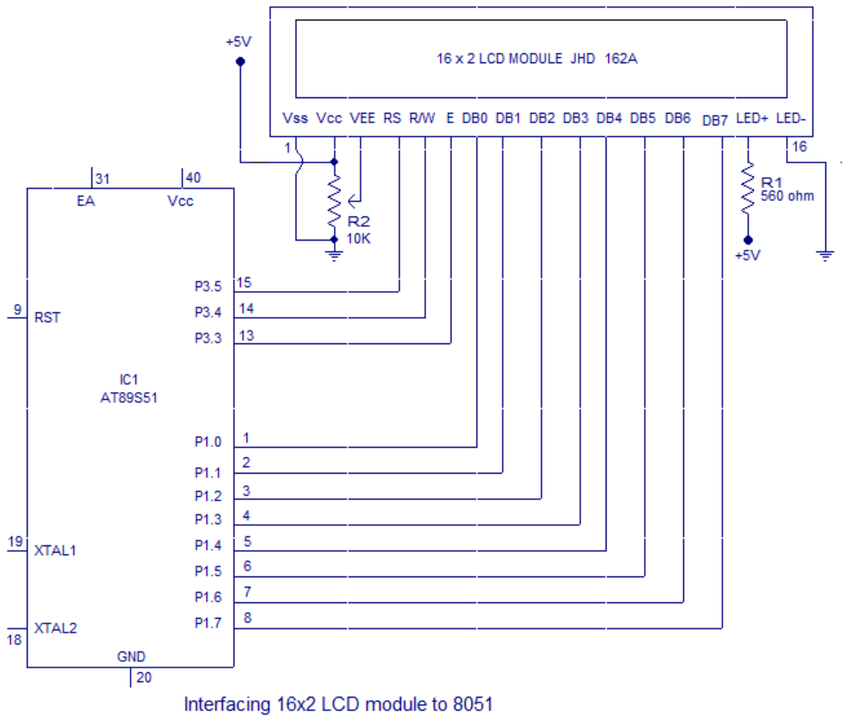

16 x 2 LCD MODULE JHD 162A
Vss Vcc VEE RS R/W E DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7 LED+ LED-
+5V
1
16
R1
560 ohm
+5V
R2
10K
31
40
EA
Vcc
9
RST
P3.5
P3.4
P3.3
15
14
13
IC1
AT89S51
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
1
2
3
4
5
6
7
8
19
XTAL1
XTAL2
18
GND
20
Interfacing 16x2 LCD module to 8051

Fig 16. LCD interface to 8051 microcontroller

Program:

```
#include<reg51.h>
#define display_port P2    //Data pins connected to port 2 on microcontroller
sbit rs = P3^5;                 //RS pin connected to pin 2 of port 3
sbit rw = P3^4;                 // RW pin connected to pin 3 of port 3
sbit e =  P3^3;                 //E pin connected to pin 4 of port 3
void msdelay(unsigned int time)  // Function for creating delay in milliseconds.
{
        unsigned int i,j ;
        for(i=0;i<time;i++)
        for(j=0;j<1275;j++);
}
```

```
void lcd_cmd(unsigned char command)  /*Function to send command instruction to
LCD*/
{
        display_port = command;
        rs= 0;
        rw=0;
        e=1;
        msdelay(1);
        e=0;
}
void lcd_data(unsigned char disp_data)  //Function to send display data to LCD
{
        display_port = disp_data;
        rs= 1;
        rw=0;
        e=1;
        msdelay(1);
        e=0;
}
void lcd_init()    //Function to prepare the LCD  and get it ready
{
        lcd_cmd(0x38);  // for using 2 lines and 5X7 matrix of LCD
        msdelay(10);
        lcd_cmd(0x0F);  // turn display ON, cursor blinking
        msdelay(10);
        lcd_cmd(0x01);  //clear screen
        msdelay(10);
        lcd_cmd(0x81);  // bring cursor to position 1 of line 1
        msdelay(10);
}
void main()
{
        unsigned char a[15]="Hello World";    //string of 14 characters with a null
        //terminator.
        int l=0;
        lcd_init();
        for(l=0;l<12;l++) // searching the null terminator in the sentence
        {
                lcd_data(a[l]);
                msdelay(50);
        }
}
```

<u>Probable viva questions:</u>

1. What is bit size of 8051 Microcontroller?
2. What is the size of internal RAM memory?
3. What is the size of internal ROM memory?
4. How many ports are there in 8051 microcontroller?
5. How interrupts an 8051 microcontroller can handle?
6. What is the size of ports in 8051 microcontroller?
7. What is the difference between bit address and byte address?
8. Mention the names of registers which are 16 bit in size.
9. What is a Special Function Register?
10. How many general purpose registers are there in 8051 microcontroller?
11. What is the size of general purpose register?
12. Which port is used to carry control signal?
13. What is the physical address of Program Counter?
14. How many bit addressable SFRs are there in 8051 microcontroller?
15. What is frequency of operation of 8051 microcontroller?
16. What is the range of frequencies with which 8051 microcontroller can work?
17. Which ports carries 16 bit address when 8051 is interfaced with external device?
18. Which port carries 8 bit data when 8051 is interfaced with external device?
19. How many timers are there in 8051 microcontroller?
20. What is the size of timer in 8051 microcontroller?
21. What is the difference between timer and counter?
22. What is the basic purpose of timer?
23. What are math or arithmetic or CPU registers in 8051 microcontroller?
24. What is the maximum size of internal ROM that can be accommodated in 8051 microcontroller?
25. What is the maximum size of external memory that can be interfaced to 8051 microcontroller?

References:

1. Muhammad Ali Mazidi and Janice Gillespie Mazidi and Rollin D. McKinley, “The 8051Microcontroller and Embedded Systems – using assembly and C”-, PHI, 2006 / Pearson,2006.

2. Kenneth J. Ayala, 2e "The 8051 Microcontroller Architecture, Programming &Applications”, Penram International, 1996 / Thomson Learning 2005.

3. V. Udayashankar and MalikarjunaSwamy, “The 8051 Microcontroller”, TMH, 2009.

4. <u>https://robu.in/</u>