MALLA REDDY ENGINEERING COLLEGE

(AUTONOMOUS)

An UGC Autonomous Institution, affiliated to JNTUH,
Accredited by NAAC with 'A' Grade & NBA and
Recipient of World Bank Assistance under TEQIP–II S.C.1.1

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LAB MANUAL

54126 - EMBEDDED SYSTEM LABORATORY

I M.Tech I Semester

(MR-15 Regulations)

Prepared By
Thota Ravi Theja,
Assistant Professor,
ECE Department,
MREC (A).

Verified By

Approved By

LIST OF EXPERIMENTS

Note: The following programs are to be implement on 89C51 Development board using Embedded C Language on Keil IDE and Flash magic.

List of experiments:

1. Program to toggle all the bits of Port P1 continuously with 250 mS delay.
2. Program to toggle only the bit P1.5 continuously with some delay. Use Timer 0, mode 1 to create delay.
3. Program to interface a switch and a buzzer to two different pins of a Port such that the buzzer should sound as long as the switch is pressed.
4. Program to interface LCD data pins to port P1 and display a message on it.
5. Program to interface keypad. Whenever a key is pressed, it should be displayed on LCD.
6. Program to interface seven segment display unit.
7. Program to transmit a message from Microcontroller to PC serially using RS232.
8. Program to receive a message from PC serially using RS232.
9. Program to get analog input from Temperature sensor and display the temperature value on PC Monitor.
10. Program to interface Stepper Motor to rotate the motor in clockwise and anticlockwise directions.
11. Program to interfacing RFID.
12. Implementation of Traffic light controller.

INDEX

Introduction to Embedded System and Development Board

UTS-MC-KIT-M7 is next version of the UTS-MC-KIT-M5 with placement modification in chips and functionally it same as M5 board. UTS-MC-KIT-M7 is high end efficient and versatile controller module built with Phillips Microcontroller P89V51RD2. This controller module can be used for wide range of applications right from simple traffic light controller to lift controller ,data loggers applications and many.

The UTS-MC-KIT-M7 has got P89V51RD2 microcontroller which has got 64Kilo Bytes of on chip Flash memory and 1 Kilo Bytes of RAM. The kit is has got on board 11.0592MHz crystal for generating the on chip clock of 11.0592MHz.

A Key feature of the board is it has got so many interfaces with different on board peripherals and has got expansion capability to add any further sensor and peripherals in future. The another key featre of the board is it has got configurable address space for peripherals by programming the on chip PLD chip.

The most important feature of this board is it has got Insystem Programmable feature, where this feature allows to program the microcontroller from PC through the serial port link. So this avoids the extra requirement of the programmer device for programming the microcontroller and development cycle also improves as it does not require to pull out the microcontroller chip for programming for every programming iteration.

The board has got on chip peripherals like on board 32 KB bytes of RAM, Eight Light Emitting Diodes, Eight Push Buttons, Eight Slide Switches, Two Seven Segment Displays, 16X2 Liquid Crystal Character Display(LCD), Analog to Digital Convertor, Temperature sensor interface, Real time clock, RS-232 serial interface.

Requirements

The UTS-MC-KIT-M7 kit requires power adaptor, serial cable for connecting to PC. And few connecting cables. in the PC you should install the UTS EDS software / Keil evaluation Software is required for writing the application in C/Assembly language, compiling and finally for generation of the hex file for downloading into the microcontroller.

Make sure you have satisfied the below hardware software requirements:-

Hardware Rquirements :

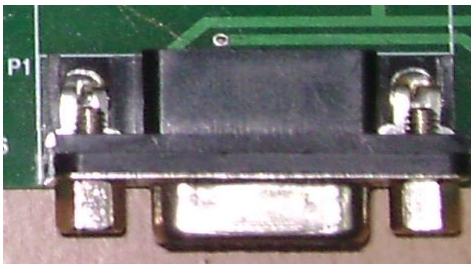
1. UTS-MC-KIT-M7
2. A serial cable , 9 pin cable wired one to one from female connector to male connector
3. PC with serial port
4. 5V adaptor

Software Rquirements :

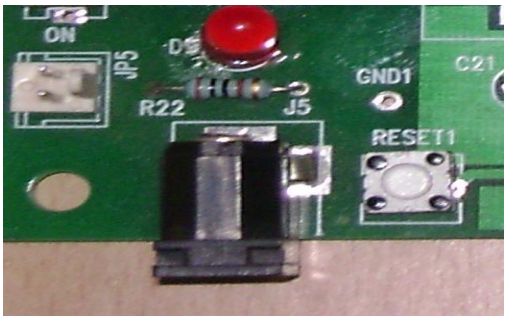
1. UTS EDS or Keil evaluation software
2. Flash Magic tool.

Connecting UTS-MC-KIT-M7

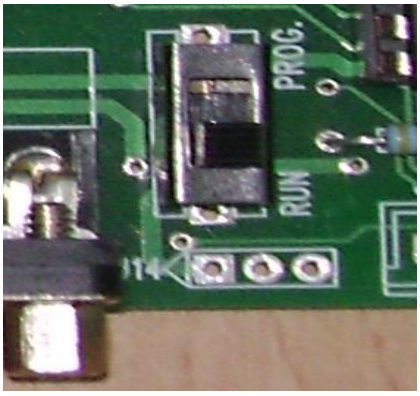
1. Connect supplied serial port cable to external serial port jack(RS-232 jack) on the PC back side



2. Connect the power supply adaptor to power supply adaptor socket(J5 connector) on the board.



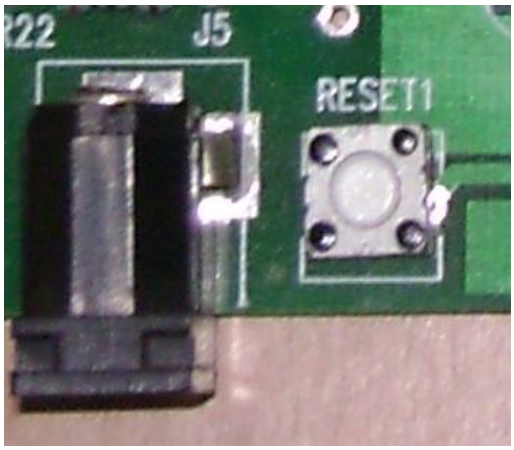
3. The serial port allows you to download the hex files into the microcontroller thorough the flash magic tool from the PC.
4. Select the switch to PROG position apply the RESET before downloading the hex file into the microcontroller (Not applicable for P89V51RD2).



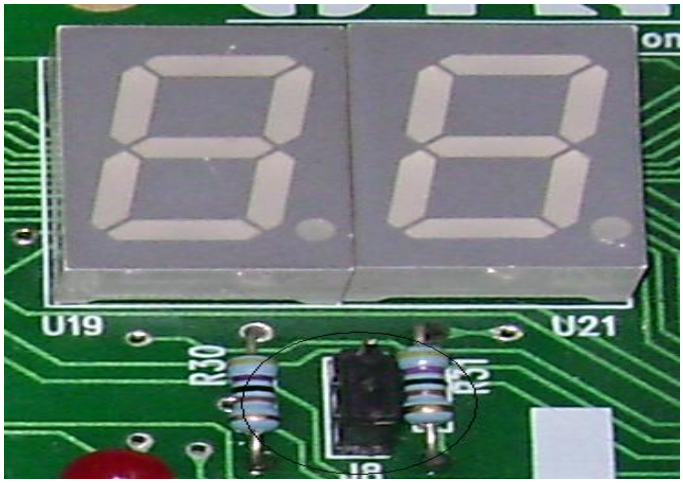
5. Select the switch to RUN position and apply the RESET. Then downloaded application will be running.

Jumper, Connectors & Switch Settings on the Board

1. PROG/RUN Switch (Not applicable for P89V51RD2)
    a. Keep in PROG position and apply RESET by pressing the RESET1 pushbutton on the board to configure the board in programming mode for accepting the hex file from Flash Magic tool running on PC.
    b. Keep in RUN position and apply RESET by pressing the RESET1 pushbutton on the board to configure the board in Run mode for running the downloaded hex file application.



2. J8 Connector
    a. Place jumper as per shown in the figure bottom side for configuring the Seven segment display in common anode configuration.



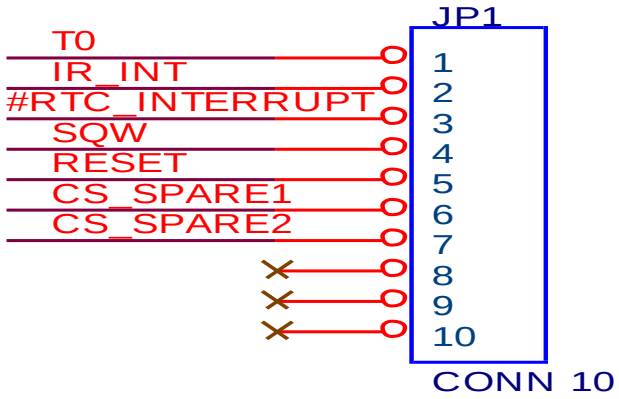
b. Place jumper as per shown in the figure to top side for configuring the Seven segment display in common cathode configuration.

Note : Default setting is jumper position is in bottom side

3. JP1 Connector

Pin No	Name	Remarks
1	T0	Connected to pin of the Microcontroller.
2	INT1	Connected to INT1 pin of the MC
3	RTC Interrupt	Connected to Real Time Clock IRQ pin
4	SQW	Connected to Real Time Clock SQW pin
5	RESET	Connected to MC Reset Pin
6	CS_SPARE1	Extra Chip select signal for future expansion of the board
7	CS_SPARE2	Extra Chip select signal for future expansion of the board

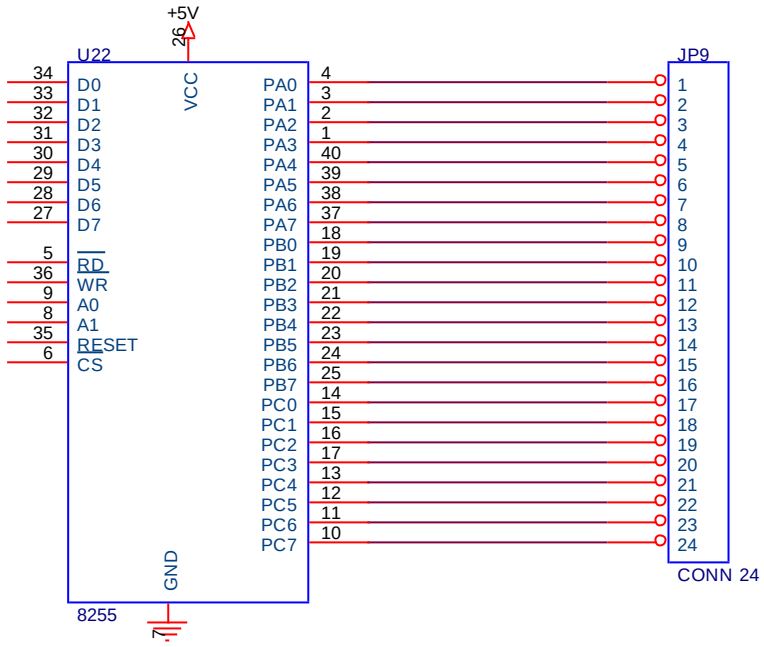
8,9,10	No Connection	



4. J7 Connector
    a. External Supply Connector for Stepper Motor. But small stepper motor which is supplied with the UTS-MC-KIT-M7 can be directly used with out supplying the external supply for motor. As it draws the supply from the internal supply of the board. You can leave connector with out connected any thing
5. JP7 Connector
    a. Connector for Stepper Motor -1
6. JP8 Connector
    a. Connector for Stepper Motor – 2
7. JP9 Connector
    a. Expansion I/O Connector

Pin No	Name	Remarks
1.	Port A 0	I/O Expansion Connector. The 8255_3 can be programmed for configuring the I/O lines direction.
2.	Port A 1	
3.	Port A 2	
4.	Port A 3	
5.	Port A 4	
6.	Port A 5	
7.	Port A 6	
8.	Port A 7	
9.	Port B 0	

10.	Port B 1	
11.	Port B 2	
12.	Port B 3	
13.	Port B 4	
14.	Port B 5	
15.	Port B 6	
16.	Port B 7	
17.	Port C 0	
18.	Port C 1	
19.	Port C 2	
20.	Port C 3	
21.	Port C 4	
22.	Port C 5	
23.	Port C 6	
24.	Port C 7	
25.	No Connection	
26.	No Connection	

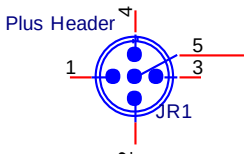


8. P1 Connector
    a. Serial Port Connector Female.
9. J6 Connector

    Not used
10. JP6 Connector

    Not Used
11. JP3 Connector
    a. ADC input connector. Right Side pin is input to ADC and left side pin is ground.
12. J5 Connector
    a. ADC negative input Pin connector. Connect the jumper to top side to make grounding the negative input of the ADC. If negative input is required to use like for taking the wein bridge output then connect the jumper to down direction.
13. JR1 Connector
    a. Used for Selecting the input to the ADC. There are four inputs can be selected to ADC input.
        i. Temperature Sensor output
        ii. Wein bridge output
        iii. External input
        iv. Audio Amplifier output (Provision has been made for future expansion for connection the audio signal to board. In this current version the audio amplifier circuit is not implemented.)

Audio Amplifier Output



Temperature Sensor output Wein Bridge Output

External Input

The Corresponding direction jumper has to be placed

14. JP5

a. External 5V Power Supply connector. In case adaptor is not available, it is possible to give external supply between 5 V.
b. Top Pin is Positive and Bottom pin is ground

15. Power1 Switch
    a. Power ON switch to Board
16. J5 Connector
    a. Power Adaptor Socket. Inner is Positive and outer is ground.

(Caution: when you are using non standard adaptor, ensure that inner one is positive and outer is ground. The supply voltage is 5V)

17. JP2 Connector
    a. Power Connector. Provides 5V and Ground. 10V,-10V, 12V pins are not used. Can be used to connect to small loads like external sensors.

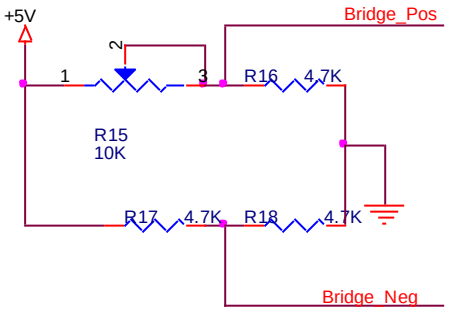
18. J4 Connector
    a. Audio Jack for connecting the Audio input to board. This option is dropped for this version of the board.
19. J9 Connector
    a. External Jack for taking the external analog input to ADC.
20. J3 Connector
    a. PORT 1 and LED external cable connector

Pin No	Name	Remarks
1	P1_0	Connected to Port1 pins of Microcontroller.
2	P1_1	
3	P1_2	
4	P1_3	
5	P1_4	
6	P1_5	
7	P1_6	
8	P1_7	
9	VCC(+5V)	
10	VCC(+5V)	

11	LED 1	For Future Expansion. To connect the remote LEDs to board.
12	LED 2	
13	LED 3	
14	LED 4	
15	LED 5	
16	LED 6	
17	LED 7	
18	LED8	
19,20	No Connection	

POT setting on the boards

1. P2
    a. LCD intensity Adjust POT. Location near to LCD. Adjust the POT till Text/Squares appear on the display
2. R10
    a. Location near to the ADC chip
    b. Adjusts the reference voltage to Analog to Digital Convertor (ADC). Adjust the POT R10 and observe the voltage at Pin 9 of U9 IC, i.e ADC0804. The reference value can be varied from 0 to 2.5Volts by varying the POT.
3. R12
    a. Location Near to Audio Amplifier
    b. Ineffective for the current version of the board.
4. R15
    a. One of the arm of the wein bridge.



    b. To connect the bridge positive and negative to ADC inputs Positive and negative inputs. The following jumper configuration has to be done.

c. Connect the J5 Jumper to down side and connect the JR1 jumper to Right side. Vary the R15 and observe the change in the input voltage at the input of the ADC.

Power Indications

1. D9
    a. Power ON Indicator LED.

Memory Map of UTS-MC-KIT-M7 Board

Sl No	Peripheral / Registers	Address
1	8255_1 Control Register	8003
2	Digital Analog Convertor (PORT A). Not used this version.	8000
3	Slide Switches (PORT B)	8001
4	Stepper Motor -1 (PORT C)	8002 Lower Nibble
5	Stepper Motor – 2 (PORT C)	8002 Higher Nibble
6	8255_2 Control Register	9003
7	LED (PORT A)	9000
8	Seven Segment Display-1 (PORT B)	9001
9	Seven Segment Display-2 (PORT C)	9002
10	8255_3 Control Register	A003
11	PORT A Register	A000
12	PORT B Register	A001
13	PORT B Register	A002
14	LCD #	B000
15	Real Time Clock (RTC)	C000
16	SPARE_1	E000
17	SPARE_2	F000

RS control for LCD is given from P3_5 pin of the microcontroller.

The above address shall be used to access the peripherals on the board. Before accessing the peripherals, the corresponding 8225 ports to be configured.

Component Description

Microcontroller

The P89V51RD2 device contains a non-volatile 64KB Flash program memory. In-System

Programming (ISP) allows the user to download new code while the microcontroller sits in the application. A default serial loader (boot loader) program in ROM allows serial In-System programming of the Flash memory via the UART without the need for a loader in the Flash code.

This device executes one machine cycle in 6 clock cycles, hence providing twice the speed of a conventional 80C51. An OTP configuration bit lets the user select conventional 12 clock timing if desired.

This device is a Single-Chip 8-Bit Micro controller manufactured in advanced CMOS process and is a derivative of the 80C51 micro controller family. The instruction set is 100% compatible with the 80C51 instruction set.

The device also has four 8-bit I/O ports, three 16-bit timer/event counters, a multi-source, four-priority-level, nested interrupt structure, an enhanced UART and on-chip oscillator and timing circuits.

The added features of the P89V51RD2 makes it a powerful micro controller for applications that require pulse width modulation, high-speed I/O and up/down counting capabilities such as motor control.

Features:

- 80C51 Central Processing Unit
- On-chip Flash Program Memory with In-System Programming (ISP)
- Boot ROM contains low level Flash programming routines for downloading via the UART
- Can be programmed by the end-user application (IAP)
- Six clocks per machine cycle operation (standard)
- 12 clocks per machine cycle operation (optional)
- Speed up to 20 MHz with 6 clock cycles per machine cycle (40 MHz equivalent performance); up to 33 MHz with 12 clocks per machine cycle
- Fully static operation
- RAM expandable externally to 64 k bytes
- Four interrupt priority levels
- Seven interrupt sources
- Four 8-bit I/O ports
- Full-duplex enhanced UART
    - Framing error detection

- Automatic address recognition
- Power control modes
  - Clock can be stopped and resumed
  - Idle mode
  - Power down mode
- Programmable clock out
- Second DPTR register
- Asynchronous port reset
- Low EMI (inhibit ALE)
- Programmable Counter Array (PCA)
  - PWM
  - Capture/compare

RAM:

The CY62256 is a high-performance CMOS static RAM organized as 32K words by 8 bits. Easy memory expansion is provided by an active LOW chip enable (CE) and active LOW output enable (OE) and three-state drivers. This device has an automatic power-down feature, reducing the power consumption by 99.9% when deselected. An active LOW write enable signal (WE) controls the writing/reading operation of the memory. When CE and WE inputs are both LOW, data on the eight data input/output pins(I/O0 through I/O7) is written into the memory location addressed by the address present on the address pins (A0 through A14). Reading the device is accomplished by selecting the device and enabling the outputs, CE and OE active LOW, while WE remains inactive or HIGH. Under these conditions, the contents of the location addressed by the information on address pins are present on the eight data input/output pins. The input/output pins remain in a high-impedance state unless the chip is selected, outputs are enabled, and write enable.

Features

- Temperature Ranges
  —Commercial: 0°C to 70°C
- High speed: Access time 70 ns
- Voltage range: 4.5V–5.5V operation
- Low active power (70 ns, LL version, Com'l and Ind'l)
  —275 mW (max.)
- Low standby power (70 ns, LL version, Com'l and Ind'l)

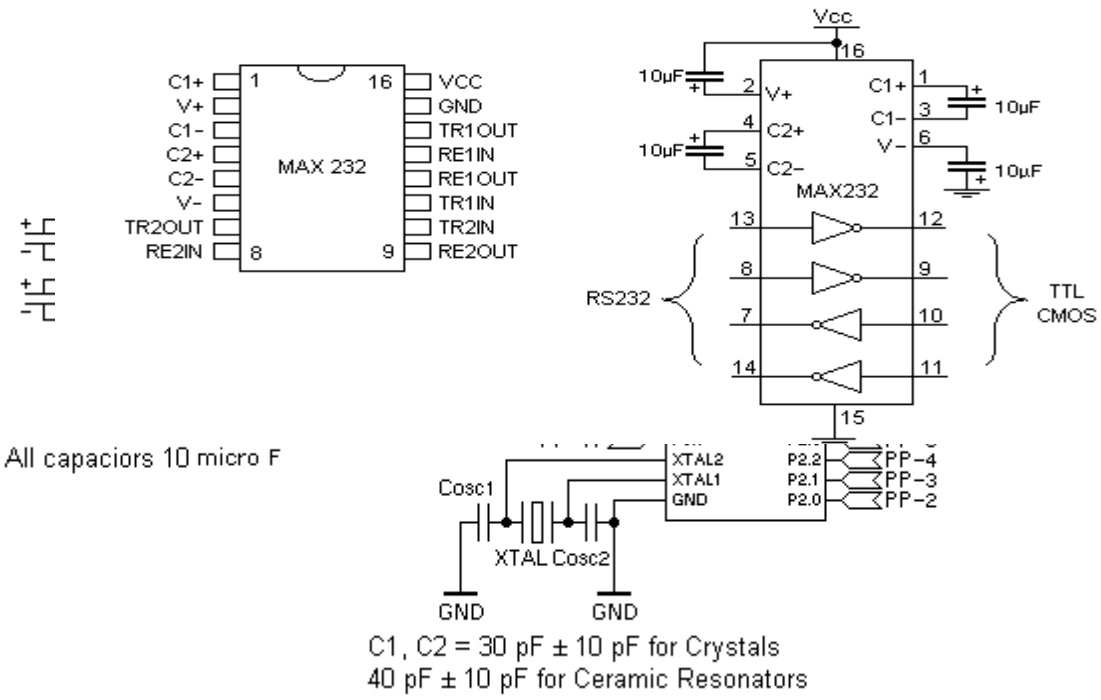
—28 μW (max.)

- Easy memory expansion with CE and OE features
- TTL-compatible inputs and outputs
- Automatic power-down when deselected
- CMOS for optimum speed/power
- Package : 600-mil 28-lead PDIP packages

RS-232 Interface

Standard serial interfacing of microcontroller (TTL) with PC or any RS-232 standard device, requires TTL to RS-232 level converter. A MAX232 is used for this purpose. It provides 2-channel RS-232 port and requires external 10uf capacitor.

- The driver requires a single supply of +5v.
- MAX232 includes a charge pump. which generates +10v and -10v from a single 5v supply.



The Seven-Segment LED

One common requirement for many different digital devices is a visual numeric display. Individual LEDs can of course display the binary states of a set of latches or flip-flops. However,

we're far more used to thinking and dealing with decimal numbers. To this end, we want a display of some kind that can clearly represent decimal numbers without any requirement of translating binary to decimal or any other format.

This requires just seven LEDs (plus an eighth one for the decimal point, if that is needed). A common technique is to use a shaped piece of translucent plastic to operate as a specialized optical fiber, to distribute the light from the LED evenly over a fixed bar shape. The seven bars are laid out as a squared-off figure "8". The result is known as a seven-segment LED.

We've all seen seven-segment displays in a wide range of applications. Clocks, watches, digital instruments, and many household appliances already have such displays.

The illustration to the right shows the basic layout of the segments in a seven-segment display. The segments themselves are identified with lower-case letters "a" through "g," with segment "a" at the top and then counting clockwise. Segment "g" is the center bar.

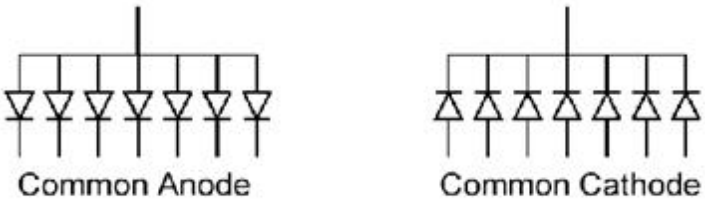
Most seven-segment digits also include a decimal point ("dp"), and some also include an extra triangle to turn the decimal point into a comma. This improves readability of large numbers on a calculator, for example. The decimal point is shown here on the right, but some display units put it on the left, or have a decimal point on each side.

Led displays are

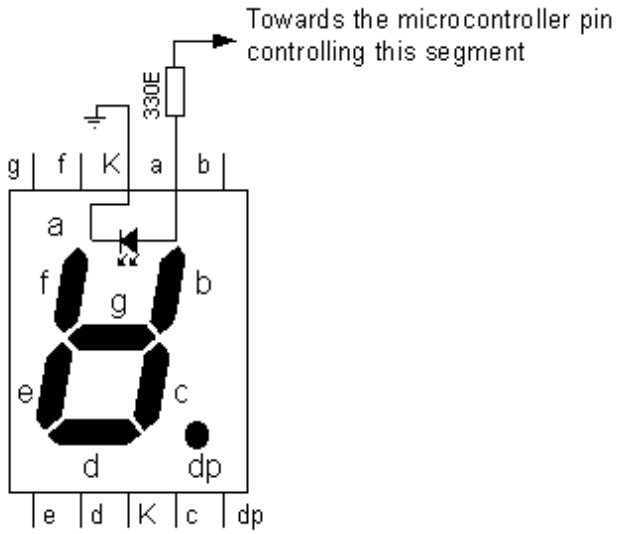
- power-hungry(10ma per led)
- pin-hungry(8-pins per 7-segment display)
- 7-SEG displays are available in two types.

  1.common anode

  2.common cathode



The UTS-MC-KIT-M7 comes with common anode type. There are two seven segment displays are connected to 8255_2 port B & C. The hex code for displaying the corresponding values is given in



below table. The corresponding hexadecimal values has to be written on the corresponding ports of the 8255 for displaying on the seven segment display.

Values to Be displayed on Seven Segment Display	**Value to be sent** to Seven segment display (In Hexdecimal)
0	0x40
1	0x79
2	0x24
3	0x30
4	0x19
5	0x12
6	0x02
7	0x78
8	0x00
9	0x18

A	0x08
B	0x03
C	0x46
D	0x21
E	0x06
F	0x0E

Table : Seven Segment Decoding Values

Analog To Digital Convertor (ADC)

The ADC0804 is CMOS 8-Bit, successive approximation A/D converters which use a modified potentiometric ladder and are designed to operate with the processors/microcontrollers control bus via three-state outputs. These converters appear to the processor as memory locations or I/O ports, and hence no interfacing logic is required. The differential analog voltage input has good common-mode-rejection and permits offsetting the analog zero-input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

Features:

• Compatible with 8051 μc derivatives-no interfacing logic needed - access time - 135 ns

• Easy interface to all microprocessors, or operates "stand alone"

• Differential analog voltage inputs Conversion Time <100μs

• Logic inputs and outputs meet both MOS and TTL voltage level specifications

• Works with 2.5V (LM336) voltage reference

• On-chip clock generator

• 0V to 5V analog input voltage range with single 5V supply

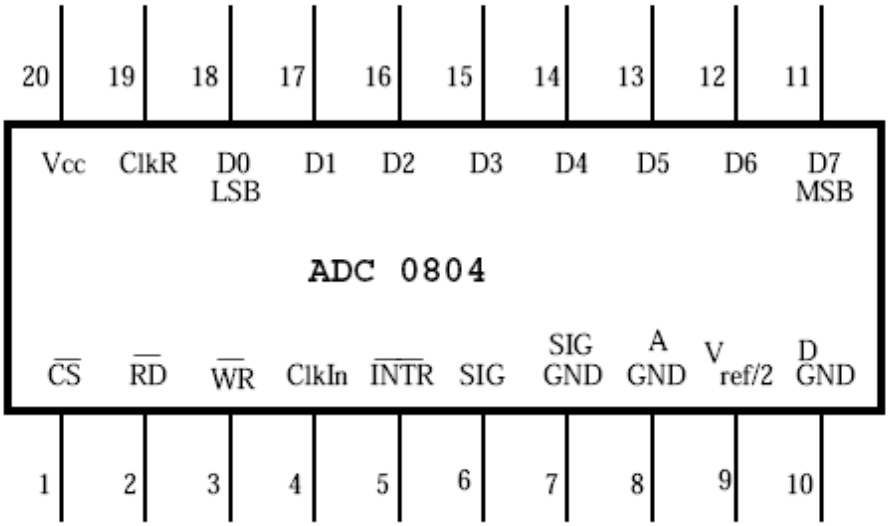
• No zero adjust required

• 0.3[Prime] standard width 20-pin DIP package

• 20-pin molded chip carrier or small outline package

• Operates ratiometrically or with 5 V_{DC}, 2.5 V_{DC}, or analog span adjusted voltage reference

PIN DIAGRAM:



Pin Number Description

1 CS - Chip Select (Active Low)

2 RD - Read (Active Low)

3 WR - Write (Active Low)

4 CLK IN - Clock IN

5 INTR - Interrupt (Active Low)

6 Vin+ - Analog Voltage Input

7 Vin- - Analog Voltage Input

8 AGND - Analog Ground

9 Vref/2 - Voltage Reference / 2

10 DGND - Digital Ground

11 DB7 - Data Bit 7 (MSB)

12 DB6 - Data Bit 6

13 DB5 - Data Bit 5

14 DB4 - Data Bit 4

15 DB3 - Data Bit 3

16 DB2 - Data Bit 2

17 DB1 - Data Bit 1

18 DB0 - Data Bit 0 (LSB)

19 CLKR - Clock Reset

20 Vcc - Positive Supply or Vref

Temperature Sensor Interfacing to Microcontroller using ADC804:

The ADC804 has 8-bit resolution with a maximum of 256 steps and the LM35 temperature sensor provides 10mV for every degree of temperature change.

We shall do Calibration such that for temperature range of 0 to 100 C, Voltage in at input of ADC will be 0 to 2.56 v.

we need to set Vref/2 = 1.28V

so step size will be 2560mv/256 = 10mv also for every degree change in temp. LM35 output changesby10mv ,so every degree change in temp. will produce 1 unit change in digital out of ADC

Thus resolution of our system will be 1deg C , which is Smallest temp. that we can measure with this system. If resolution to be with 0.5deg C the Vref value to be adjusted to 0.64V.

The Vref voltage can be adjusted varying the POT R10.

R10 POT

a. Location near to the ADC chip
b. Adjusts the reference voltage to Analog to Digital Convertor (ADC). Adjust the POT R10 and observe the voltage at Pin 9 (i.e Vref Pin of the ADC) of U9 IC, i.e ADC0804. The reference value can be varied from 0 to 2.5Volts by varying the POT.

LCD Interfacing

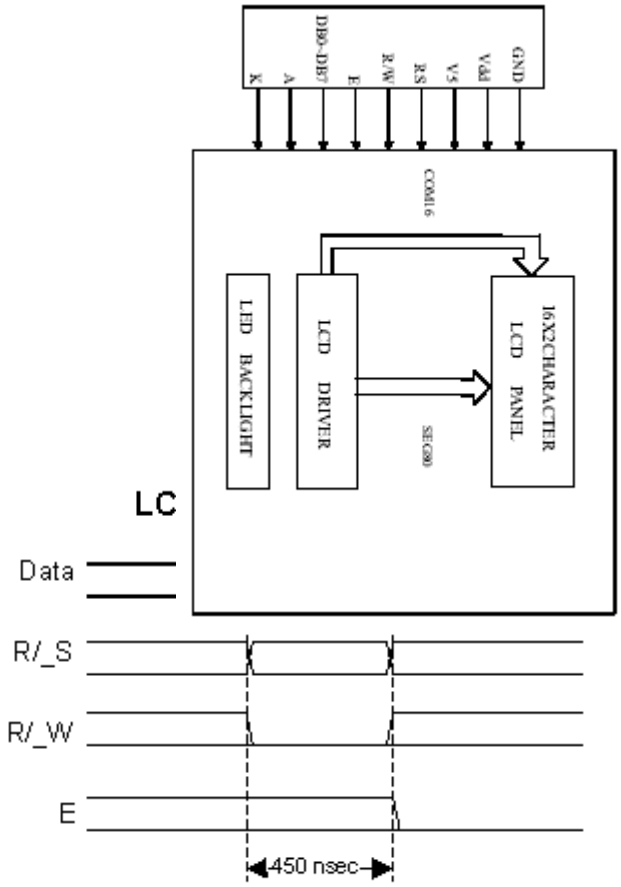
LCDs can add a lot to your application in terms of providing an useful interface for the user, debugging an application or just giving it a "professional" look. LCDs can be added quite easily to an application and use as few as three digital output pins for control.

The most common connector used for the LCDs is 16 pins in a row. The pins are wired as:

Pin NO.	Symbol	Function		Remark
1	GND	Power supply	0V	
2	Vdd		+5V	
3	V5		For LCD	Variable
4	RS	Register Select(H=Data,L=Instruction)		
5	R/W	Read/Write L=MPU to LCM,H=LCM to MPU		
6	E	Enable		
7	DB0	Data bus bit 0		
8	DB1	Data bus bit 1		
9	DB2	Data bus bit 2		
10	DB3	Data bus bit 3		
11	DB4	Data bus bit 4		
12	DB5	Data bus bit 5		
13	DB6	Data bus bit 6		
14	DB7	Data bus bit 7		
15	A	Anode of LED Unit		
16	K	Cathode of LED Unit		

- In this Kit 4 bit mode is configured.
- R/S is controlled by P2_5 of the Microcontroller.
- R/W is permanently grounded to make it in always write mode to LCD
- Data Bus are connected to microcontroller Buffered Data Bus.

Internal Block Diagram of the LCD is shown in below figure



The interface is a parallel bus, allowing simple and fast writing of data to the LCD. This waveform will write an ASCII Byte out to the LCD's screen. The ASCII code to be displayed is eight bits long and is sent to the LCD either four or eight bits at a time. If four bit mode is used, two "nibbles" of data (Sent high four bits and then low four bits with an "E" Clock pulse with each nibble) are sent to make up a full eight bit transfer. The "E" Clock is used to initiate the data transfer within the LCD.

Sending parallel data as either four or eight bits are the two primary modes of operation. While there are secondary considerations and modes, deciding how to send the data to the LCD is most critical decision to be made for an LCD interface application. Eight bit mode is best used when speed is required in an application and at least ten I/O pins are available. Four bit mode requires a minimum of six bits. To wire a microcontroller to an LCD in four bit mode, just the top four bits (DB4-7) are written to.

The "R/S" bit is used to select whether data or an instruction is being transferred between the microcontroller and the LCD. If the Bit is set, then the byte at the current LCD "Cursor" Position can be read or written. When the Bit is reset, either an instruction is being sent to the LCD or the execution status of the last instruction is read back (whether or not it has completed).

The different instructions available for use LCD are shown in below:

1. Clear display
2. Return home
4. Decrement cursor
6. Increment cursor
5. Shift display right
6. Increment cursor.
7. Shift display left
8. Display off,Cursor off
A. Display off,Cursor on
C. Display on,Cursor off
E. Display on Cursor blinking on
F. Display on Cursor blinking off
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 of 1st line

C0. Force cursor to beginning of 2nd line

38. 2 lines and 5x7 matrix

For further in detail description of the registers can go through the Data sheet of 16X2 character LCD display.

Before you can send commands or data to the LCD module, the Module must be initialized

LEDS

There are eight LEDs on the board from D1 to D7 for displaying eight bit information. D1 is LSB and D7 is MSB. These eight LEDs are connected to 8255_2 PORT A. so by configuring the 8255_2 PORTA in output mode and can write on to LEDs. The LEDS are connected in common cathode fashion. So the data written on Port A will be displayed with LEDs indication directly.

PUSH Buttons

There are eight pushbuttons are available in the Kit named as PSB1 to PSB7. These push buttons are connected to PORT 1 of the microcontroller directly. PSB1 is connected to P1_ 0 and PSB2 is connected to P1_1 ……. And PSB7 is connected to P1_7.

Experiment 1:

Program to toggle all the bits of Port P1 continuously with 250ms Delay.

Solution:

```
#include <reg51.h>
void main( )
{
unsigned char i;
for( ; ; )
{
P1=0X55;
for(i=0;i<250;i++);
P1=0XAA;
}
}
```

Results:-

Experiment 2 :

Write an 8051 C program to toggle only bit P1.5 continuously with some delay(eg. every 50ms).Use Timer 0, mode 1 (16-bit) to create the delay.

Solution:

```
/*Assume XTAL=11.0592 MHz=> T=1.085µs Count=50ms/1.085µs =46083
Inital count = 65536-46083 =19453
Count in Hex = 4BFDH*/

#include <reg51.h>
void T0M1Delay(void);
sbit mybit=P1^5;
void main(void)
{
while (1)
{
mybit=~mybit; //togle P1.5
T0M1Delay();
}}
void T0M1Delay(void)
{
TMOD=0x01; / /Timer 0, mode 1
TL0=0xFD;
TH0=0x4B;
TR0=1;
while (TF0=0);
TR0=0;
TF0=0;   }
```

Results:-

Experiment 3:

Program to interface a switch and a buzzer to two different pins of a Port such that the buzzer should sound as long as the switch is pressed.

Solution:

```
#include <reg51.h>
void T0M1Delay(void);
sbit mybit=P1^5;
sbit Buzzer=P1^7;
void main(void)
{
while (1)
{
mybit=1;
Buzzer=mybit; //togle P1.5
T0M1Delay();
mybit=0;
Buzzer=0;
}
}
void T0M1Delay(void)
{
TMOD=0x01; // Timer 0, mode 1
TL0=0xFD;
TH0=0x4B;
TR0=1;
while (TF0=0);
TR0=0;
TF0=0;
}
```

Results:-

Experiment 4:

Write a program for interfacing the 16x2 LCD MODULES

Aim:
Write a program for interfacing the 16x2 lcd module

Introduction:
In this experiment a program is written in C language to display a message in the lcd first and second line and then clearing the first and second line. This program is kept in an continuous loop and the output can be seen on the board.

The program flow chart

- Write a program for lcd interface
- Compile the program and generate Hex file.
- The generated hex file will be downloaded in to the MC and verify the result.

KIEL software is used for compilation of C files and the hex file can be generated.

Experimental procedure:

1. Open the kiel ide and you can see a menu.
2. To create a new project just click the project menu and click new microvision project
3. Then a dialog box will be opened to give the destination as shown in the figure. You can give the destination folder and save it.
4. To write a c file click the file menu and select the new document. The desired code is written in the document. And this document is needed to be added to the project.
5. In the project window on the left hand side you can see a project window .in the window a folder called target can be seen to that folder another folder can be seen as target source group1. the c file which is written nedded to be added to that folder . to add a c file to that folder right click the source group1 folder and a menu will be appeared.
6. Before bulding the project some settings are to made, from the main menu click the flash, you can see the option as **configure flash tools** . you click that option.From that menu click output and select an option called **create hex file**. And in the target menu give the oscillator frequency as **11.0592.**
7. Now all the hard work we have done to build the project and to create the hex file.From the main menu select the project and in that an option called **build target** can be seen , just click that . if all the coding is correct and the syntaxes are correct an hex file will be generated.

Down loading the hex file to the target:

Follow the same steps as mentioned in Ex 1-a for downloading the HEX File onto the microcontroller board

Results/Output verification: Now the lcd program is running on Microcontroller. And the output can be seen in the board.

C SOURCE CODE

```
#include <reg51.h>
#include"lcd.h"          // refer LCD.H code for more to about LCD

 /* Hardware : Controller  -> P89V51RD2
 *          XTAL       -> 11.0592 MHz
 *
 * I/O     : RS         -> P2.5
 *          Enable     -> P2.4
 *          Data4567    -> P2.0,P2.1,P2.2,P2.3
 * Compiler : uvision 4
 * Author   : siva.eh@gmail.com
 * Date          : 18/05/10
 */
void Delay_sec(unsigned char s);
main( )
{
   LCD_init();
         while(1){
         LCD_clear();
         Delay_sec(1);
         LCD_row1();
         LCD_puts(" HELLO LCD123. ");
   LCD_row2();
         LCD_puts(" TEST PROGRAME ");
         Delay_sec(5);
         LCD_clear();
         Delay_sec(1);
   LCD_row1();
         LCD_puts("  Hello Welcome ");
   LCD_row2();
         LCD_puts("UNISTRING TECH P L");
         Delay_sec(5);
         }
  }
void Delay_sec(unsigned char s)
{
         unsigned char n;

         for (n=0; n<s; n++){
         LCD_delay(250);
         LCD_delay(250);

         }
}
```

```
// Filename : lcd.h
#define LCD_DELAY 150 /* Delay for 1 ms */
 #define LCD_clear() LCD_command(0x1)  /* Clear display LCD */
 #define LCD_origin() LCD_command(0x2)/* Set to origin LCD */
 #define LCD_row1() LCD_command(0x80)          /* Begin at Line 1 */
 #define LCD_row2() LCD_command(0xC0)   /* Begin at Line 2 */
 /*******************************************************
 * Prototype(s)                                 *
 *******************************************************/
sbit  LCD_en=P2^4;
sbit  LCD_rs=P2^5;
void LCD_delay(unsigned char ms);
void LCD_enable();
void LCD_command(unsigned char command);
void LCD_putc(unsigned char ascii);
void LCD_puts(unsigned char *lcd_string);
void LCD_init();
/*******************************************************
 * Sources                                      *
 *******************************************************/
void LCD_delay(unsigned char ms)
{
        unsigned char n;
        unsigned int i;
        for (n=0; n<ms; n++)
        {
                for (i=0; i<LCD_DELAY; i++); /* For 1 ms */
        }

}
void LCD_enable()
{
   LCD_en = 0; /* Clear bit P2.4 */
   LCD_delay(1);
   LCD_en = 1; /* Set bit P2.4 */
}
void LCD_command(unsigned char command)
{
   LCD_rs = 0; /* Clear bit P2.5 */
   P2 = (P2 & 0xF0)|((command>>4) & 0x0F);
   LCD_enable();
   P2 = (P2 & 0xF0)|(command & 0x0F);
   LCD_enable();
   LCD_delay(1);
}
void LCD_putc(unsigned char ascii)
{
   LCD_rs = 1; /* Set bit P2.5 */
```

```
   P2 = (P2 & 0xF0)|((ascii>>4) & 0x0F);
   LCD_enable();
   P2 = (P2 & 0xF0)|(ascii & 0x0F);
   LCD_enable();
   LCD_delay(1);
}
void LCD_puts(unsigned char *lcd_string)
{
        while (*lcd_string)
        {
                LCD_putc(*lcd_string++);
        }
}
void LCD_init()
{
   LCD_en = 1; /* Set bit P2.4 */
   LCD_rs = 0; /* Clear bit P2.5 */
   LCD_command(0x28);
        LCD_delay(2);
   LCD_command(0x0C);
        LCD_delay(2);
   LCD_command(0x06);
        LCD_delay(2);
   LCD_command(0x01); /* Clear */
   LCD_delay(256);
}
```

Results:-

Experiment – 5

Write a program for interfacing a keypad to the microcontroller

Aim:
Write a program for interfacing a keypad to the microcontroller and the output is shown in the seven segment display whenever a key is pressed.

Introduction:
In this experiment a program is written in C language to read data from the keypad and the data is sent to the seven segment display on the board.

The program flow chart

- Write a program for interfacing keypad and the output is shown in the seven segment display when ever a key is pressed.
- Compile the program and generate Hex file.
- The generated hex file will be downloaded in to the MC and verify the result.

KIEL software is used for compilation of C files and the hex file can be generated.

Experimental procedure:

1. Open the kiel ide and you can see a menu.
2. To create a new project just click the project menu and click new microvision project
3. Then a dialog box will be opened to give the destination as shown in the figure. You can give the destination folder and save it.
4. To write a c file click the file menu and select the new document. The desired code is written in the document. And this document is needed to be added to the project.
5. In the project window on the left hand side you can see a project window .in the window a folder called target can be seen to that folder another folder can be seen as target source group1. the c file which is written nedded to be added to that folder . to add a c file to that folder right click the source group1 folder and a menu will be appeared.
6. Before bulding the project some settings are to made, from the main menu click the flash, you can see the option as **configure flash tools** . you click that option.From that menu click output and select an option called **create hex file**. And in the target menu give the oscillator frequency as **11.0592.**
7. Now all the hard work we have done to build the project and to create the hex file.From the main menu select the project and in that an option called **build target** can be seen , just click that . if all the coding is correct and the syntaxes are correct an hex file will be generated.

Down loading the hex file to the target:

Follow the same steps as mentioned in Ex 1-a for downloading the hex file into the microcontroller

C SOURCE CODE

```
#include <reg51.h>
#include"lcd.h"
sbit R1 = P1^0;
sbit R2 = P1^1;
sbit R3 = P1^2;
sbit R4 = P1^3;
sbit C1 = P1^4;
sbit C2 = P1^5;
sbit C3 = P1^6;
sbit C4 = P1^7;
 unsigned char key;
void Delay(unsigned int);
void delay()
{
        unsigned int i;
        for (i=0; i<10; i++); /* For 1 ms */
}
unsigned char  READ_SWITCHES (void)                    // initialize the port for inputs
        {
                                // P1.0 to p1.3 are outpot; and P1.4 to P1.7 are inputs
                                    // the keybad is connected to port 1
                R4=1;           //  make all rows = 1
                R3=1;
                R2=1;
                R1=0;
                //test row 1
                if (C1 == 0){   // key 1 is presed
                    delay();                    //depounce
                   while (C1==0);               //wait until release the key
                        return 1;                       //
                }
                if (C2 == 0){           //key 2 is pressed
                   delay();             //depounce
                        while (C2==0);          //wait until release the key
                        return 2;
           }
        if (C3 == 0){           //key 3 is pressed
                  delay();                      //depounce
             while (C3==0);             //wait until release the key
                   return 3;
                }
        if (C4 == 0){             //key 4 is pressed
                        delay();                    //depounce
                while (C4==0);          //wait until release the key
                   return 4;
                }
        //test row 2
```

```
R4=1;
R3=1;
R1=1;
R2=0;
if (C1 == 0){                 //key 5 is pressed
   delay();                         //depounce
        while (C1==0);                  //wait until release the key
        return 5;
}
if (C2 == 0) {                  //key 6 is pressed
        delay();                         //depounce
        while (C2==0);                  //wait until release the key
        return 6;
}
if (C3 == 0){                   //key 6 is pressed
   delay();                   //depounce
        while (C3==0);                  //wait until release the key
        return 7;
}
if (C4 == 0) {                   //key 7 is pressed
        delay();                         //depounce
        while (C4==0);                  //wait until release the key
        return 8;
}
//test row 3
R4=1;
R1=1;
R2=1;
R3=0;
if (C1 == 0) {                //key 8 is pressed
        delay();                    //depounce
        while (C1==0);                  //wait until release the key
        return 9;
}

if (C2 == 0) {                  //key 9 is pressed
        delay();                         //depounce
        while (C2==0);                  //wait until release the key
return 10;
}
if (C3 == 0) {              //key A is pressed
        delay();                  //depounce
        while (C3==0);                  //wait until release the key
        return 11;
}
if (C4 == 0) {                  //key B is pressed
        delay();                  //depounce
        while (C4==0);                  //wait until release the key
```

```
                return 12;
        }

                //test row 4
        R1=1;
        R2=1;
        R3=1;
        R4=0;
        if (C1 == 0) {                  //key C is pressed
                delay();                  //depounce
                while (C1==0);                  //wait until release the key
                return 13;
        }
        if (C2 == 0){                   //key D is pressed
                delay();                  //depounce
                while (C2==0);                  //wait until release the key
                return 14;
        }
    if (C3 == 0){                   //key E is pressed
                delay();                  //depounce
                while (C3==0);                  //wait until release the key
                return 15;
        }
        if (C4 == 0) {              //key C is pressed
                delay();                   //depounce
                while (C4==0);                  //wait until release the key
                return 16;
        }
    return 0;                                   // Means no key has been pressed
    }
void main (void)
 {
        P1 =0xf0;
        P3=0x00;
   LCD_init();
   LCD_row1();
   LCD_puts("< SERIAL KEYPAD >");
   LCD_row2();
   LCD_puts("  INTERFACING ");
   Delay(3);
   LCD_clear();
   LCD_row1();
   LCD_puts("< SERIAL KEYPAD >");
         LCD_row2();
   LCD_puts("KEY : ");
       while(1){
        key=READ_SWITCHES();
       if(key){
```

```
            P3 = key-1;
        LCD_clear();
            LCD_row1();
    LCD_puts("< SERIAL KEYPAD >");
            LCD_row2();
    LCD_puts("KEY : ");
      switch(key)
    {
        case 1:
            {
                        LCD_putc('0');
                        break;
                    }

            case 2:
                    {
                        LCD_putc('1');
                        break;
                    }

            case 3:
                    {
                        LCD_putc('2');
                        break;
                    }

            case 4:
                    {
                        LCD_putc('3');
                        break;
                    }
            case 5:
                    {
                        LCD_putc('4');
                        break;
                    }

            case 6:
                    {
                        LCD_putc('5');
                        break;
                    }

           case 7:
                    {
                        LCD_putc('6');
                        break;
                    }
```

```
case 8:
        {
                LCD_putc('7');
                break;
        }

case 9:
        {
                LCD_putc('8');
                break;
        }

case 10:
        {
                LCD_putc('9');
                break;
        }

case 11:
        {
                LCD_putc('A');
                break;
        }

case 12:
        {
                LCD_putc('B');
                break;
        }

case 13:
        {
                LCD_putc('C');
                break;
        }

case 14:
        {
                LCD_putc('D');
                break;
        }

case 15:
        {
                LCD_putc('E');
                break;
        }
```

```
                case 16:
                                {
                                        LCD_putc('F');
                                        break;
                                }

                }// switch

    }//if

  }//while


}//main
void Delay(unsigned int duration)
 {
   unsigned int r2;
         for (r2 = 0; r2<= duration;r2++){
                                LCD_delay(250);
                                LCD_delay(250);
                        }
 }
/*
 * Filename : lcd.h
 * Hardware : Controller  -> P89V51RD2
 *          XTAL       -> 11.0592 MHz
 *
 * I/O     : RS         -> P2.5
 *          Enable     -> P2.4
 *          Data4567    -> P2.0,P2.1,P2.2,P2.3 */


 #define LCD_DELAY 150 /* Delay for 1 ms */
 #define LCD_clear() LCD_command(0x1)  /* Clear display LCD */
 #define LCD_origin() LCD_command(0x2)/* Set to origin LCD */
 #define LCD_row1() LCD_command(0x80)          /* Begin at Line 1 */
 #define LCD_row2() LCD_command(0xC0)   /* Begin at Line 2 */
 /*****************************************************
 * Prototype(s)                                      *
 *****************************************************/
sbit  LCD_en=P2^4;
sbit  LCD_rs=P2^5;
void LCD_delay(unsigned char ms);
void LCD_enable();
void LCD_command(unsigned char command);
void LCD_putc(unsigned char ascii);
void LCD_puts(unsigned char *lcd_string);
```

```
void LCD_init();
/*****************************************************
 * Sources                                           *
 *****************************************************/
void LCD_delay(unsigned char ms)
{
        unsigned char n;
        unsigned int i;
        for (n=0; n<ms; n++)
        {
                for (i=0; i<LCD_DELAY; i++); /* For 1 ms */
        }

}
void LCD_enable()
{
   LCD_en = 0; /* Clear bit P2.4 */
   LCD_delay(1);
   LCD_en = 1; /* Set bit P2.4 */
}

void LCD_command(unsigned char command)
{
   LCD_rs = 0; /* Clear bit P2.5 */
   P2 = (P2 & 0xF0)|((command>>4) & 0x0F);
   LCD_enable();
   P2 = (P2 & 0xF0)|(command & 0x0F);
   LCD_enable();
   LCD_delay(1);
}

void LCD_putc(unsigned char ascii)
{
   LCD_rs = 1; /* Set bit P2.5 */
   P2 = (P2 & 0xF0)|((ascii>>4) & 0x0F);
   LCD_enable();
   P2 = (P2 & 0xF0)|(ascii & 0x0F);
   LCD_enable();
   LCD_delay(1);
}
void LCD_puts(unsigned char *lcd_string)
{
        while (*lcd_string)
        {
                LCD_putc(*lcd_string++);
        }
}
void LCD_init()
```

```
{
   LCD_en = 1; /* Set bit P2.4 */
   LCD_rs = 0; /* Clear bit P2.5 */
   LCD_command(0x28);
         LCD_delay(2);
   LCD_command(0x0C);
         LCD_delay(2);
   LCD_command(0x06);
         LCD_delay(2);
   LCD_command(0x01); /* Clear */
   LCD_delay(256);
}
```

Results:-

Experiment – 6

Program to Interface Seven Segment Display Unit.

Solution:

```
#include<reg51.h>
delay_ms(int time)              // Time delay function
{
        int i,j;
        for(i=0;i<time;i++)
        for(j=0;j<1275;j++);
}

void main()
{
        char num[]={0x40,0xF9,0x24,0x30,0x19,0x12,0x02,0xF8,0x00,0x10}; // Hex values
corresponding to digits 0 to 9
        int c;
        while(1)
        {
                for(c=0;c<10;c++)
                {
                P2=num[c];
                delay_ms(200);
                }
        }
}
```

Results:-

Experiment – 7

Program to transmit a message from Microcontroller to PC serially using RS232

Aim:
Write a program for serial communication using RS232 interface and communicate with PC using polling mode.

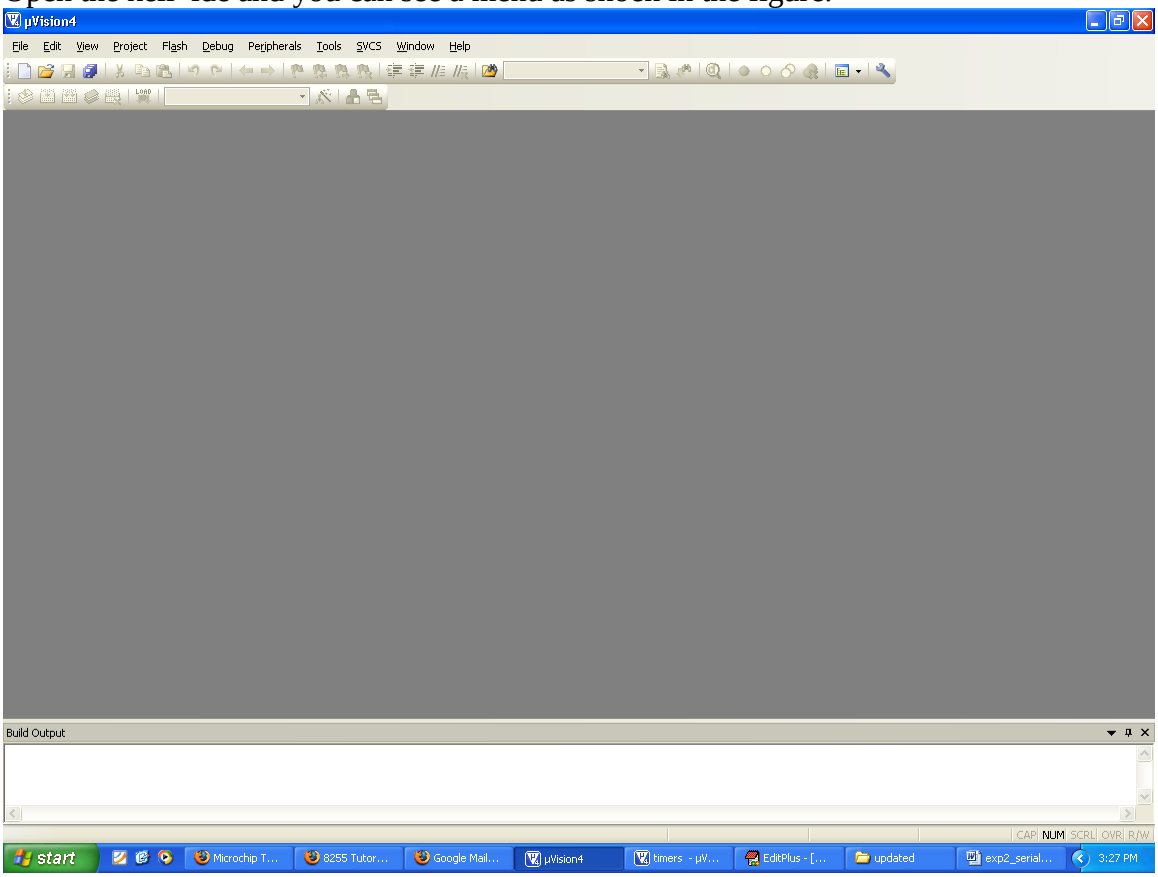
Introduction:
In this experiment a program is written in C language to read data from PC and the same data is echoed back to the computer through RS232 interface available on the board.

The program flow chart

- Write a program for serial communication using polling method.
- Compile the program and generate Hex file.
- The generated hex file will be downloaded in to the MC and verify the result.

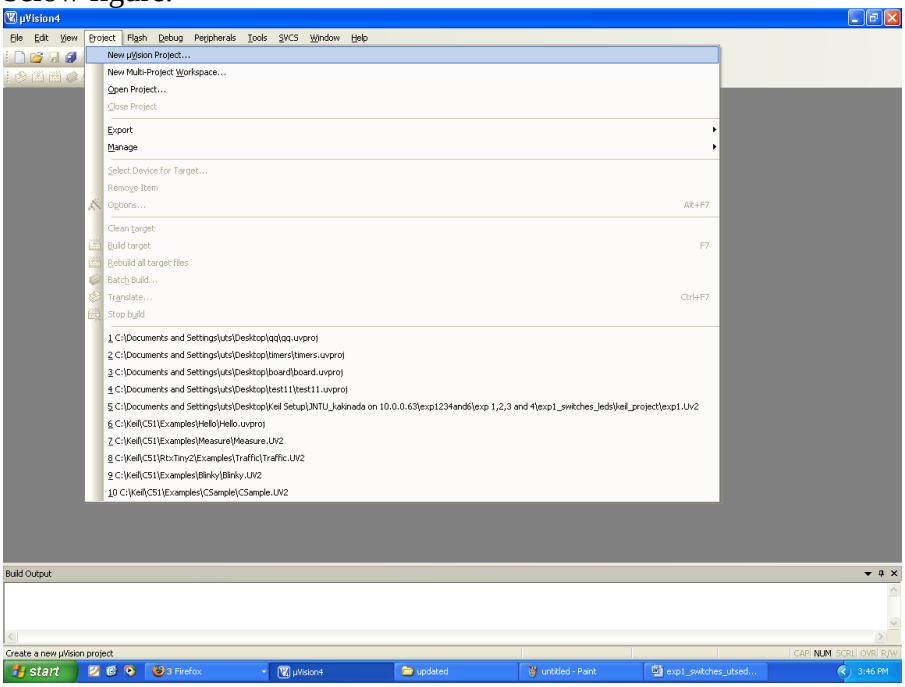
KIEL software is used for compilation of C files and the hex file can be generated.

Experimental procedure:
Step1:
Open the keil ide and you can see a menu as shoen in the figure.



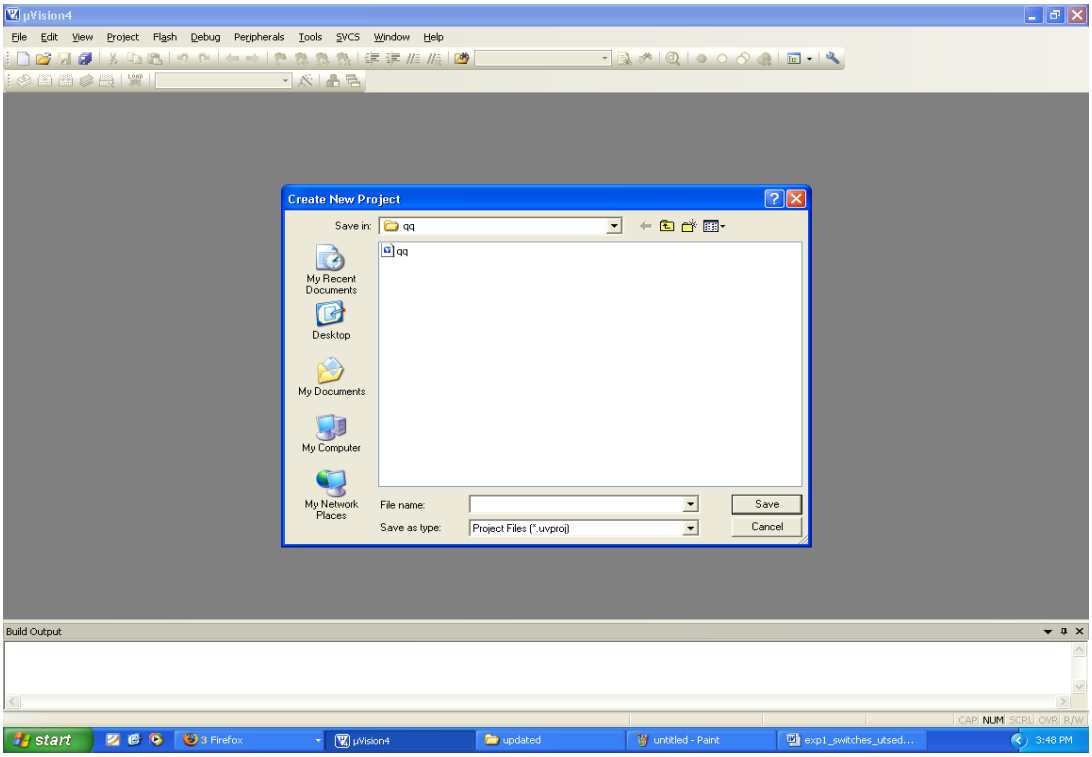
Step 2:

To create a new project just click the project menu and click new microvision project , as shown in below figure.



Step 3:

Then a dialog box will be opened to give the destination as shown in the figure. You can give the destination folder and save it.

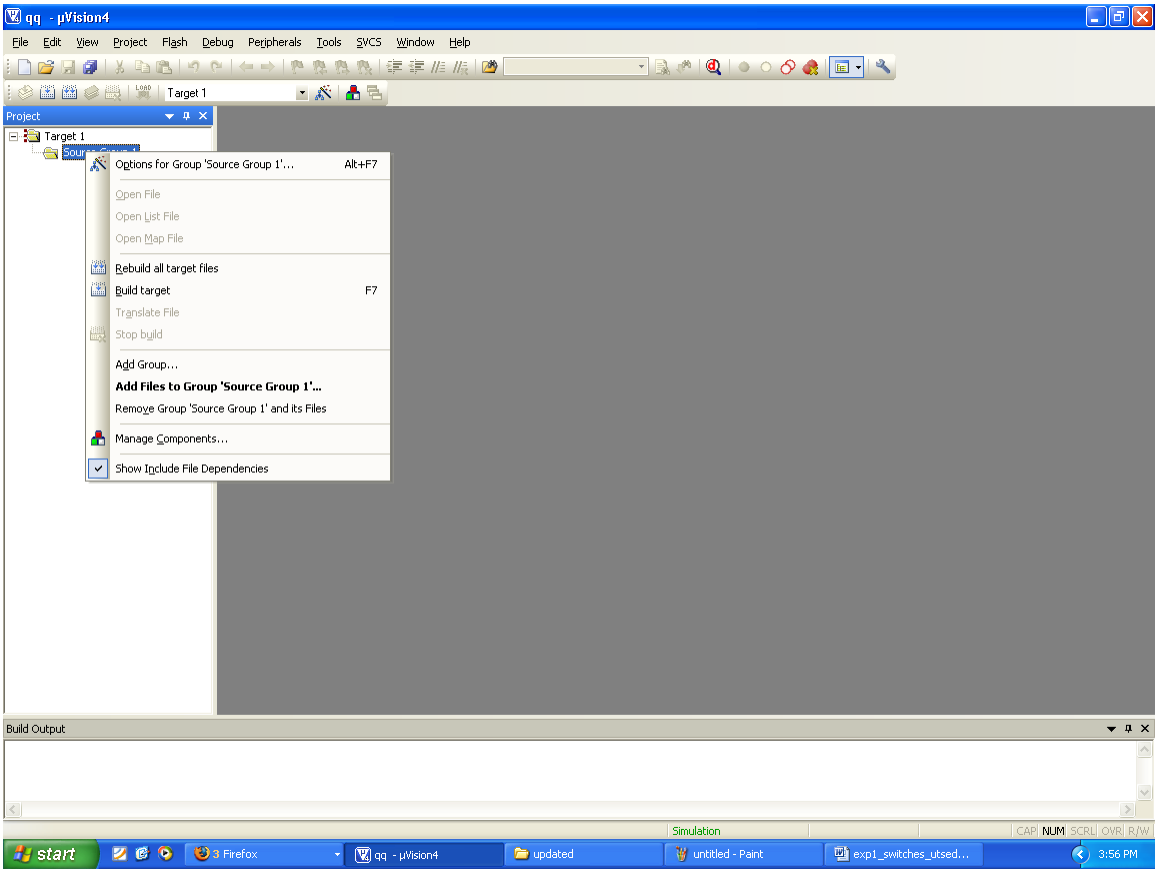


Step 4:

To write a c file click the file menu and select the new document. The desired code is written in the document. And this document is needed to be added to the project.

Step 5:

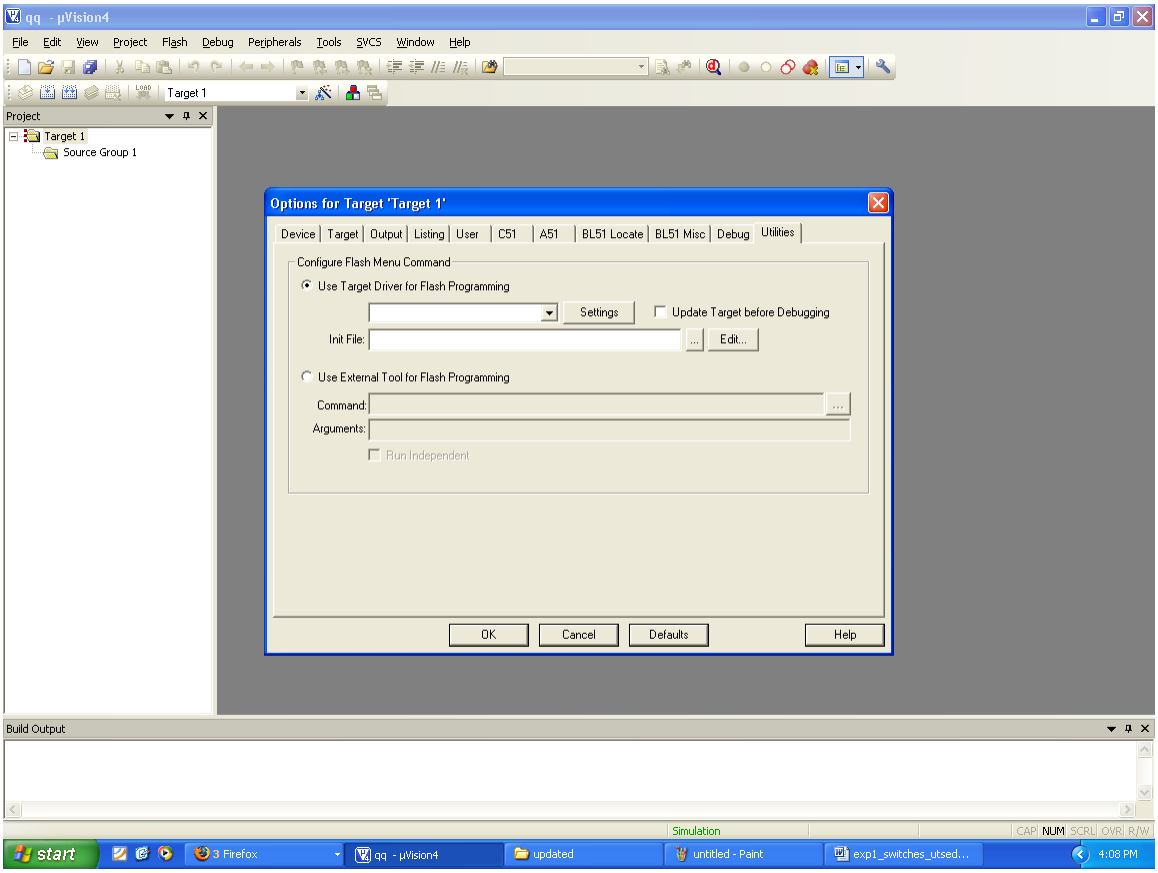
In the project window on the left hand side you can see a project window .in the window a folder called target can be seen to that folder another folder can be seen as target source group1. the c file which is written nedded to be added to that folder . to add a c file to that folder right click the source group1 folder and a menu will be appeared as shown in the figure.



From that menu you can see **add files to source group** option. If you click that option the c file will be added to the project.

Step 6:

Before bulding the project some settings are to made, from the main menu click the flash, you can see the option as **configure flash tools** . you click that option you can find a menu as shown in the figure.



From that menu click output and select an option called **create hex file**. And in the target menu give the oscillator frequency as **11.0592.**

Step 7:

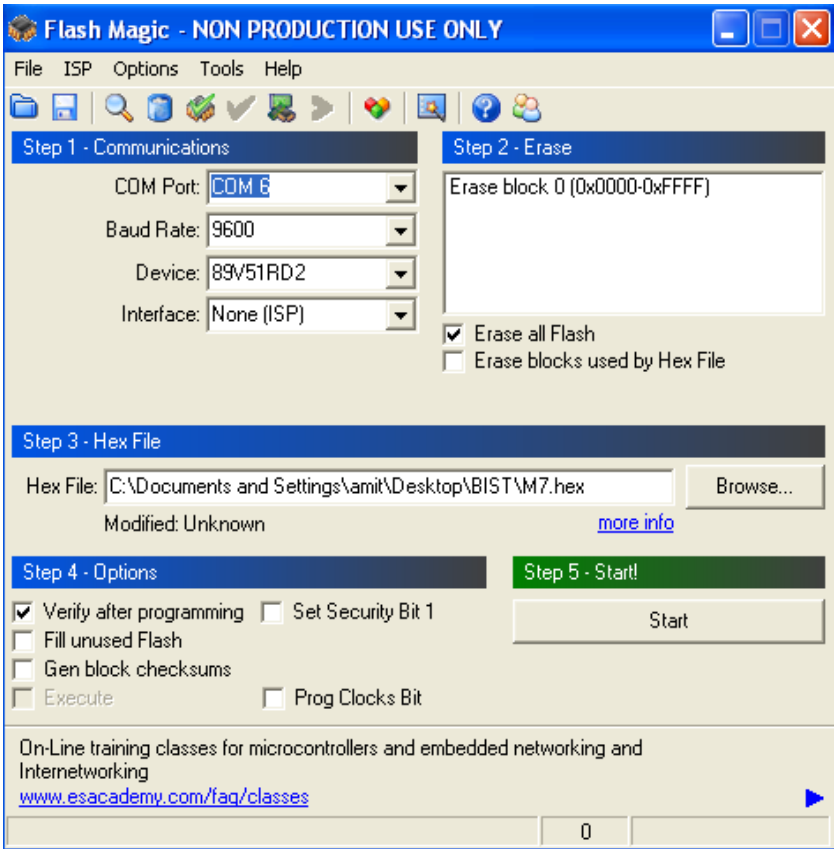
Now all the hard work we have done to build the project and to create the hex file

From the main menu select the project and in that an option called **build target** can be seen , just click that . if all the coding is correct and the syntaxes are correct an hex file will be generated.

Down loading the hex file to the target:

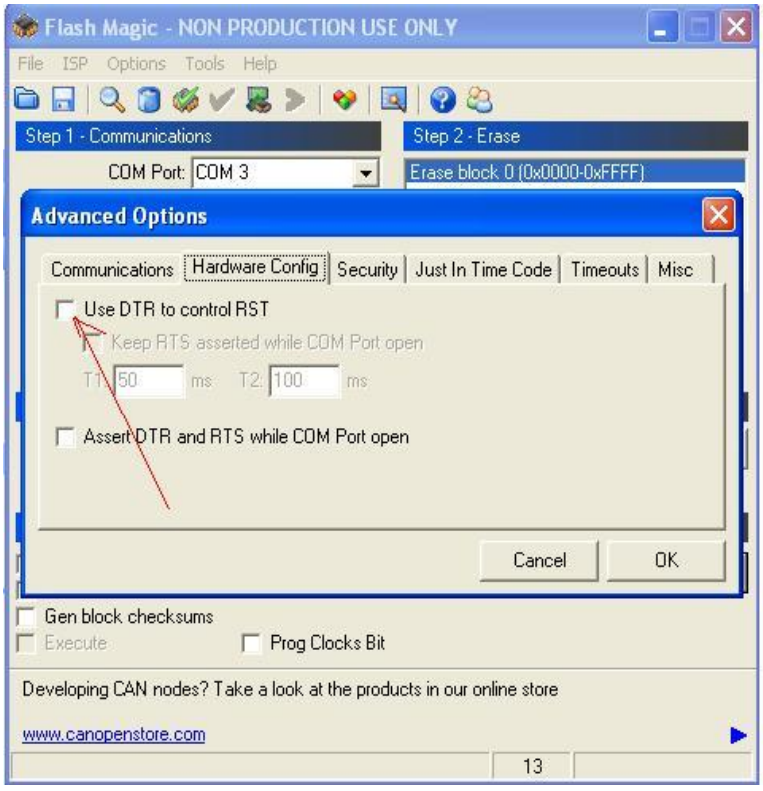
- Open the Flash Magic tool for downloading into the Microcontroller Board. Click on Device menu select option you will be popped up with a window named **choose device.** Under choose device options select 8051 and click on Ok button to open flash magic tool to download the hex file in to the MC and is as shown below.

(1) Flash Magic Opens as shown in the figure



- Change setting for Flash Magic as shown in the above figure

COM Port Choose the Appropriate COM Port of the computer. Generally it is COM1
Baud Rate 9600
Device 89V51RD2
Interface None(ISP)
Oscillator Freq (MHZ) 11.0592
Tick Erase All Flash+Security+Clks
Tick Verify after Programming
Now Goto
Option-> Advanced Options → Hardware Configuration
and uncheck the "Use DTR To Control Reset" as shown in the figure below

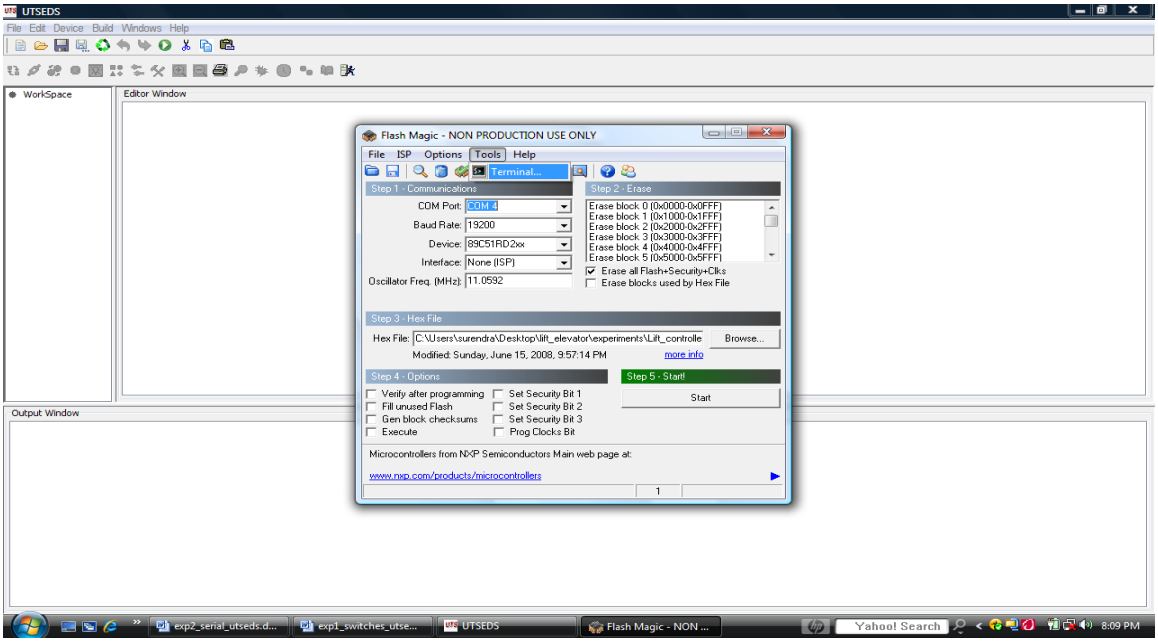


- Browse for Hex file to be downloaded into the microcontroller

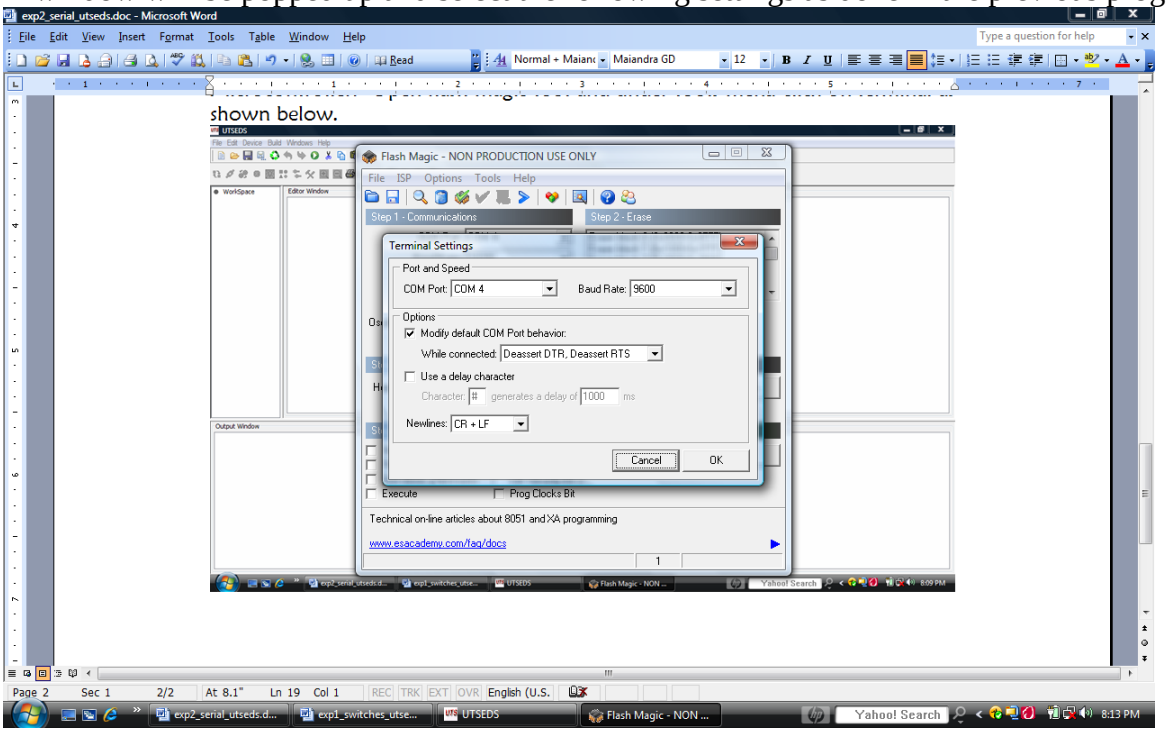
- Select the **switches.Hex** file from the Folder in which switches.c file is located.

- Turn On the Board Power. The RUN/PROG button can be any where (as 89V51RD2)
- Connect the serial port to Board from Computer
- Press and hold the reset button on board and press start button on flash magic. wait until a small window comes with message "Reset The Device Into ISP Mode "comes. Once this window comes release the reset button.
- You can see the messages in flash magic tool as **Erasing the Device, Programming the Device ,Verifying and Finished**
- (10)Once you get finished message in the flash magic. press reset once to enter in run mode.

Results/Output verification: Now serial communication program is running on Microcontroller. Open flash magic tool and under tools menu click on terminal as shown below.



A window will be popped up and select the following settings as done in the previous program .



You can observe a black window appearing having two parts. One is output and another one is input to communicate to the serial port. Press reset button and you can see a message saying “press any key” under output pane. Now what ever the message you type at input pane, the same message will be displayed on the output pane.

C SOURCE CODE

```
        // This program reads the character from serial port and retransmits the same back to PC

#include <reg51.h>                      /* define 8051 registers */
void SendChar(unsigned char x);
void DisplayMesPC(unsigned char *);

unsigned char *mes;

void main (void)  {                     /* main program */

        unsigned char temp;
        TMOD = 0x20;   /* GATE OFF,C/#T = 0, M1 M0 = 10(8 BIT AUTO RELOAD) TIMER 0
,TIMER 1 IN MODE 2(AUTO RELOAD MODE)*/
        SCON = 0x50;  /* SERIAL PORT IN MODE2 8-BIT UART VARIABLE BAUDRATE */
      TH1  = 0xfd;  /* TIMER 1 FOR BAUD RATE GEN(9.6K)*/
      TR1  = 1;   /* baud rate timer start*/
        mes = "\r\nPress Any Key\r\n";
        DisplayMesPC(mes);

        while (1)
        {
        while (!RI);       // waiting for character from PC
        temp = SBUF;   // reading the character into temp variable from Serial Buffer

        RI = 0;                    // Resetting the Receive Flag
        SendChar(temp);         // Calling function for retransmitting the character back to PC
        //---------------------------------------SendChar(temp);          //       Calling       function       for
retransmitting the character back to PC
        }

}
void SendChar(unsigned char x)         // transmit function to send character to PC
{
 SBUF =x;                // wrting the character into the serial buffer
TI = 0;           // Clearing the Transmit empty flag

  while(!TI);    // wating for end of trasmission. after transmission the TI flag will set.

}

void DisplayMesPC(unsigned char *mes)
{
int counter;
for (counter=0;mes[counter]!='\0';counter++)
  {
    SendChar(mes[counter]);
```

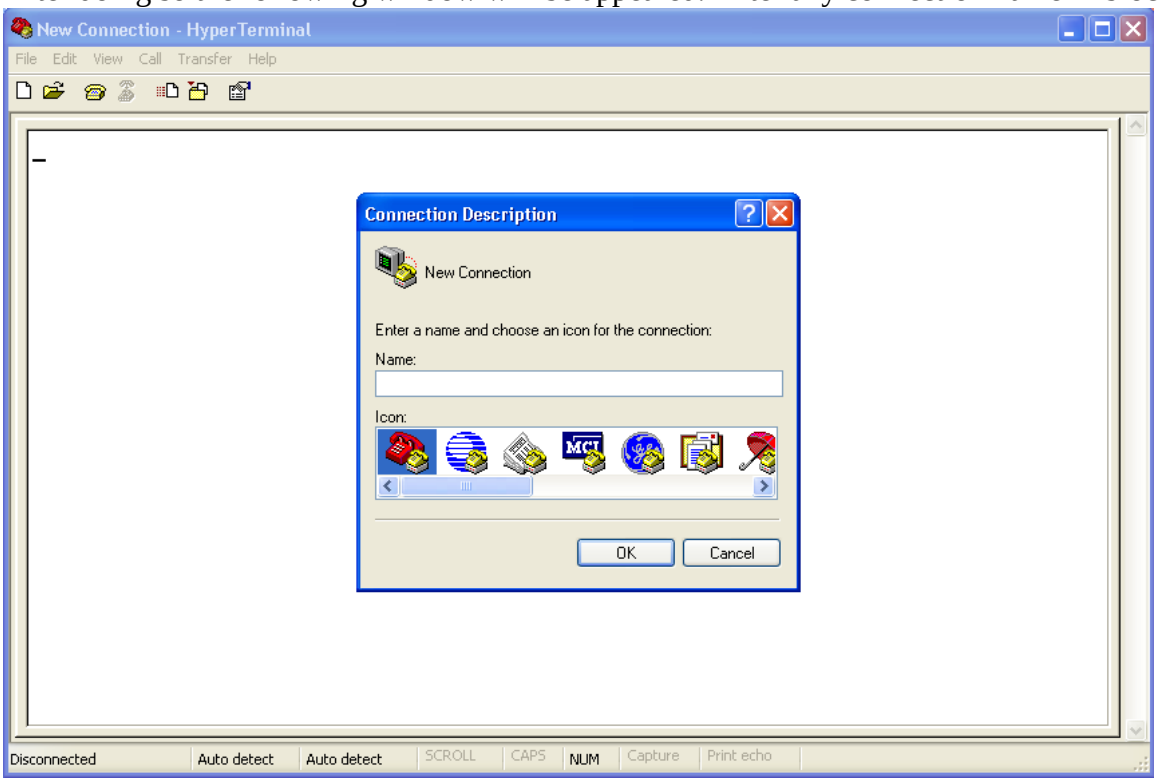
```
  }
}
```

HARWARE CONFIGURATION

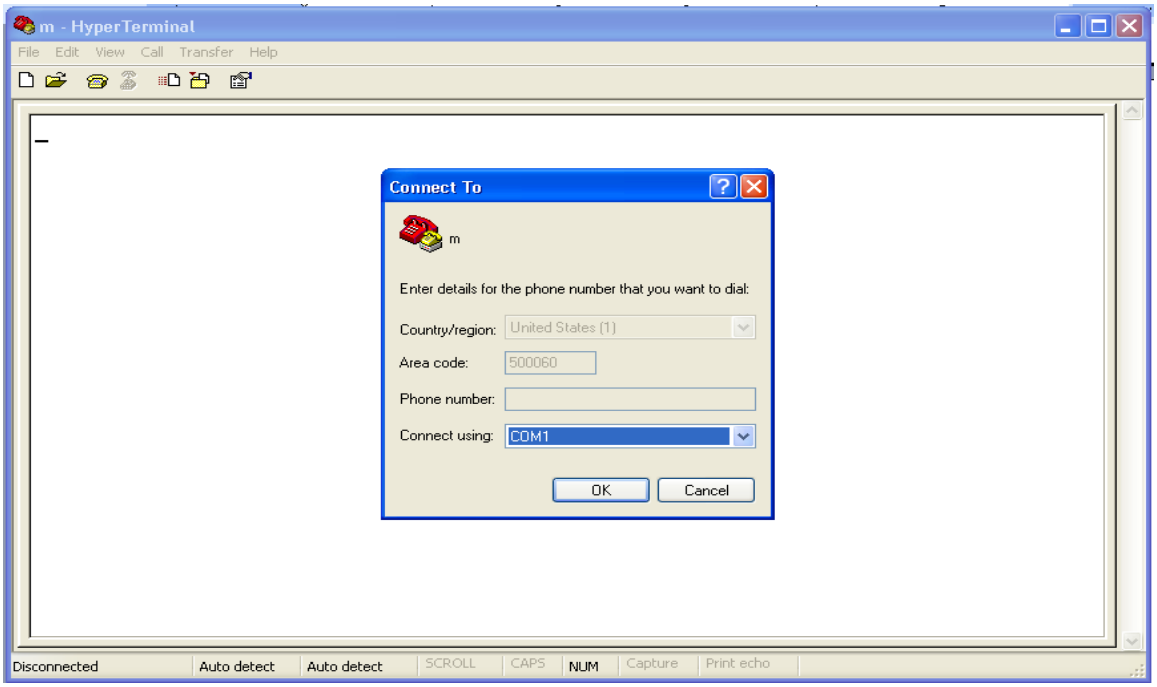
To test the board

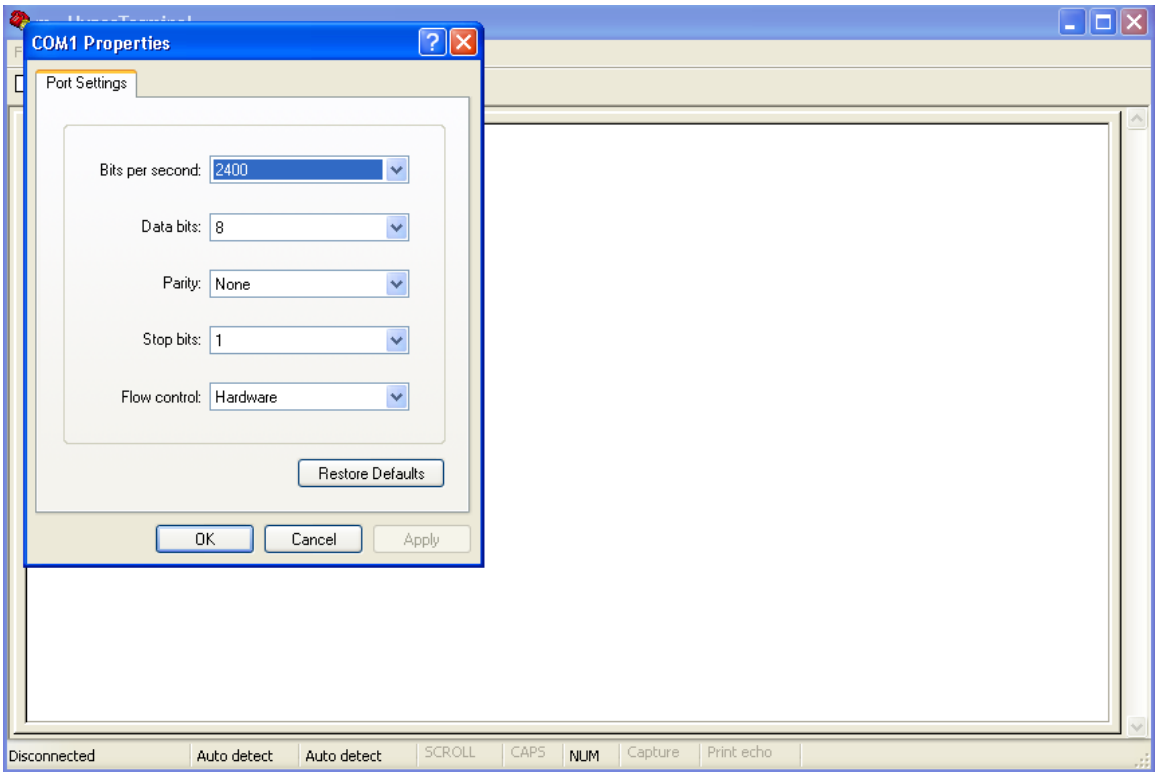
Power supply cable should be connected to the 8051 Development board and serial cable must be connected to the computer. Now go to the start menu and in that go to the all programs menu . in that go to the accessories , there in the sub menu you will find a menu called communications. In the communications menu select the hyper terminal.

After doing so the following window will be appeared. Enter any connection name in block.

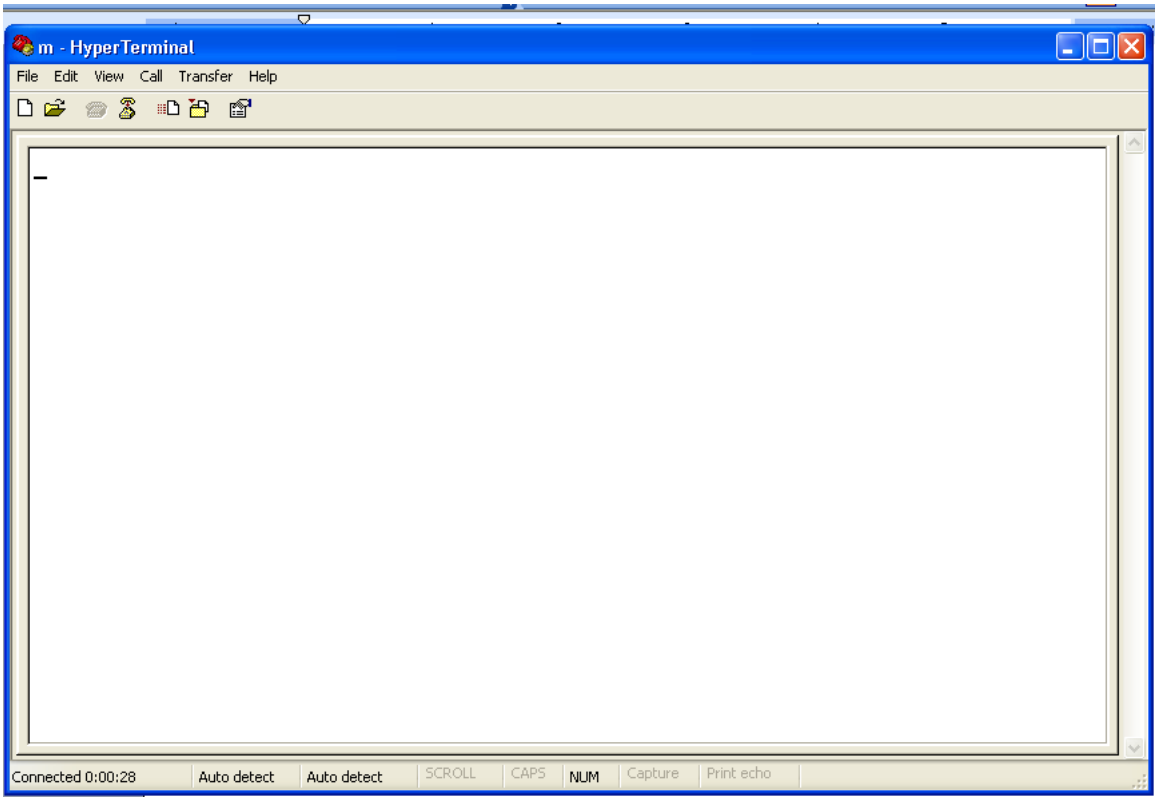


Select COM1 and then click ok





In that select the new connection and set the options as **restore to defaults.**
This will keep the default baud rate as 9600,1 start bit,1 stop bit and hardware as none.



To check the output now, if any key is pressed from the keyboard that will be displayed on the hyper terminal.

Results:-

Experiment – 8

Program to receive a message from PC serially using RS232.

Aim:
Writing a program for serial communication using RS232 interface and communicate with PC

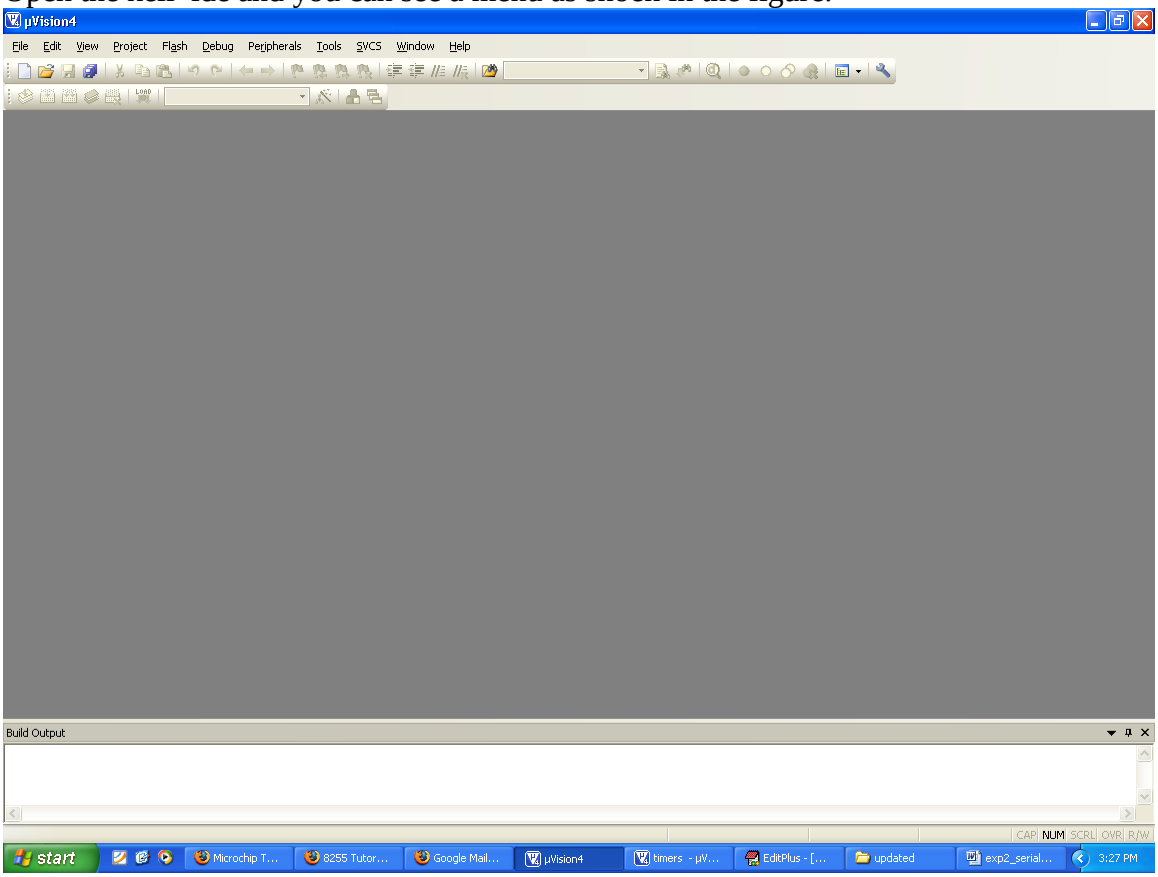
Introduction:
In this experiment a program is written in C language to read data from PC and the same data is echoed back to the computer through RS232 interface available on the board.

The program flow chart

- Write a program for serial communication using polling method.
- Compile the program and generate Hex file.
- The generated hex file will be downloaded in to the MC and verify the result.

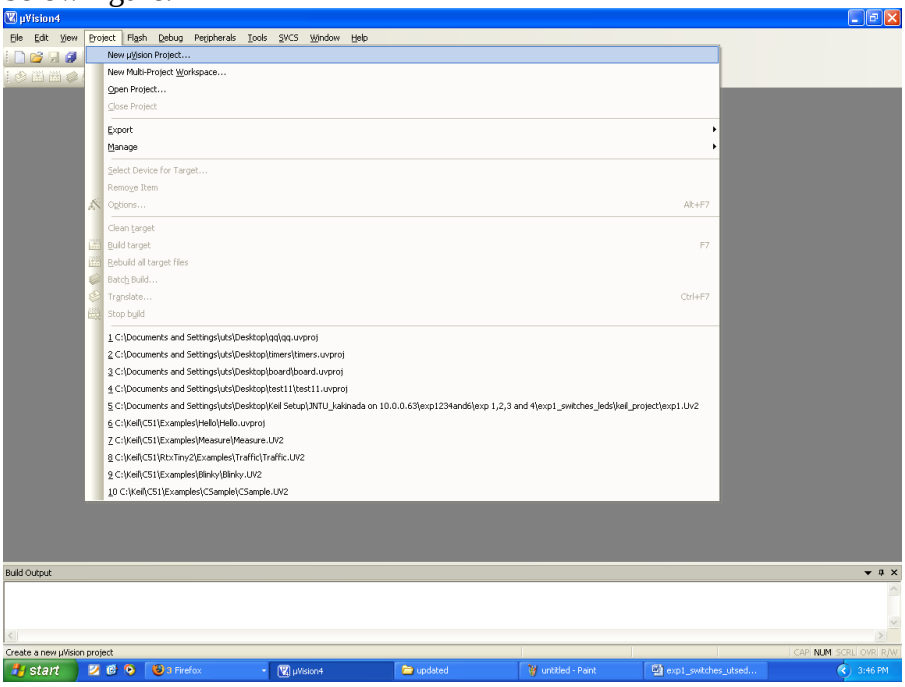
KIEL software is used for compilation of C files and the hex file can be generated.

Experimental procedure:
Step1:
Open the keil ide and you can see a menu as shoen in the figure.



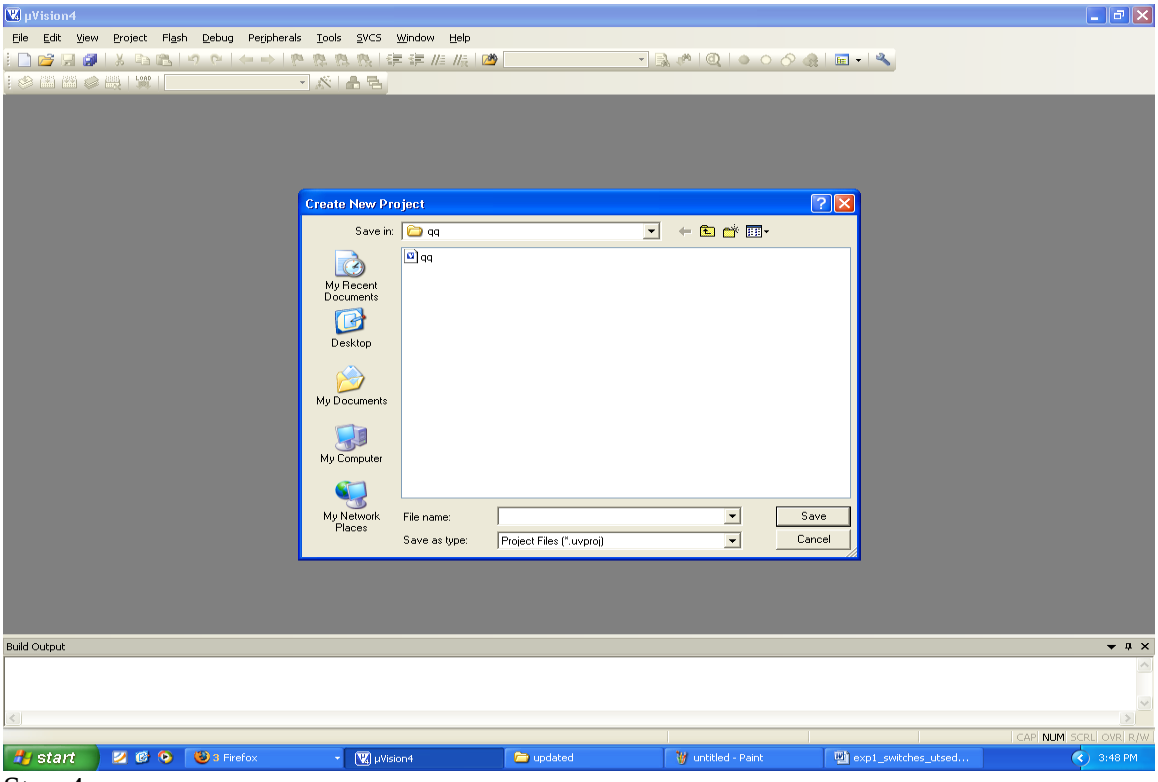
Step 2:

To create a new project just click the project menu and click new microvision project , as shown in below figure.



Step 3:

Then a dialog box will be opened to give the destination as shown in the figure. You can give the destination folder and save it.

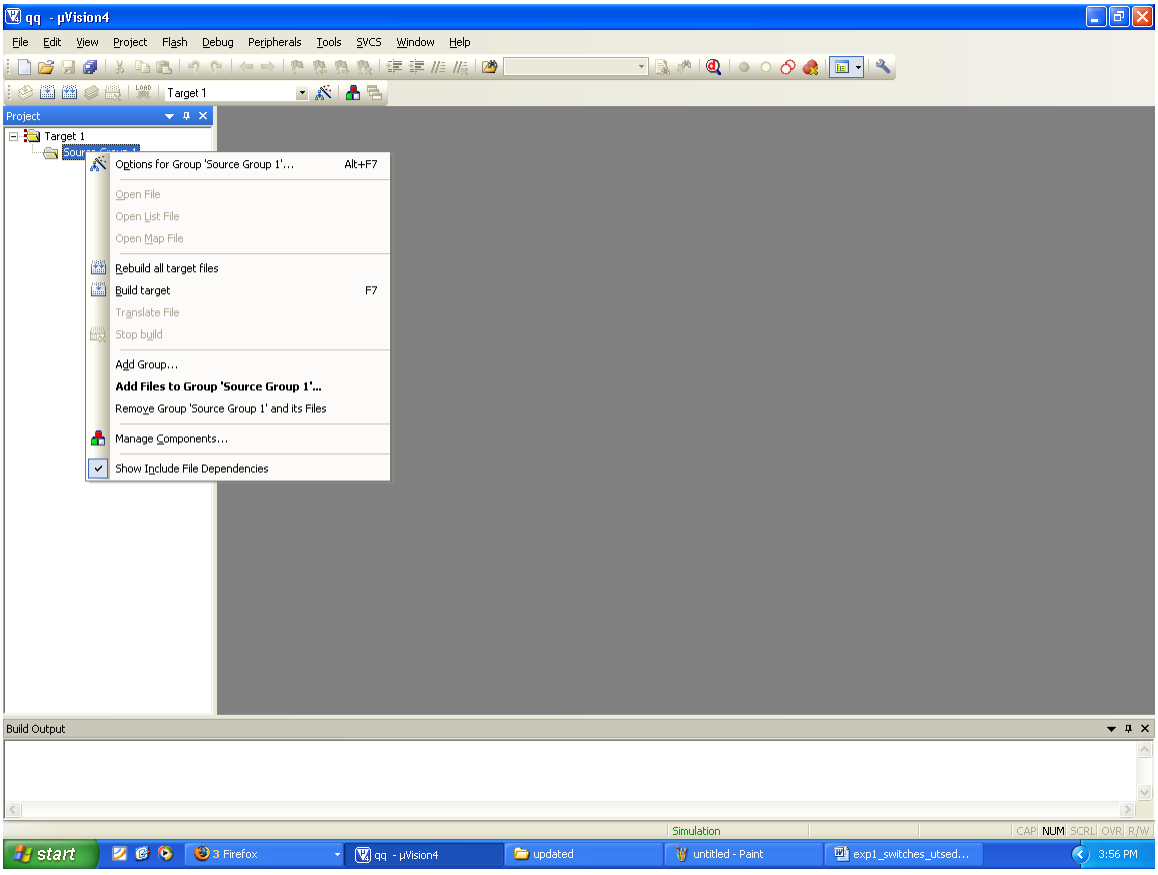


Step 4:

To write a c file click the file menu and select the new document. The desired code is written in the document. And this document is needed to be added to the project.

Step 5:

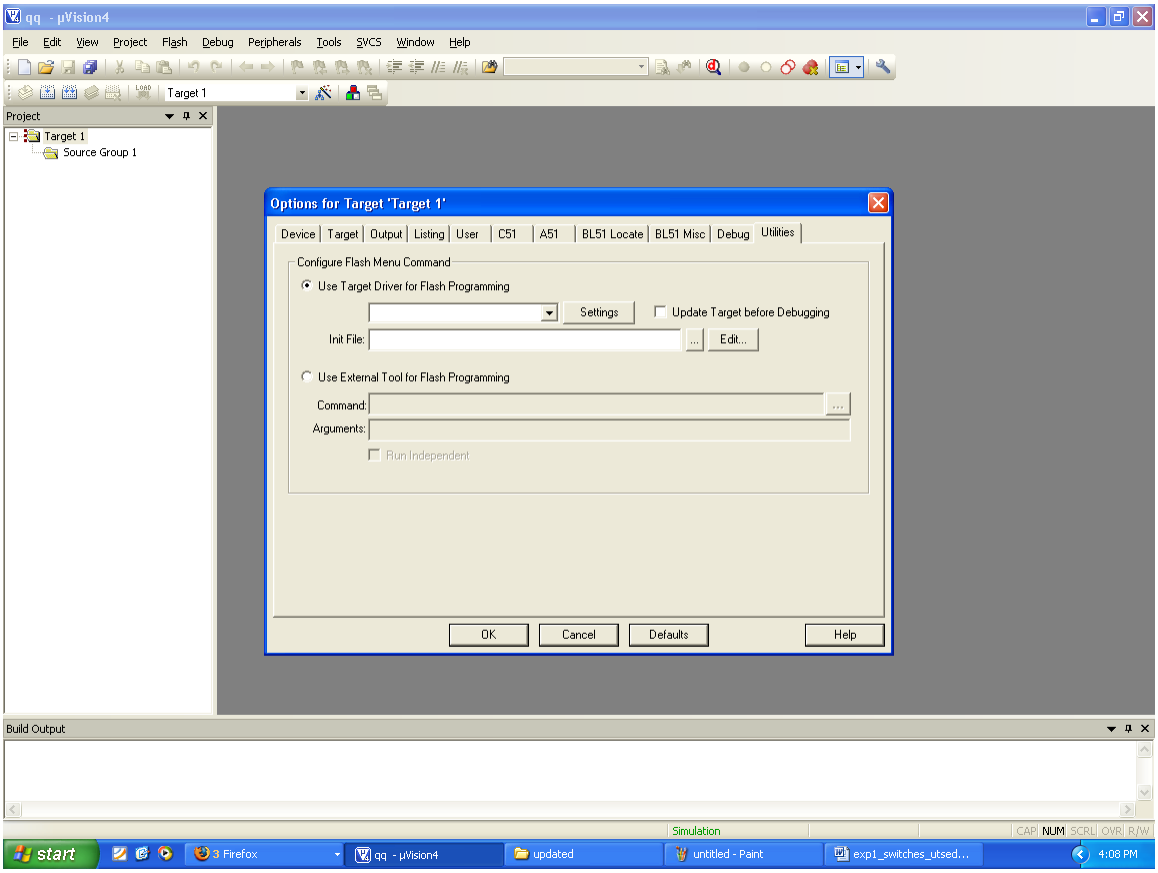
In the project window on the left hand side you can see a project window .in the window a folder called target can be seen to that folder another folder can be seen as target source group1. the c file which is written nedded to be added to that folder . to add a c file to that folder right click the source group1 folder and a menu will be appeared as shown in the figure.



From that menu you can see **add files to source group** option. If you click that option the c file will be added to the project.

Step 6:

Before bulding the project some settings are to made, from the main menu click the flash, you can see the option as **configure flash tools** . you click that option you can find a menu as shown in the figure.



From that menu click output and select an option called **create hex file**. And in the target menu give the oscillator frequency as **11.0592.**

Step 7:

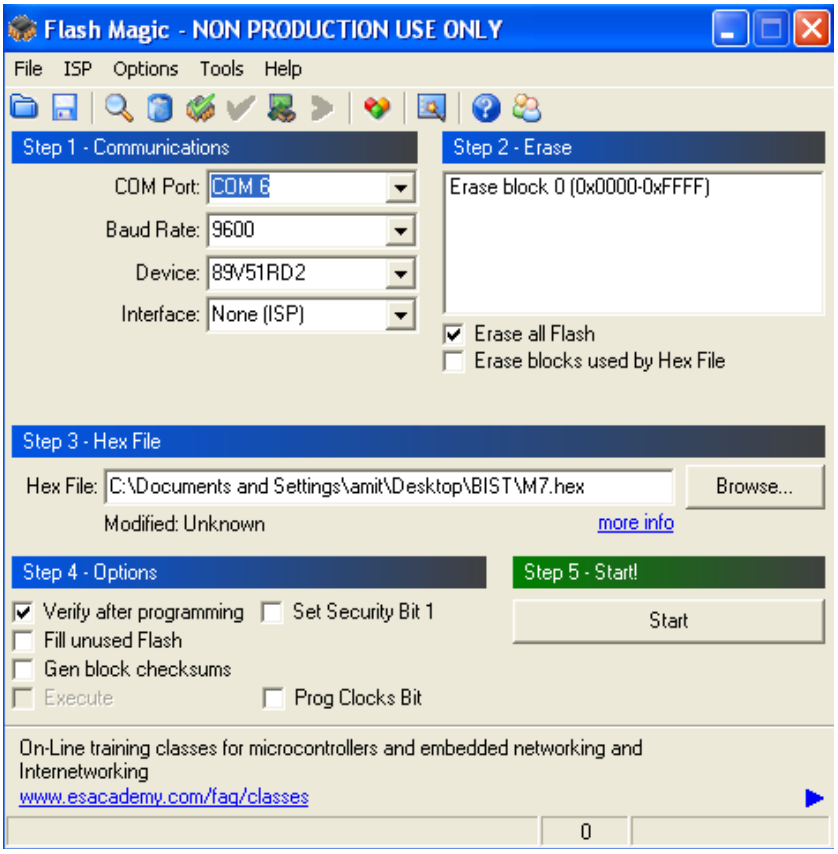
Now all the hard work we have done to build the project and to create the hex file

From the main menu select the project and in that an option called **build target** can be seen , just click that . if all the coding is correct and the syntaxes are correct an hex file will be generated.

Down loading the hex file to the target:

- Open the Flash Magic tool for downloading into the Microcontroller Board. Click on Device menu select option you will be popped up with a window named **choose device.** Under choose device options select 8051 and click on Ok button to open flash magic tool to download the hex file in to the MC and is as shown below.

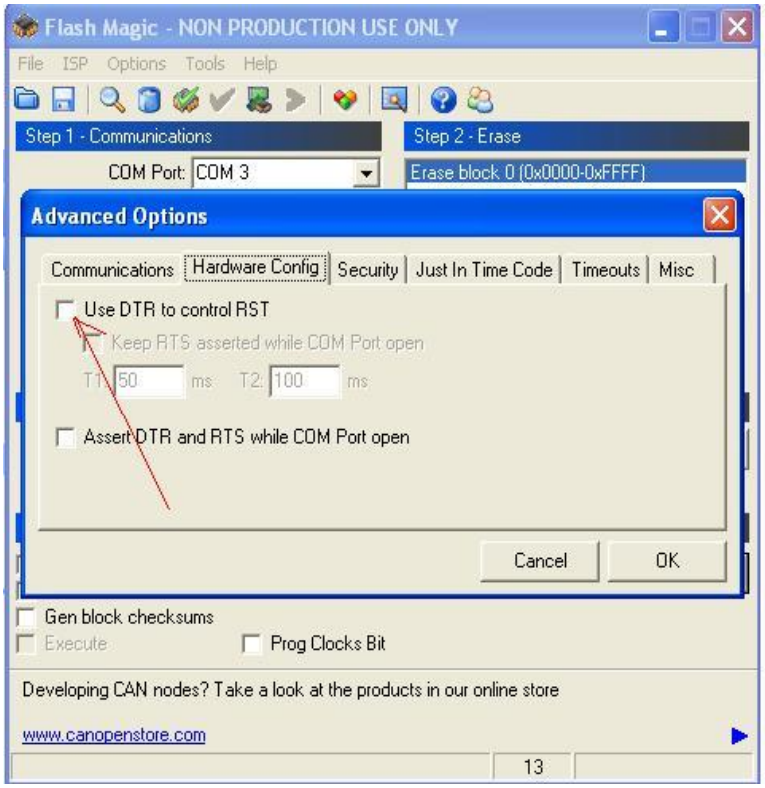
(2) Flash Magic Opens as shown in the figure



- Change setting for Flash Magic as shown in the above figure

COM Port Choose the Appropriate COM Port of the computer. Generally it is COM1
Baud Rate 9600
Device 89V51RD2
Interface None(ISP)
Oscillator Freq (MHZ) 11.0592
Tick Erase All Flash+Security+Clks
Tick Verify after Programming
Now Goto
Option-> Advanced Options → Hardware Configuration

and uncheck the “Use DTR To Control Reset” as shown in the figure below

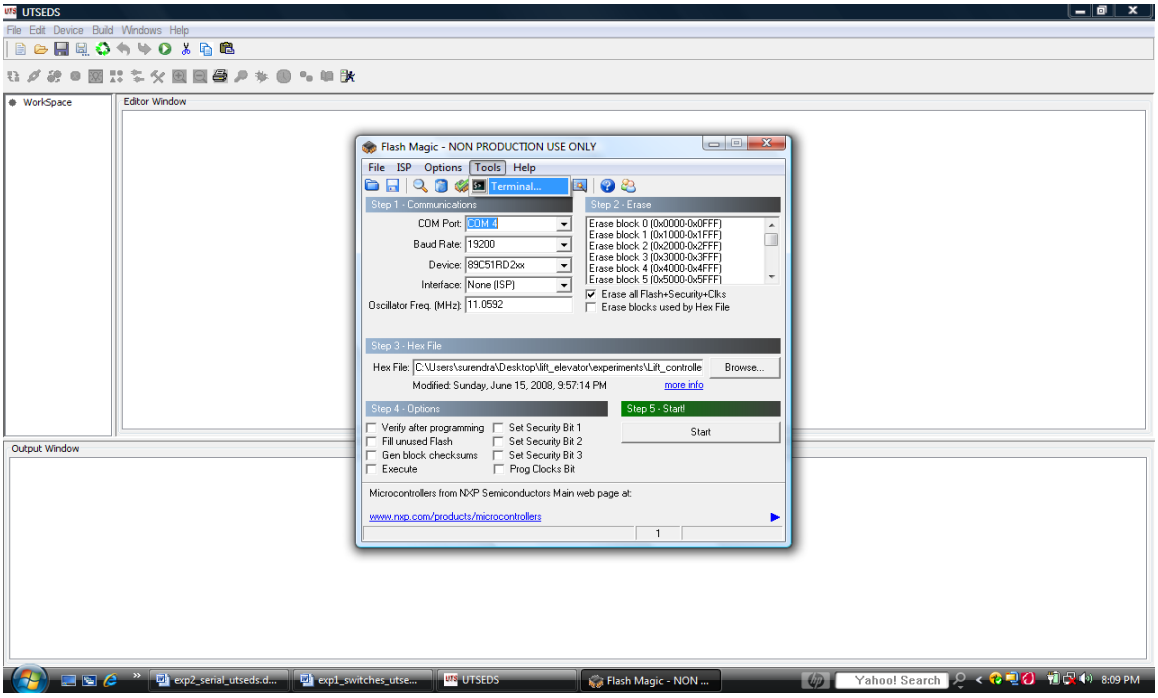


- Browse for Hex file to be downloaded into the microcontroller

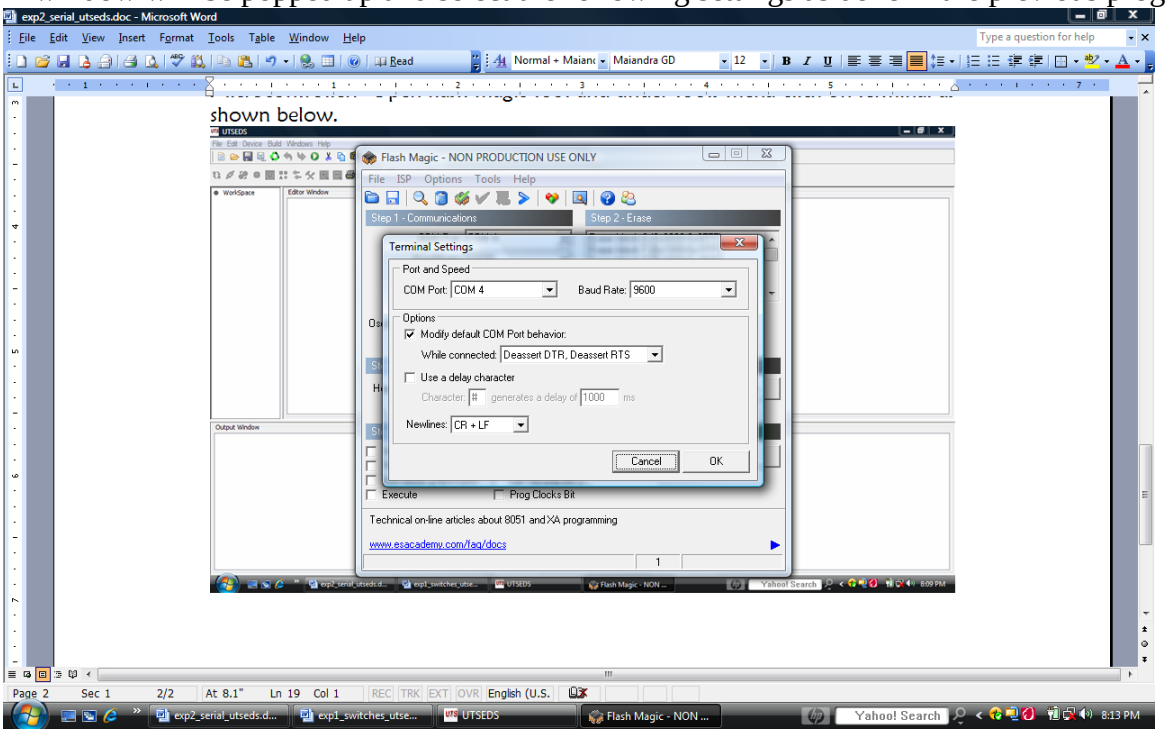
- Select the **switches.Hex** file from the Folder in which switches.c file is located.

- Turn On the Board Power. The RUN/PROG button can be any where (as 89V51RD2)
- Connect the serial port to Board from Computer
- Press and hold the reset button on board and press start button on flash magic. wait until a small window comes with message "Reset The Device Into ISP Mode "comes. Once this window comes release the reset button.
- You can see the messages in flash magic tool as **Erasing the Device, Programming the Device ,Verifying and Finished**
- (10)Once you get finished message in the flash magic. press reset once to enter in run mode.

Results/Output verification: Now serial communication program is running on Microcontroller. Open flash magic tool and under tools menu click on terminal as shown below.



A window will be popped up and select the following settings as done in the previous program .



You can observe a black window appearing having two parts. One is output and another one is input to communicate to the serial port. Press reset button and you can see a message saying “press any key” under output pane. Now what ever the message you type at input pane, the same message will be displayed on the output pane.

C SOURCE CODE

```
        // This program reads the character from serial port and retransmits the same back to PC

#include <reg51.h>                      /* define 8051 registers */
void SendChar(unsigned char x);
void DisplayMesPC(unsigned char *);

unsigned char *mes;

void main (void)  {                     /* main program */

        unsigned char temp;
        TMOD = 0x20;  /* GATE OFF,C/#T = 0, M1 M0 = 10(8 BIT AUTO RELOAD) TIMER 0
,TIMER 1 IN MODE 2(AUTO RELOAD MODE)*/
        SCON = 0x50;  /* SERIAL PORT IN MODE2 8-BIT UART VARIABLE BAUDRATE */
      TH1  = 0xfd;  /* TIMER 1 FOR BAUD RATE GEN(9.6K)*/
      TR1  = 1;  /* baud rate timer start*/
        mes = "\r\nPress Any Key\r\n";
        DisplayMesPC(mes);

        while (1)
        {
        while (!RI);      // waiting for character from PC
        temp = SBUF;  // reading the character into temp variable from Serial Buffer

        RI = 0;                 // Resetting the Receive Flag
        SendChar(temp);         // Calling function for retransmitting the character back to PC
        //---------------------------------------SendChar(temp);          //      Calling      function      for
retransmitting the character back to PC
        }

}
void SendChar(unsigned char x)        // transmit function to send character to PC
{
 SBUF =x;               // wrting the character into the serial buffer
TI = 0;          // Clearing the Transmit empty flag

  while(!TI);    // wating for end of trasmission. after transmission the TI flag will set.

}

void DisplayMesPC(unsigned char *mes)
{
int counter;
for (counter=0;mes[counter]!='\0';counter++)
  {
    SendChar(mes[counter]);
```

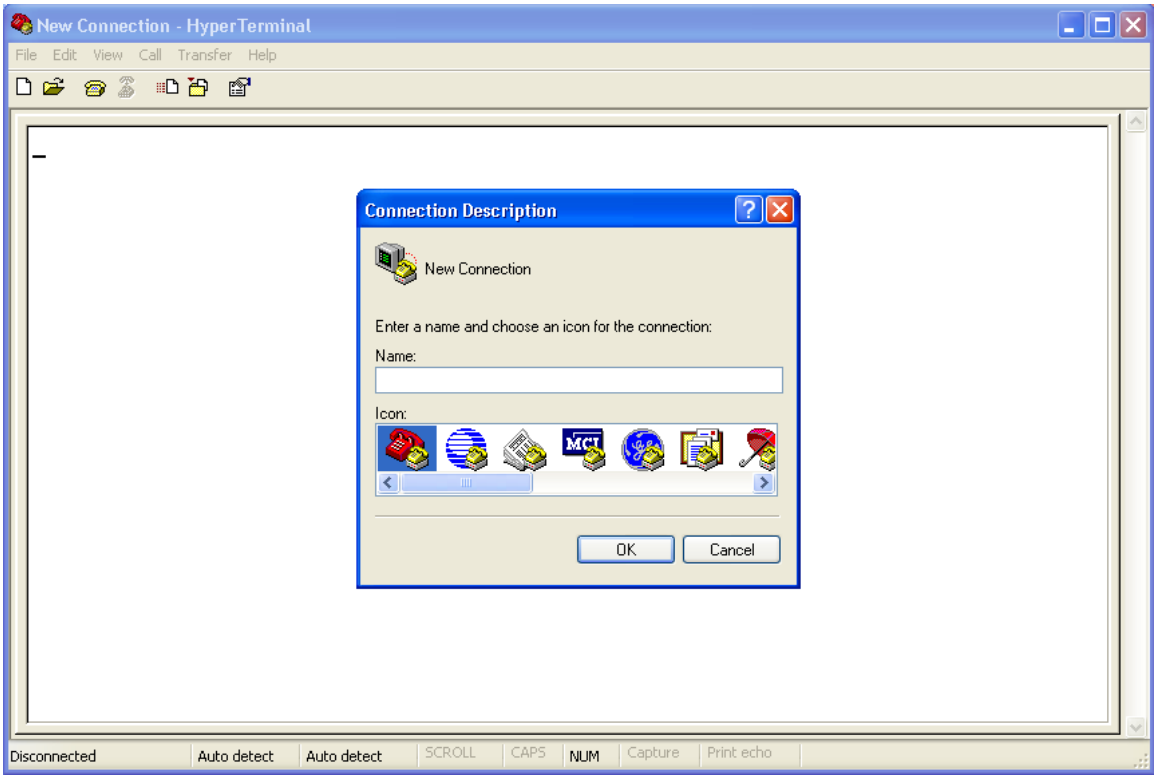
```
  }
}
```

HARWARE CONFIGURATION

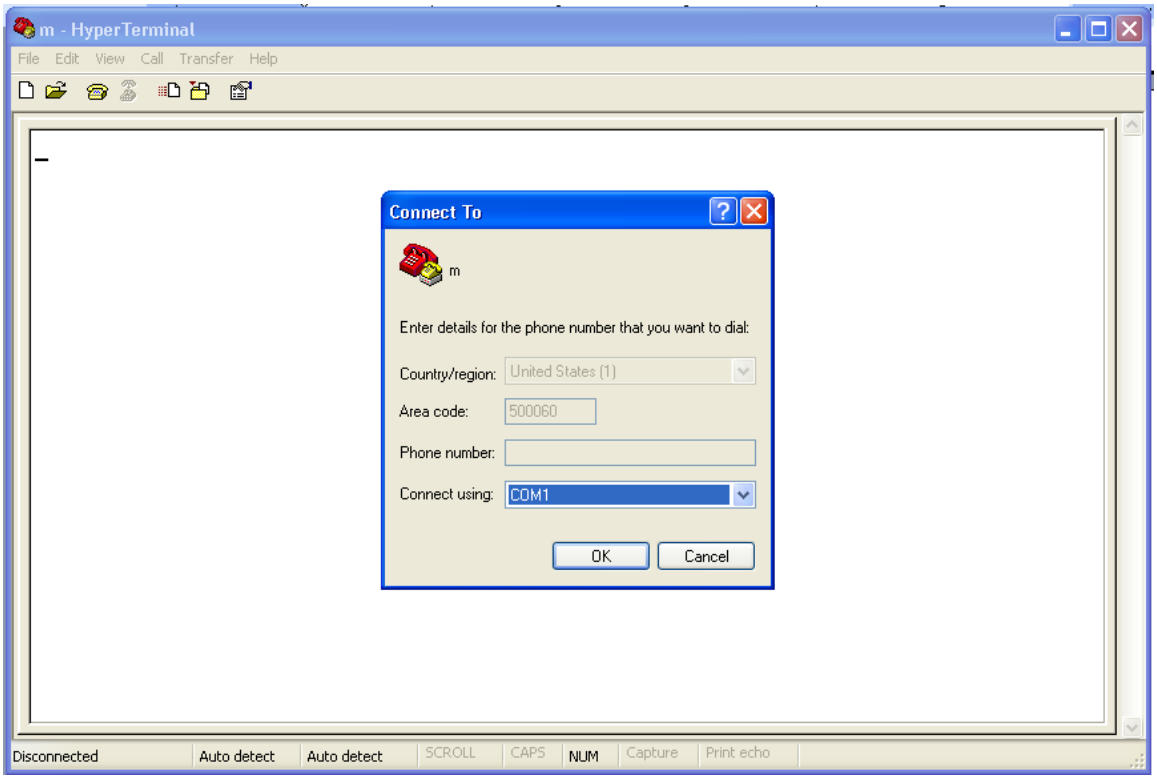
To test the board

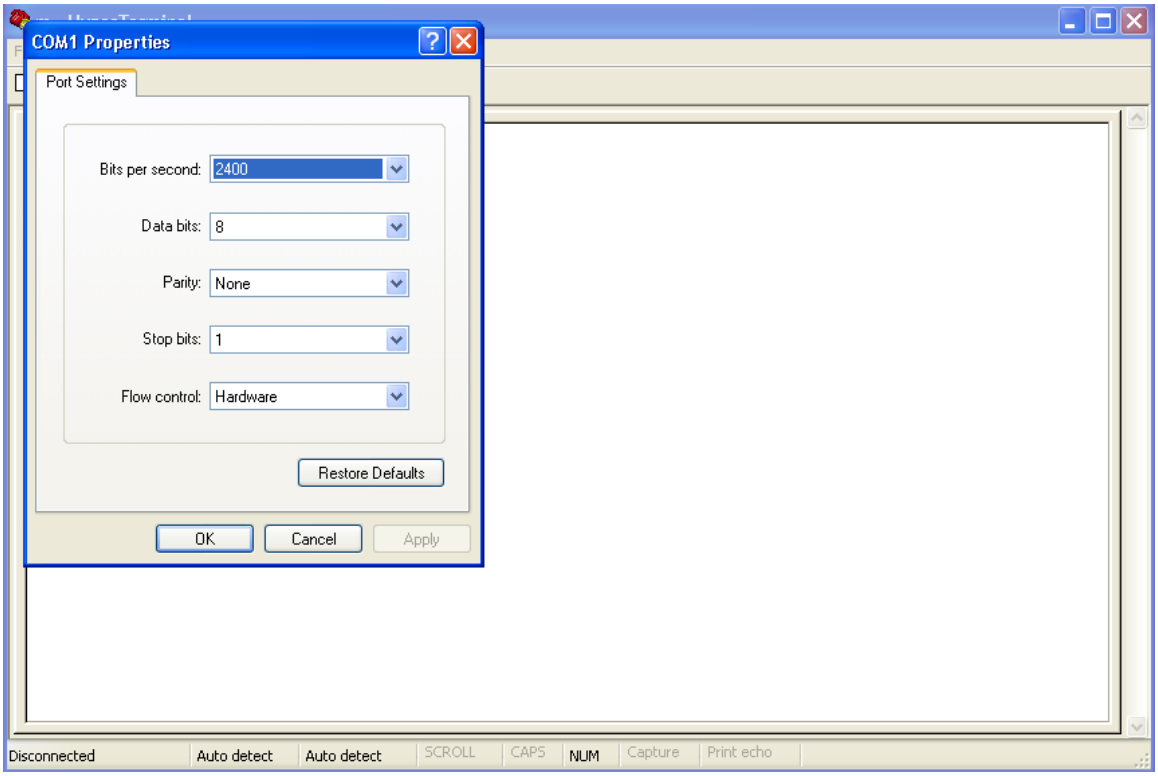
Power supply cable should be connected to the 8051 Development board and serial cable must be connected to the computer. Now go to the start menu and in that go to the all programs menu . in that go to the accessories , there in the sub menu you will find a menu called communications. In the communications menu select the hyper terminal.

After doing so the following window will be appeared. Enter any connection name in block.

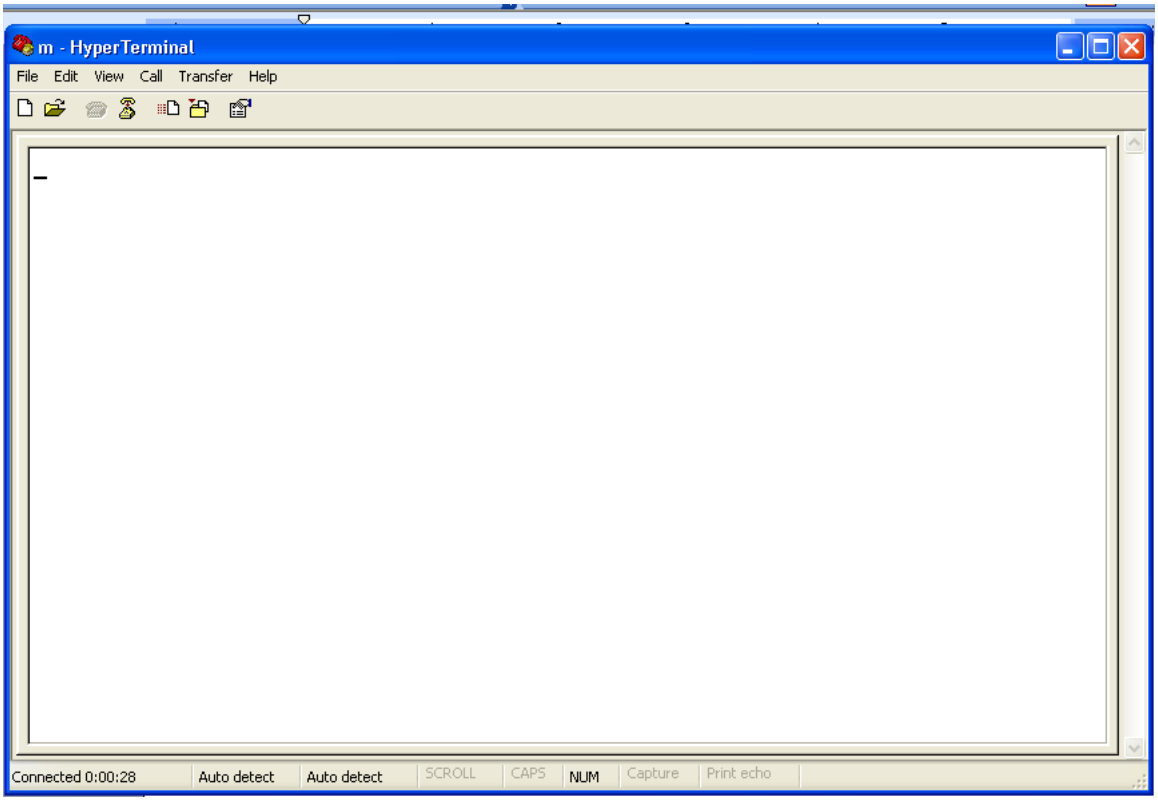


Select COM1 and then click ok





In that select the new connection and set the options as **restore to defaults.**
This will keep the default baud rate as 9600,1 start bit,1 stop bit and hardware as none.



To check the output now, if any key is pressed from the keyboard that will be displayed on the hyper terminal.

Results:-

Experiment 9:

Program to get analog input from Temperature sensor and display the temperature value on PC Monitor

Solution:

```
#include reg51.h
#define input P1;
double newtemp,pass1,pass2,T;        //variables used
  //LCD
sbit rs = P3^0;       //register select pin
sbit rw = P3^1;       //read write pin
sbit e = P3^2;       //enable pin
//ADC
sbit rd=P3^7;              //defines rd pin of ADC use for reading purposes
sbit wr=P3^6;              // define wr pin of ADC use for writing purposes
sbit intr=P3^4;            //defines intr pin use for sending interrupts from microcontroller
void delay(unsigned int time)    //Function to provide time delay in msec.
{
int i,j ;
for(i=0;i<time;i++)
for(j=0;j<1275;j++);
}
double adc()  // Function to read the values from ADC and send to controller.
{
double temp;
rd=1;
wr=0;
delay(1);
wr=1;
while(intr==1);
{rd=0;
temp=input;
delay(3);}
return temp;
}
void lcdcmd(unsigned char item)     //Function to send commands to LCD see command tables in LCD
Link
{
P2 = item;
rs= 0;
rw=0;
e=1;                                          //send to high to low pulse while writing
delay(1);
e=0;
}
void lcddata(double item)    //Function to send data to LCD
{
```

```
P2 = item;
rs= 1;
rw=0;
e=1;                                        //send high to low pulse while writing
delay(1);
e=0;
}
void disp_temp(double num)          //displays number on LCD
{
unsigned char UnitDigit  = 0;  //It will contain unit digit of number
unsigned char TenthDigit = 0;  //It will contain 10th position digit of number
unsigned char HundDigit = 0;  //It will contain 100th position digit of number
unsigned char decimal=0;        //It will contain the decimal position of number
int point;
point=num*10;
HundDigit=(num/100);
if( HundDigit != 0)              // If it is zero, then don't display
lcddata(HundDigit+0x30);       // Make Character of HundDigit and then display it on LCD
TenthDigit = num - HundDigit*100;  // Findout Tenth Digit
TenthDigit = TenthDigit/10;
if (HundDigit==0 && TenthDigit==0){}              // If it is zero, then don't display
else
lcddata(TenthDigit+0x30);       // Make Char of TenthDigit and then display it on LCD
UnitDigit = num - HundDigit*100;
UnitDigit = UnitDigit - TenthDigit*10;
lcddata(UnitDigit+0x30);        // Make Char of UnitDigit and then display it on LCD
lcddata('.');
decimal=(point%10);
lcddata (decimal+0x30);                  // Make Char of Decimal Digit and then display it on LCD
lcddata(' '); lcddata('C');
}
void read(){      // Displays "READING" while controller reads from ADC
lcdcmd(0x0E);    //turn display ON for cursor blinking
lcdcmd(0x01);    //clear screen
lcdcmd(0x06);            //increment cursor
lcddata('R');lcddata('E');lcddata('A');lcddata('D');lcddata('I');lcddata('N');lcddata('G');lcddata(' ');
}
void main()
{
P0=0x00;                                  //intialize port 0 to low use while controller reads the temperature
from
                          //ADC
read();                                   // show reading on LCD while controller reads from ADC
while(1){                          // use for checking errors while reading the value from ADC
newtemp=adc();                          //reads first value  from ADC
delay(60);                        //waits 60 msec
pass1=adc();                      // reads the Second value from ADC
delay(60);                        // waits 60 msec
```

```
  if (newtemp==pass1){              //compare first and second value
      break;                        // if first and second value is same breaks the loop
        }
}
while(1){                           //enters in the permanent loop
T=160;                          //set reference voltage acting multiplier factor for temperature accuration
newtemp=(((newtemp*T)/255));       //converts the temperature value according to reference adjusted in
decimal
lcdcmd(0x0E);                  //turn display ON for cursor blinking
lcdcmd(0x01);                  //clear screen
lcdcmd(0x06);                  //increment
disp_temp(newtemp);            //show temperature
delay(300);                       //waits 3sec before re-measure the value of temperature

while(1){                          // re-measure the value from ADC but this time double check
newtemp=adc();
delay(60);
pass1=adc();
delay(60);
pass2=adc();
  if (newtemp==pass1){
    if(pass1==pass2){
          break;                }
        }
}
                          // end ADC while loo[
          }                         // end while permanent loop
}                                   // end main loop
```

Results:-

Experiment 10:

Program to interface Stepper Motor to rotate the motor in clockwise and anticlockwise directions

Solution:

```
#include<reg51.h>
sfr stepper=0xA0;

void delay(unsigned int count)
{
int i;
for(i=0;i<count;i++);
}

void main()
{
while(1)
{
stepper=0x01;
delay(350);
stepper=0x02;
delay(350);
stepper=0x04;
delay(350);
stepper=0x08;
delay(350);
}
}
/*****************************/

/**** Half Drive Stepping ****/
#include<reg51.h>
sfr stepper=0xA0;

void delay(unsigned int count)
{
int i;
for(i=0;i<count;i++);
}

void main()
{
while(1)
{
stepper=0x01;
```

```
delay(300);
stepper=0x03;
delay(300);
stepper=0x02;
delay(300);
stepper=0x06;
delay(300);
stepper=0x04;
delay(300);
stepper=0x0C;
delay(300);
stepper=0x08;
delay(300);
stepper=0x09;
delay(300);
}
}
```

Results :-

Experiment 11:

Program to interfacing RFID

Solution:

```
//Program to interface RFID with 8051 microcontroller (AT89C51) #include<reg51.h>
unsigned int data_out,command=0x80,temp;
sfr lcd_data_pin=0xA0; //P2 port
sbit rs=P1^0;        //Register select
sbit rw=P1^1;         //Read/Write
sbit en=P1^2;        //Enable pin
unsigned char card_id[12];

void delay(unsigned int count)    //Function to provide delay
{
   int i,j;
    for(i=0;i<count;i++)
    for(j=0;j<1275;j++);
}

void lcd_command(unsigned char comm)   //Lcd command funtion
{
    lcd_data_pin=comm;
    en=1;
    rs=0;
    rw=0;
    delay(1);
    en=0;
}

void lcd_data(unsigned char disp)  //Lcd data function
{
    lcd_data_pin=disp;
    en=1;
    rs=1;
    rw=0;
    delay(1);
    en=0;
}

lcd_string(unsigned char *disp)     //Function to send string
{
    int x;
    for(x=0;disp[x]!=0;x++)
   {
       lcd_data(disp[x]);
   }
}
void lcd_ini()                //Function to initialize the LCD
{
   lcd_command(0x38);
   delay(5);
   lcd_command(0x0F);
   delay(5);
   lcd_command(0x80);
```

```
   delay(5);
}

void recieve()     //Function to recieve data serialy from RS232
{
   unsigned char k;
    for(k=0;k<12;k++)
    {
       while(RI==0);
        card_id[k]=SBUF;
        RI=0;
   }
}

void main()
{
   int l;
   TMOD=0x20;           //Enable Timer 1
   TH1=0XFD;
   SCON=0x50;
   TR1=1;               // Triggering Timer 1
   lcd_ini();
   lcd_command(0x81);       //Place cursor to second position of first line
   lcd_string("UNIQUE CARD ID:");
   delay(200);
   while(1)
   {
       recieve();
       lcd_command(0xC1);       //Place cursor to second position of second line
       for(l=0;l<12;l++)
       {
           lcd_data(card_id[l]);
        }
    }
}
```

Results:-

Experiment 12:

Implementation of Traffic light controller

Solution:

```
#include<reg51.h>
sbit G1 = P2^4;                 //assigning Port2 Bit 4 as G1
sbit Y1 = P2^5;                 //assigning Port2 Bit 5 as Y1
sbit R1 = P2^6;                 //assigning Port2 Bit 6 as R1
sbit G2 = P1^1;                 //assigning Port1 Bit 1 as G2
sbit Y2 = P1^2;                 //assigning Port1 Bit 2 as Y2
sbit R2 = P1^3;                 //assigning Port1 Bit 3 as R2
sbit G3 = P2^1;                 //assigning Port2 Bit 1 as G3
sbit Y3 = P2^2;                 //assigning Port2 Bit 2 as Y3
sbit R3 = P2^3;                 //assigning Port2 Bit 3 as R3
sbit G4 = P3^1;                 //assigning Port3 Bit 1 as G4
sbit Y4 = P3^2;                 //assigning Port3 Bit 2 as Y4
sbit R4 = P3^3;                 //assigning Port3 Bit 3 as R4
void main(void)
{
unsigned int  x;
while(1)
{
G1 = 0;         //Making G1 0
Y1 = 1;         //Making Y1 0
R1 = 1;         //Making R1 0
G2 = 1;         //Making G2 0
Y2 = 1;         //Making Y2 0
R2 = 0;         //Making R2 0
G3 = 1;         //Making G3 0
Y3 = 1;         //Making Y3 0
R3 = 0;         //Making R3 0
G4 = 1;         //Making G4 0
Y4 = 1;         //Making Y4 0
R4 = 0;         //Making R4 0
for(x=0;x<40000;x++);                   //Making time delay of 40000
```

```
G1 = 1;
Y1 = 0;
R1 = 1;
G2 = 1;
Y2 = 0;
R2 = 1;
G3 = 1;
Y3 = 1;
R3 = 0;
G4 = 1;
Y4 = 1;
R4 = 0;
for(x=0;x<40000;x++);                    //Making time delay of 40000

G1 = 1;
Y1 = 1;
R1 = 0;
G2 = 0;
Y2 = 1;
R2 = 1;
G3 = 1;
Y3 = 1;
R3 = 0;
G4 = 1;
Y4 = 1;
R4 = 0;
for(x=0;x<40000;x++);                    //Making time delay of 40000
G1 = 1;
Y1 = 1;
R1 = 0;
G2 = 1;
Y2 = 0;
R2 = 1;
G3 = 1;
Y3 = 0;
R3 = 1;
```

```
G4 = 1;
Y4 = 1;
R4 = 0;
for(x=0;x<40000;x++);                    //Making time delay of 40000
G1 = 1;
Y1 = 1;
R1 = 0;
G2 = 1;
Y2 = 1;
R2 = 0;
G3 = 0;
Y3 = 1;
R3 = 1;
G4 = 1;
Y4 = 1;
R4 = 0;
for(x=0;x<40000;x++);                    //Making time delay of 40000
G1 = 1;
Y1 = 1;
R1 = 0;
G2 = 1;
Y2 = 1;
R2 = 0;
G3 = 1;
Y3 = 0;
R3 = 1;
G4 = 1;
Y4 = 0;
R4 = 1;
for(x=0;x<40000;x++);                    //Making time delay of 40000
G1 = 1;
Y1 = 1;
R1 = 0;
G2 = 1;
Y2 = 1;
R2 = 0;
```

```
G3 = 1;
Y3 = 1;
R3 = 0;
G4 = 0;
Y4 = 1;
R4 = 1;
for(x=0;x<40000;x++);                    //Making time delay of 40000
G1 = 1;
Y1 = 0;
R1 = 1;
G2 = 1;
Y2 = 1;
R2 = 0;
G3 = 1;
Y3 = 1;
R3 = 0;
G4 = 1;
Y4 = 0;
R4 = 1;
for(x=0;x<40000;x++);                    //Making time delay of 40000
}
}
```

Results:-