DEEPAK GUPTA

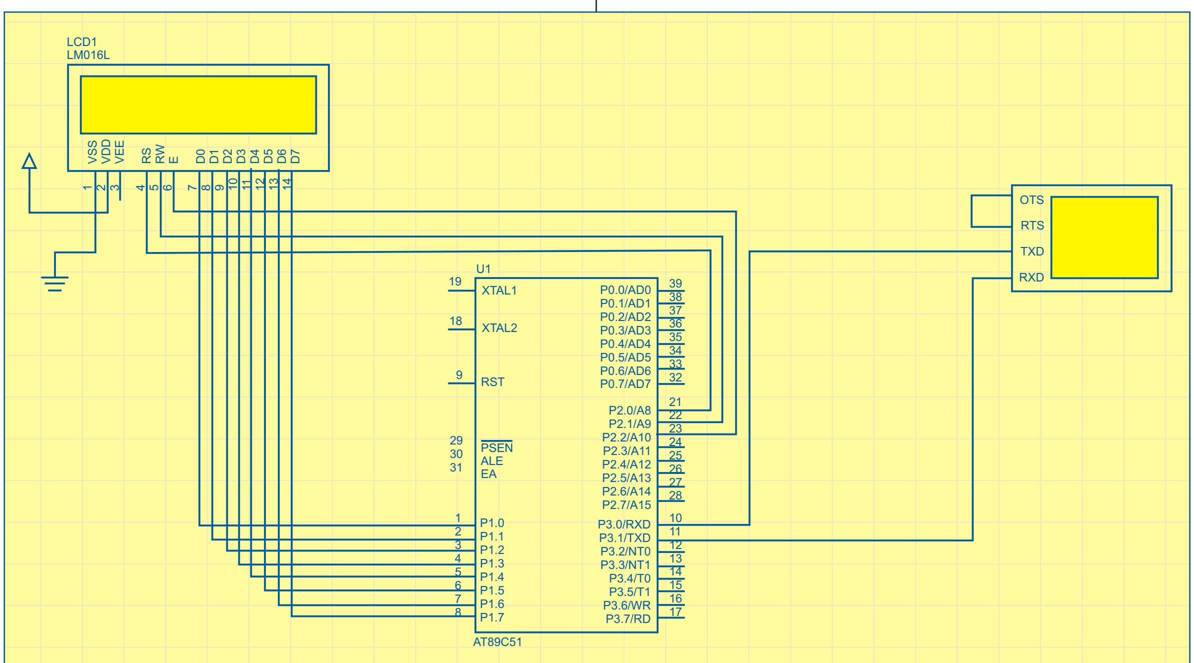
EM TESTED

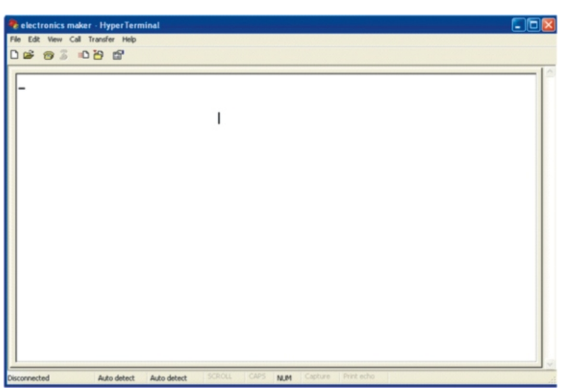
Interfacing Hyper terminal with 8051

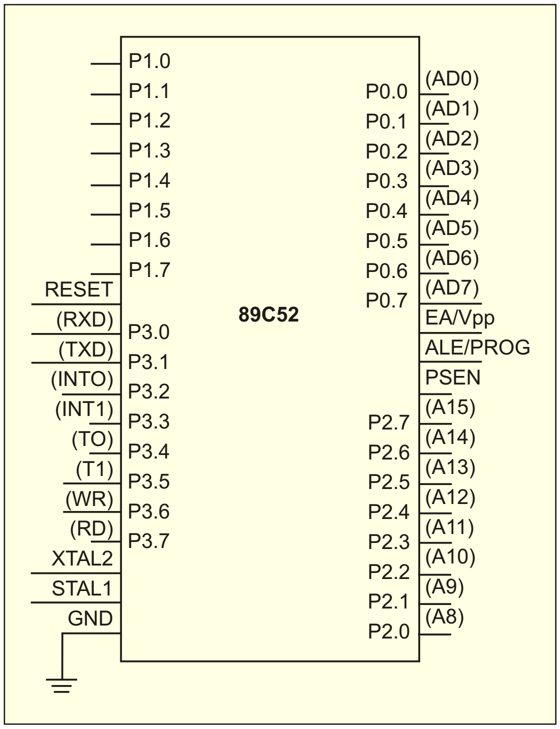
VASEEM DURRANI

HyperTerminal is a Windows application. HyperTerminal is an application you can use in order to connect your computer to other remote systems. Within HyperTerminal's user interface, you will find menus, buttons, icons, and messages. All these elements and controls work together so as to provide convenience for the user, especially for accessing the necessary features and performing various tasks. This application is a useful tool, particularly for testing if your modem is working well and in verifying if you have a stable connection with other sites. In order to check if your modem's settings are configured correctly or if your modem is connected properly, you can send a set of commands through HyperTerminal and view the results given. Other functions of HyperTerminal would include the recording of data being sent to and from the service of the computer you are connected to. Through this information, you will be able to determine the stability of your connections.

In addition, communication with the user of the computer on the other side of the remote connection is made possible through the scroll feature of HyperTerminal. This is a feature that allows you to receive and view texts on the screen as well as the capability to send your own text messages. Another notable feature of







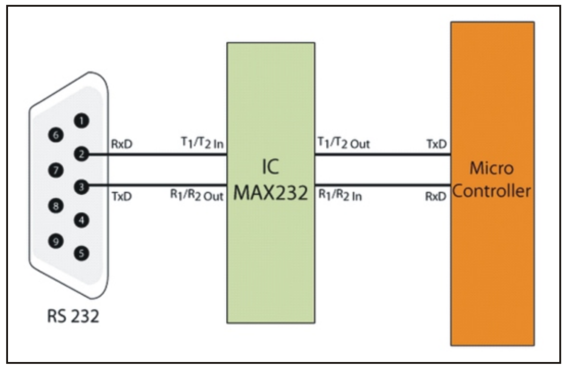
HyperTerminal is its option that enables you to transfer files from your desktop PC onto a portable device such as a laptop computer via a serial port. The circuit demonstrates the interfacing of microcontroller AT89C51 with HyperTerminal. The circuit is built on proteus and the crystal frequency is set internally.

The microcontroller AT89c51 has an inbuilt UART for carrying out serial communication. The serial communication is done in the asynchronous mode. A serial port, like other PC ports, is a physical interface to establish data transfer between computer and an external hardware or device. This transfer, through serial port, takes place bit by bit. IBM introduced the DB-9 RS-232 version of serial I/O standard, which is most widely used in PCs and several devices. In RS232, high and low bits are represented by flowing voltage ranges:

Bit	Voltage Range (in V)	
0	+3	+25
1	-25	-3

The range -3V to +3V is undefined. The TTL standards came a long time after the RS232 standard was set. Due to this reason RS232 voltage levels are not compatible with TTL logic. Therefore, while connecting an RS232 to microcontroller system, a voltage converter is required. This converter converts the microcontroller output level to the RS232 voltage levels, and vice versa. IC MAX232, also known as line driver, is very commonly used for this purpose.

The simplest connection between a PC and microcontroller requires a minimum of three pins, RxD (receiver, pin2), TxD (transmitter, pin3) and ground (pin5) of the serial port of computer. TxD pin of serial port connects to RxD pin of controller via MAX232. And similarly, RxD pin of serial port connects to the TxD pin of controller through MAX232.MAX232 has two sets of line drivers for transferring and receiving data. The line drivers used for transmission are called T1 and T2, where as the line drivers for receiver are designated as R1 and R2.



An important parameter considered while interfacing serial port is the Baud rate which is the speed at which data is transmitted serially. It is defined as number of bits transmitted or received per second. It is generally expressed in bps (bits per second). AT89C51 microcontroller can be set to transfer and receive serial data at different baud rates using software instructions. Timer1 is used to set the baud rate of serial communication for the microcontroller. For this purpose, Timer1 is used in mode2 which is an 8-bit auto reload mode. To get baud rates compatible with the PC, TH1 should be loaded with the values as shown:

In this project baud rate 9600bps is used.

Baud Rate (bps)	TH1 (Hex value)
9600	FD
4800	FA
2400	F4
1200	E8

For serial communication AT89C51 has registers SBUF and SCON (Serial control register). SBUF is an 8-bit register. For transmitting a data byte serially, it needs to be placed in the SBUF register. Similarly whenever a data byte is received serially, it comes in the SBUF register, i.e., SBUF register should be read to receive the serial byte. SCON register is used to set the mode of serial communication. The project uses Mode1, in which the data length is of 8 bits and there is a start and a stop bit. The SCON register is bit addressable register.The different modes are :

SM0	SM1	
0	0	Serial mode 0
0	1	Serial mode 1, 8-bit data, 1 start bit, 1 stop bit
1	0	Serial mode 2
1	1	Serial mode 3

TI (transmit interrupt) is an important flag bit in the SCON register. The controller raises the TI flag when the 8-bit character is transferred. This indicates that the next byte can be transferred now. The TI bit is raised at the beginning of the stop bit.

RI (receive interrupt) is also a flag bit of the SCON register. On receiving the serial data, the microcontroller skips the start and stop bits, and puts the byte is SBUF register. The RI flag bit is then raised to indicate that the byte has been received and should be picked up.

Hyper Terminal Hyper Terminal, a Windows XP application, can be used to receive or transmit serial data through RS232. To open Hyper Terminal, go to Start Menu, select all programs, go to Accessories, click on Communications and select Hyper Terminal.

To start a new connection, go to File menu and click on new connection. The connection window opens up. Give a name to your connection and select 1st icon and click on OK. Connection property window opens here. Select Bit rate as 9600bps, Data bits 8, Parity as none, Stop bit 1, Flow control none and click OK. Now the serial data can be read on hyper terminal.

In program, Timer1 is used with auto reload setting. The baud rate is fixed to 9600bps by loading TH1 to 0xFD. The value 0x50 is loaded in the SCON register. This will initialize the serial port in Mode1. whenever the enter is pressed on the keyboard the typed string is serially sent to controller using UART port and is displayed on the LCD. Since the LCD used is 16x2 ,the characters after 1 bytes are displayed on the next line of theLCD.

A string of characters is sent from the hyper terminal and is displayed on the 16x2 LCD. The controller is serially interfaced with the hyper terminal using Max232 (this is not shown in the circuit).

Component used

1: 89c51 Microcontroller

89c51 is a low power ,high performance CMOS 8 bit controller with 8K of in system programmable Flash memory. By combining a versatile 8 bit CPU within system programmable flash on a monolithic chip, the atmel AT89c51 is a powerful computer which provides a high flexibility and cost effective solutions to many embedded control applications.

The main features of 8051 microcontroller are:

I. RAM – 128 Bytes (Data memory)

Ii. ROM – 4Kbytes (ROM signify the on – chip program space)

Iii. Serial Port – Using UART makes it simpler to interface for serial communication.

Iv. Two 16 bit Timer/ Counter

V. Input/output Pins – 4 Ports of 8 bits each on a single chip.

Vi. 6 Interrupt Sources

Vii. 8 – bit ALU (Arithmetic Logic Unit)

Viii. Harvard Memory Architecture – It has 16 bit Address bus (each of RAM and ROM) and 8 bit Data Bus.

Ix. 8051 can execute 1 million one-cycle instructions per second with a clock frequency of 12MHz.

2: MAX232

The MAX232 is an integrated circuit that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.

The drivers provide RS-232 voltage level outputs (approx. ± 7.5 V) from a single + 5 V supply via on-chip charge pumps and external capacitors. This makes it useful for implementing RS-232 in devices that otherwise do not need any voltages outside the 0 V to + 5 V range, as power supply design does not need to be made more complicated just for driving the RS-232.

3: 16x2 LCD

The 16x2 LCD is very popular because of it's built in HD44780 interface module. This module makes it extremely easy to add an LCD to any project with its built in character set and easy command structure. It requires 3 control lines as well as either 4 or 8 I/O lines for the data bus. The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for the data bus). If an 8-bit data bus is used the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus).

The three control lines are referred to as EN, RS, and RW.

The EN line is called "Enable." This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

The RS line is the "Register Select" line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which sould be displayed on the screen. For example, to display the letter "T" on the screen you would set RS high.

The RW line is the "Read/Write" control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands--so RW will almost always be low.

Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.

In subsequent projects , the HyperTerminal will be replaced by the microcontroller itself; thus avoiding the need of using a Computer to establish an interface. This would lead to an independent GSM based system.

Program Code

```
#include <REGX51.H>
sbit rs= P2^0;
sbit rw= P2^1;
sbit en= P2^2;
///////////////////////////////////Delay///////////////////////////////////
void mdelay(unsigned int itime)
{
        unsigned int i,j;
        for(i=0;i<itime;i++)
        for(j=0;j<1275;j++);
}
/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / L C D
COMMAND/////////////////////////////////////
void lcdcmd(unsigned char value)
{
        P1=value;
        rs=0;
        rw=0;
        en=1;
        mdelay(1);
        en=0;
}
/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / L C D
VALUE//////////////////////////////////////
void lcd_data(unsigned char value)
{
        P1=value;
        rs=1;
        rw=0;
        en=1;
        mdelay(1);
        en=0;
}
////////////////////////////////STRING LCD///////////////////////////////
void string_lcd(unsigned char * ptr)
{
        unsigned int h,j=0;
        while(*ptr)
        {
                h++;
                if(h<15)
                {
                        lcd_data(*ptr++);
                }
                if(h>=15)
                {
                        lcdcmd(0xC0+j);
                        lcd_data(*ptr++);
                        j++;
                }
//              *(ptr++);
        }
}
/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / U A R T
INITIALIZATION/////////////////////////////////
void uart_init()
{
        TMOD=0x20;
        TH1=0xFD;
        SCON=0x50;
        TR1=1;
}
/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / L C D
INITIALIZATION////////////////////////////////
void lcd_init()
{
        lcdcmd(0x38);
        //mdelay(250);
        lcdcmd(0x0E);
        //mdelay(250);
        lcdcmd(0x01);
        //mdelay(250);
        lcdcmd(0x06);
        //mdelay(250);
        lcdcmd(0x80);
}       //mdelay(250);
//////////////////////////////////MAIN//////////////////////////////////////
/
void main()
{

        unsigned char str2[40],*ptr;
        unsigned int i;
        ptr=str2;
        uart_init();
        lcd_init();

        while(SBUF!=0x0D)       //Checking if the enter is pressed or
not
        {
                while(RI==0);
                RI=0;
                str2[i]=SBUF;
                i++;
        }
                str2[i]='\0';
        string_lcd(ptr);
        // To display on LCD
/*      for(i=0;i<6;i++)
        {
                lcd_data(str2[i]);
        }*/
        while(1);
}
```