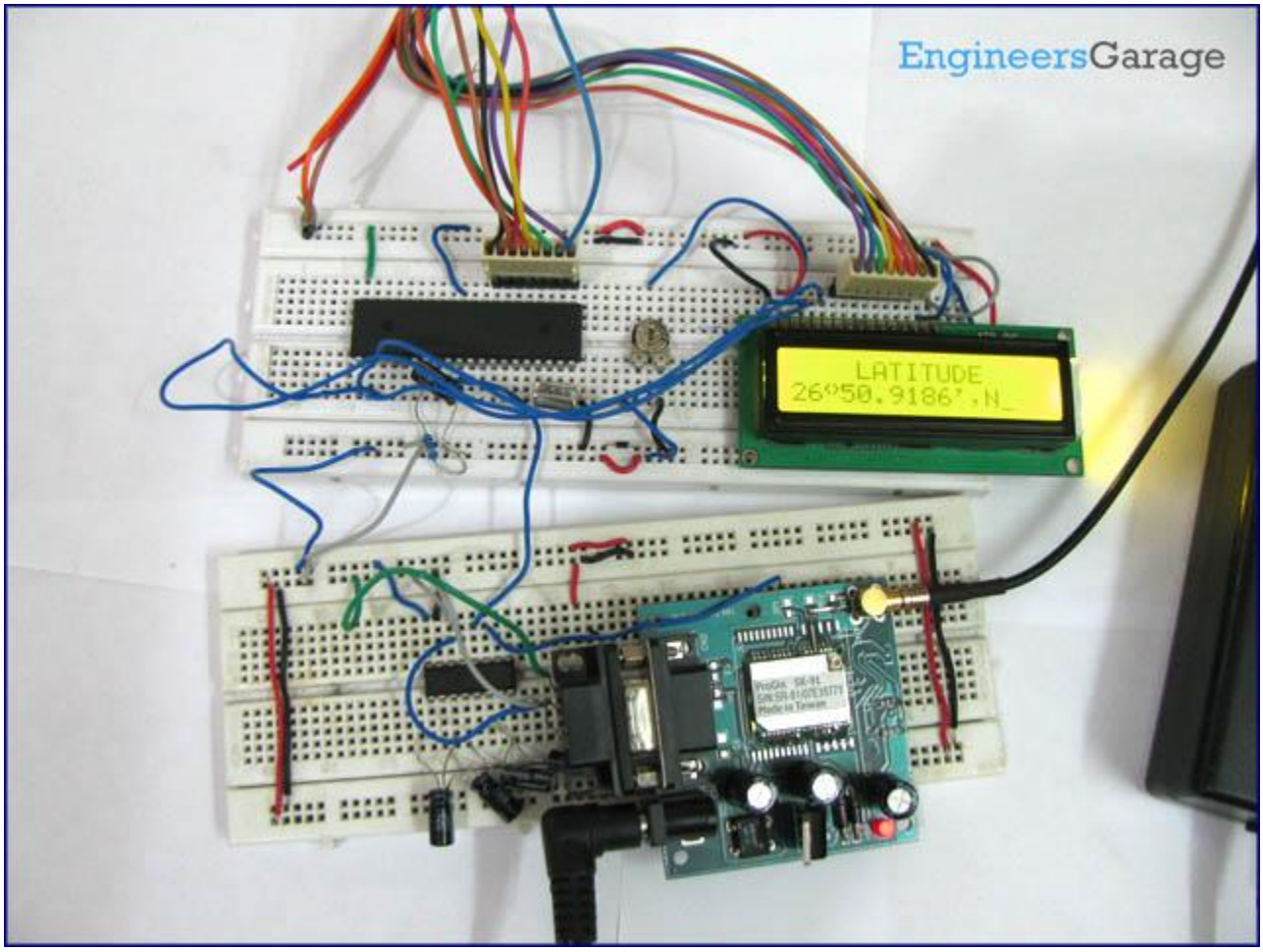
How to interface GPS with 8051 Microcontroller (AT89C51)

The GPS Module has been interfaced with AT89C51 and the location values are displayed on a 16x2 LCD interface.



The GPS module continuously transmits serial data (RS232 protocol) in the form of sentences according to NMEA standards. The latitude and longitude values of the location are contained in the **GPGGA** sentence (refer NMEA format). In this program, these values are extracted from the GPGGA sentence and are displayed on LCD.

The serial data is taken from the GPS module through MAX232 into the SBUF register of 8051 controller. The serial data from the GPS receiver is taken by using the Serial Interrupt of the controller. This data consists of a sequence of NMEA sentences from which GPGGA sentence is identified and processed.

The extraction of location values is done as follows. The first six bytes of the data received are compared with the pre-stored ($GPGGA) string and if matched then only data is further accounted for; otherwise the process is repeated again. From the comma delimited GPGGA sentence, latitude and longitude positions are extracted by finding the respective comma positions and extracting the data. The latitude and longitude positions extracted are displayed on theLCD interfaced with AT89C51.

To obtain more details (other than latitude and longitude) from the GPS receiver, GPRMS sentence can be used.

The circuit connections are as follows:

Receiver1 (R_1) of MAX232 has been used for the serial communication. The receiver pin of GPS module is connected to R_1IN (pin13) of MAX232. R_1OUT (pin 12) of MAX232 is connected to RxD (P3.0) of AT89C51.

Pins 1-3 of port P1 (P1.0, P1.1 & P1.2 respectively) of AT89C51 are connected to the control pins (RS, R/W& EN) of LCD. The data pins of LCD are connected to Port P2 of the controller. The latitude and longitude positions are displayed on the LCD.

Code :

```
/* Basic program to show latitude and longitude on LCD extracted from GPGGA statement
*/

#include<reg51.h>
#define port2 P2
sbit rs = P1^0;
sbit rw = P1^1;
sbit e = P1^2;
char info[70];
char test[6]={"$GPGGA"};
char comma_position[15];
unsigned int check=0,i;
unsigned char a;
void receive_data();
void lcd_latitude();
void lcd_longitude();

//DELAY FUNCTION
void delay(unsigned int msec)
{
        int i,j ;
        for(i=0;i<msec;i++)
        for(j=0;j<1275;j++);
```

```
}

// LCD COMMAND SENDING FUNCTION
void lcd_cmd(unsigned char item)
{
        port2 = item;
        rs= 0;
        rw=0;
        e=1;
        delay(1);
        e=0;
        return;
}

// LCD DATA SENDING FUNCTION
void lcd_data(unsigned char item)
{
        port2 = item;
        rs= 1;
        rw=0;
        e=1;
        delay(1);
        e=0;
        return;
}

 // LCD STRING SENDING FUNCTION
void lcd_string(unsigned char *str)
{
        int i=0;
        while(str[i]!='\0')
        {
        lcd_data(str[i]);
        i++;
        delay(10);
        }
        return;
}

// SERIAL PORT SETTING
void serial()
{
        TMOD=0x20;      //MODE=2
        TH1=0xfa;             // 4800 BAUD
        SCON=0x50  ;    // SERIAL MODE 1 ,8- BIT DATA ,1 STOP BIT ,1 START BIT ,
RECEIVING ON
        TR1=1;           //TIMER START
}

void find_comma()
{
        unsigned int i,count=0;
        for(i=0;i<70;i++)
        {
                if(info[i]==',')
```

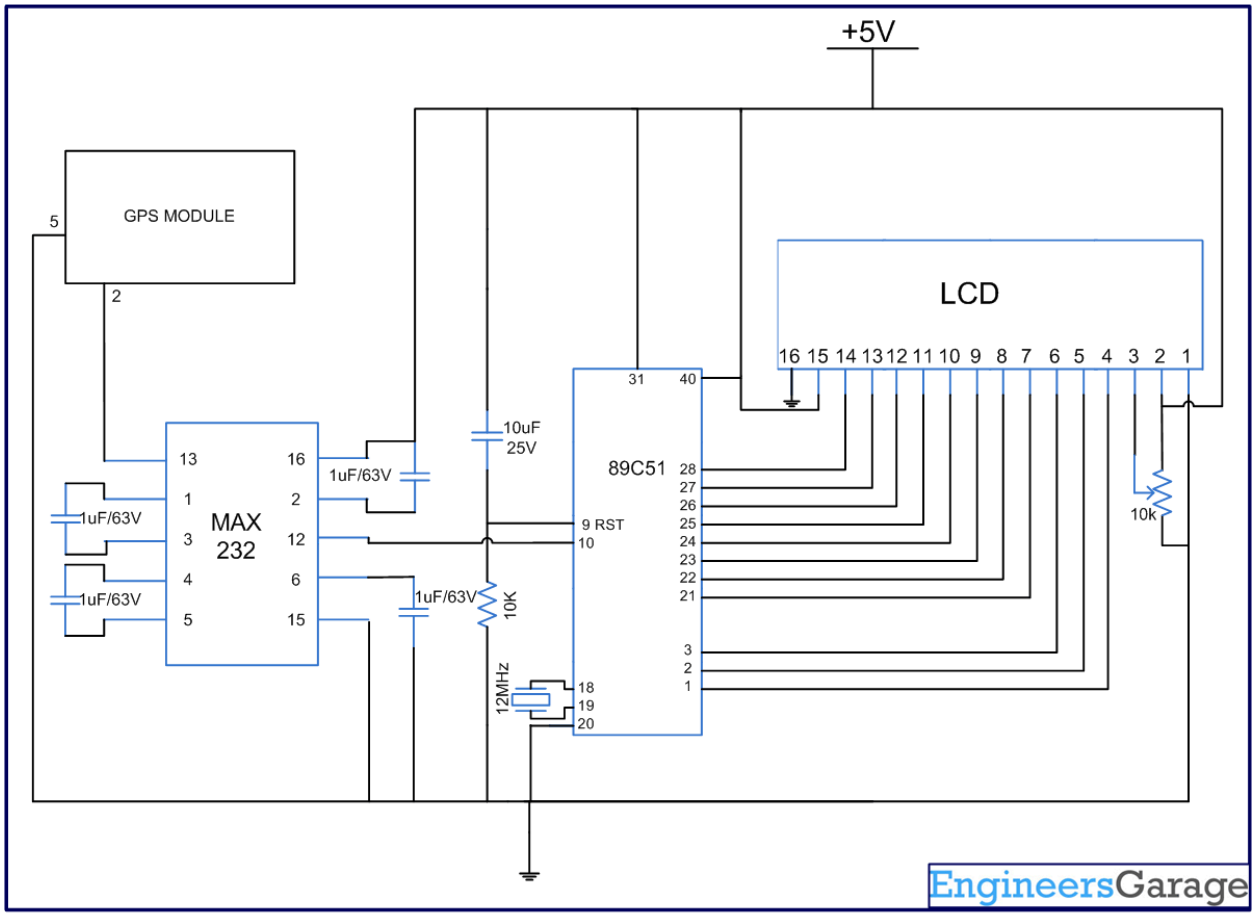
```
                {
                        comma_position[count++]=i;
                }
         }
}
void compare()
{
        IE=0x00;        //Interrupt disable
        find_comma();       //Function to detect position of comma in the string
        lcd_latitude();    //Function to show Latitude
        lcd_longitude();   //Function to show Longitude
        check=0;
        IE=0x90;            //Interrupt enable
}
void receive_data()               interrupt 4

{
        info[check++]=SBUF;         //Read SBUF
        if(check<7)               //Condition to check the required data
        {
                if(info[check-1]!=test[check-1])
                check=0;
        }
        RI=0;
}
void lcd_shape()       //Function to create shape of degree
{
        lcd_cmd(64);
        lcd_data(10);
        lcd_data(17);
        lcd_data(17);
        lcd_data(10);
        lcd_data(0);
        lcd_data(0);
        lcd_data(0);
        lcd_data(0);
}

void lcd_latitude()               //Function to display Latitude
{
        unsigned int c2=comma_position[1]; //Position of second comma
        lcd_shape();
        lcd_cmd(0x01);            // Clear LCD display
        lcd_cmd(0x84);            //Move cursor to position 6 of line 1
        lcd_string("LATITUDE");   //Showing Latitude
        lcd_cmd(0xC0);                    //Beginning of second line
        lcd_data(info[c2+1]);
        lcd_data(info[c2+2]);
        lcd_data(0);                      //Degree symbol
        lcd_data(info[c2+3]);
        lcd_data(info[c2+4]);
        lcd_data(info[c2+5]);
        lcd_data(info[c2+6]);
        lcd_data(info[c2+7]);
        lcd_data(info[c2+8]);
```

```
        lcd_data(info[c2+9]);
        lcd_data(0x27);          //ASCII of minute sign(')
        lcd_data(info[c2+10]);
        lcd_data(info[c2+11]);
        delay(250);
}

void lcd_longitude()
{
        unsigned int c4=comma_position[3];
        lcd_cmd(0x01);               //Clear LCD display
        lcd_cmd(0x84);               //Move cursor to position 4 of line 1
        lcd_string("LONGITUDE");         //Showing Longitude
        lcd_cmd(0xC0);                       //Begining of second line
        lcd_data(info[c4+1]);
        lcd_data(info[c4+2]);
        lcd_data(info[c4+3]);
        lcd_data(0);
        lcd_data(info[c4+4]);
        lcd_data(info[c4+5]);
        lcd_data(info[c4+6]);
        lcd_data(info[c4+7]);
        lcd_data(info[c4+8]);
        lcd_data(info[c4+9]);
        lcd_data(info[c4+10]);
        lcd_data(0x27);               //ASCII of minute sign(')
        lcd_data(info[c4+11]);
        lcd_data(info[c4+12]);
        delay(250);
}
void main()
{
        serial();
        lcd_cmd(0x38);           //2 LINE, 5X7 MATRIX
        lcd_cmd(0x0e);         //DISPLAY ON, CURSOR BLINKING
        IE=0x90;
        while(1)
        {
                if(check==69)
                compare();
        }
}
```


+5V
GPS MODULE
5
2
LCD
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
31
40
89C51
28
27
26
25
24
23
22
21
3
2
1
9 RST
10
18
19
20
12MHz
10uF
25V
1uF/63V
1uF/63V
1uF/63V
1uF/63V
MAX
232
13
16
1
2
3
12
4
6
5
15
10K
10k
EngineersGarage

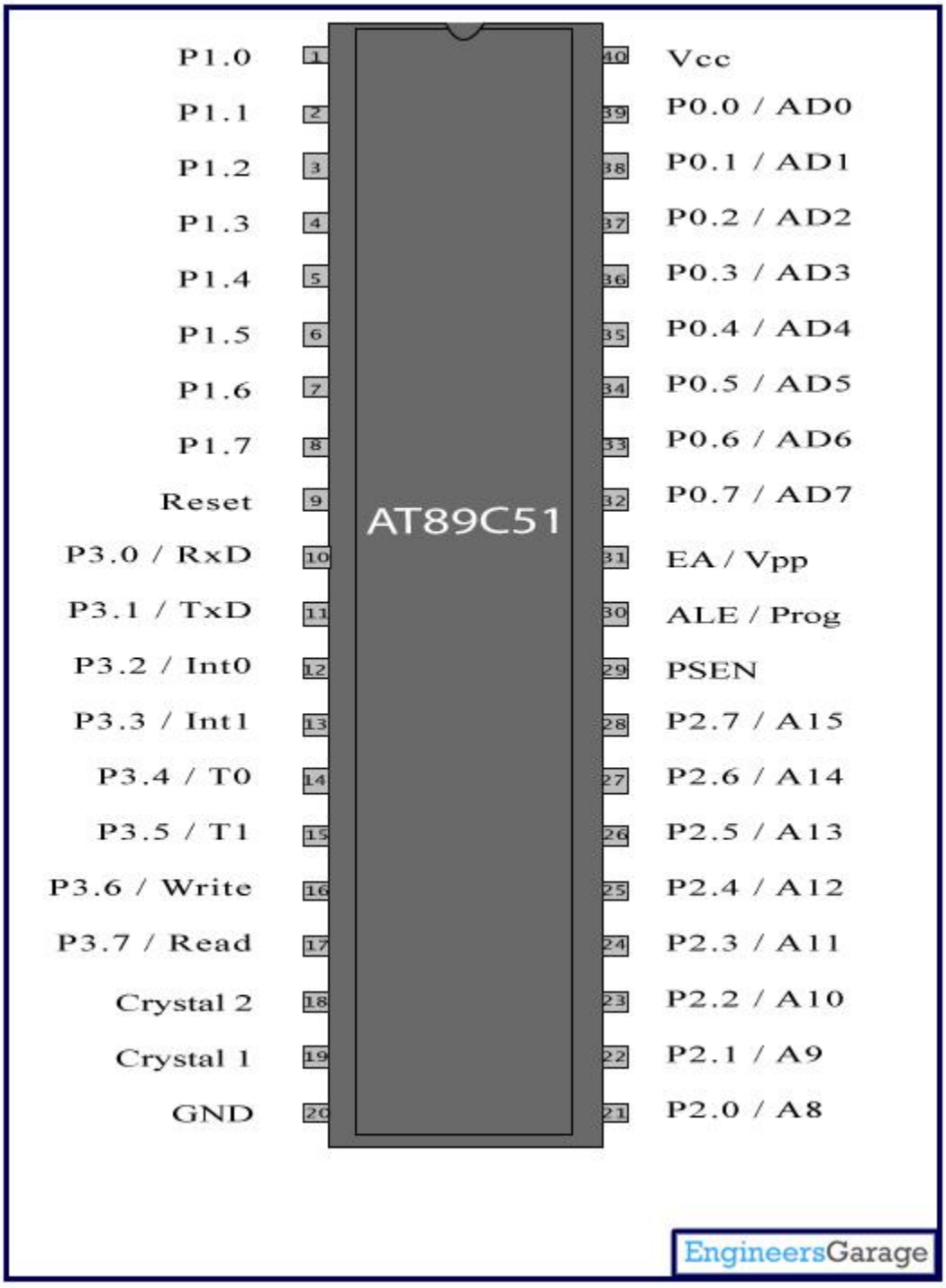
AT89C51 Microcontroller

AT89C51 is an 8-bit microcontroller and belongs to Atmel's 8051 family. **ATMEL 89C51** has 4KB of Flash programmable and erasable read only memory (PEROM) and 128 bytes of RAM. It can be erased and program to a maximum of 1000 times.

In 40 pin AT89C51, there are four ports designated as P_1, P_2, P_3 and P_0.
All these ports are 8-bit bi-directional ports, *i.e.*, they can be used as both input and output ports.
Except P_0 which needs external pull-ups, rest of the ports have internal pull-ups.
When 1s are written to these port pins, they are pulled high by the internal pull-ups and can be used as inputs.
These ports are also bit addressable and so their bits can also be accessed individually.

Port P_0 and P_2 are also used to provide low byte and high byte addresses, respectively, when connected to an external memory. Port 3 has multiplexed pins for special functions like serial communication, hardware interrupts, timer inputs and read/write operation from external memory. AT89C51 has an inbuilt UART for serial communication. It can be programmed to operate at different baud rates. Including two timers & hardware interrupts, it has a total of six interrupts.

Pin Diagram:


P1.0 1
P1.1 2
P1.2 3
P1.3 4
P1.4 5
P1.5 6
P1.6 7
P1.7 8
Reset 9
P3.0 / RxD 10
P3.1 / TxD 11
P3.2 / Int0 12
P3.3 / Int1 13
P3.4 / T0 14
P3.5 / T1 15
P3.6 / Write 16
P3.7 / Read 17
Crystal 2 18
Crystal 1 19
GND 20
AT89C51
40 Vcc
39 P0.0 / AD0
38 P0.1 / AD1
37 P0.2 / AD2
36 P0.3 / AD3
35 P0.4 / AD4
34 P0.5 / AD5
33 P0.6 / AD6
32 P0.7 / AD7
31 EA / Vpp
30 ALE / Prog
29 PSEN
28 P2.7 / A15
27 P2.6 / A14
26 P2.5 / A13
25 P2.4 / A12
24 P2.3 / A11
23 P2.2 / A10
22 P2.1 / A9
21 P2.0 / A8
EngineersGarage

Pin Description:

Pin No	Function			Name
1	8 bit input/output port (P_1) pins			$P_1.0$
2				$P_1.1$
3				$P_1.2$
4				$P_1.3$
5				$P_1.4$
6				$P_1.5$
7				$P_1.6$
8				$P_1.7$
9	Reset pin; Active high			Reset
10	Input (receiver) for serial communication	RxD	8 bit input/output port (P_3) pins	$P_3.0$
11	Output (transmitter) for serial communication	TxD		$P_3.1$
12	External interrupt 1	Int0		$P_3.2$
13	External interrupt 2	Int1		$P_3.3$
14	Timer1 external input	T_0		$P_3.4$
15	Timer2 external input	T_1		$P_3.5$
16	Write to external data memory	Write		$P_3.6$
17	Read from external data memory	Read		$P_3.7$
18	Quartz crystal oscillator (up to 24 MHz)			Crystal 2
19				Crystal 1
20	Ground (0V)			Ground
21	8 bit input/output port (P_2) pins / High-order address bits when interfacing with external memory			$P_2.0/ A_8$
22				$P_2.1/ A_9$
23				$P_2.2/ A_{10}$
24				$P_2.3/ A_{11}$
25				$P_2.4/ A_{12}$
26				$P_2.5/ A_{13}$
27				$P_2.6/ A_{14}$
28				$P_2.7/ A_{15}$
29	Program store enable; Read from external program memory			PSEN
30	Address Latch Enable			ALE
	Program pulse input during Flash programming			Prog
31	External Access Enable; Vcc for internal program executions			EA
	Programming enable voltage; 12V (during Flash programming)			Vpp
32	8 bit input/output port (P_0) pins Low-order address bits when interfacing with external memory			$P_0.7/ AD_7$
33				$P_0.6/ AD_6$
34				$P_0.5/ AD_5$
35				$P_0.4/ AD_4$
36				$P_0.3/ AD_3$
37				$P_0.2/ AD_2$
38				$P_0.1/ AD_1$
39				$P_0.0/ AD_0$
40	Supply voltage; 5V (up to 6.6V)			Vcc

MAX232

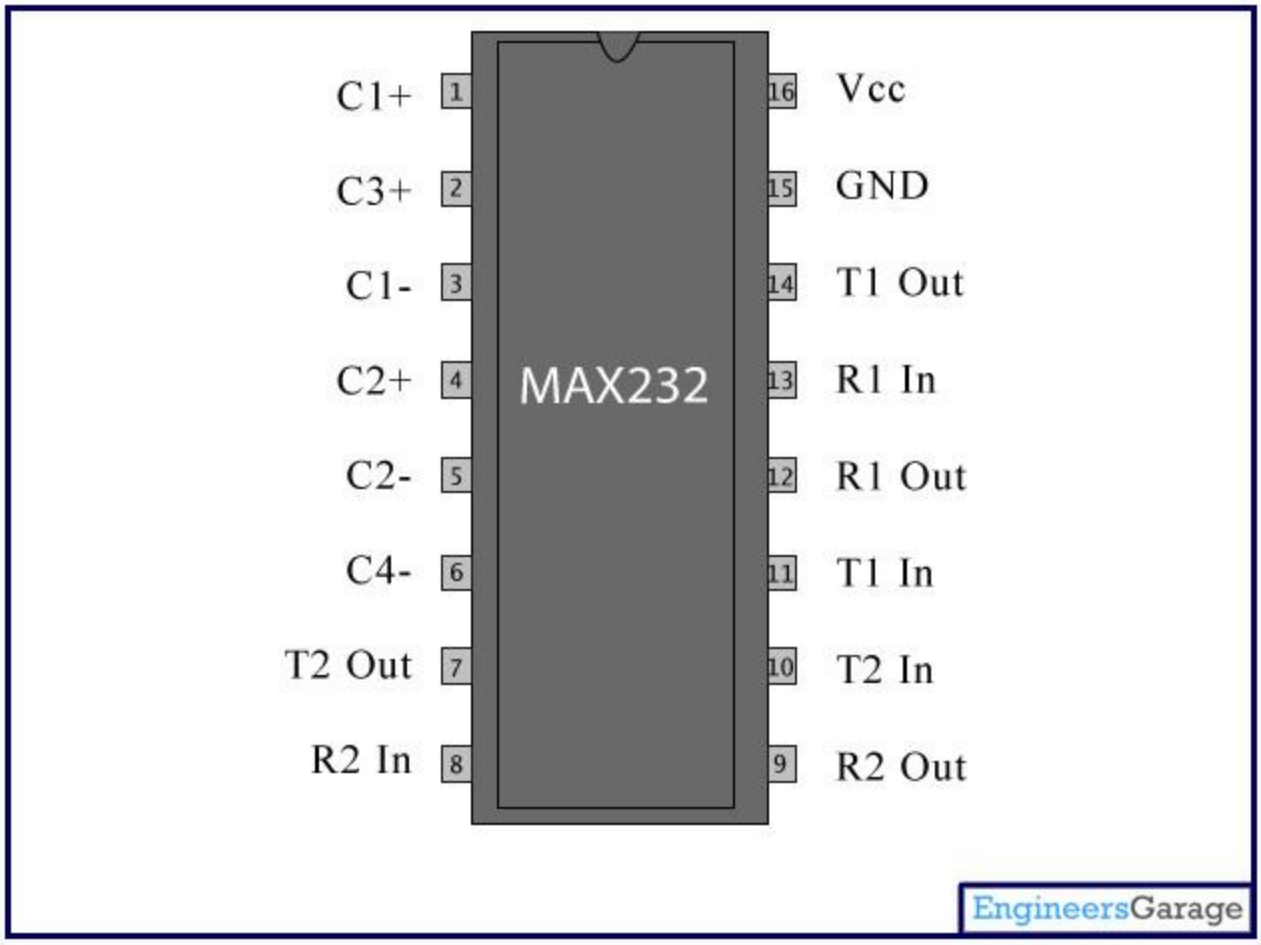
The MAX232 IC is used to convert the TTL/CMOS logic levels to RS232 logic levels during serial communication of microcontrollers with PC. The controller operates at TTL logic level (0-5V) whereas the serial communication in PC works on RS232 standards (-25 V to + 25V). This makes it difficult to establish a direct link between them to communicate with each other.

The intermediate link is provided through MAX232. It is a dual driver/receiver that includes a capacitive voltage generator to supply RS232 voltage levels from a single 5V supply. Each receiver converts RS232 inputs to 5V TTL/CMOS levels. These receivers (R_1 & R_2) can accept ±30V inputs. The drivers (T_1 & T_2), also called transmitters, convert the TTL/CMOS input level into RS232 level.

The transmitters take input from controller's serial transmission pin and send the output to RS232's receiver. The receivers, on the other hand, take input from transmission pin of RS232 serial port and give serial output to microcontroller's receiver pin. MAX232 needs four external capacitors whose value ranges from 1µF to 22µF.

Microcontroller	**MAX232**		**RS232**
Tx	$T_{1/2}$ In	$T_{1/2}$ Out	Rx
Rx	$R_{1/2}$ Out	$R_{1/2}$ In	Tx

Pin Diagram:


C1+
1
C3+
2
C1-
3
C2+
4
C2-
5
C4-
6
T2 Out
7
R2 In
8
MAX232
16
Vcc
15
GND
14
T1 Out
13
R1 In
12
R1 Out
11
T1 In
10
T2 In
9
R2 Out
EngineersGarage

Pin Description:

Pin No	Function	Name
1	Capacitor connection pins	Capacitor 1 +
2		Capacitor 3 +
3		Capacitor 1 -
4		Capacitor 2 +
5		Capacitor 2 -
6		Capacitor 4 -
7	Output pin; outputs the serially transmitted data at RS232 logic level; connected to receiver pin of PC serial port	T_2 Out
8	Input pin; receives serially transmitted data at RS 232 logic level; connected to transmitter pin of PC serial port	R_2 In
9	Output pin; outputs the serially transmitted data at TTL logic level; connected to receiver pin of controller.	R_2 Out
10	Input pins; receive the serial data at TTL logic level; connected to serial transmitter pin of controller.	T_2 In
11		T_1 In
12	Output pin; outputs the serially transmitted data at TTL logic level; connected to receiver pin of controller.	R_1 Out
13	Input pin; receives serially transmitted data at RS 232 logic level; connected to transmitter pin of PC serial port	R_1 In
14	Output pin; outputs the serially transmitted data at RS232 logic level; connected to receiver pin of PC serial port	T_1 Out
15	Ground (0V)	Ground
16	Supply voltage; 5V (4.5V – 5.5V)	Vcc

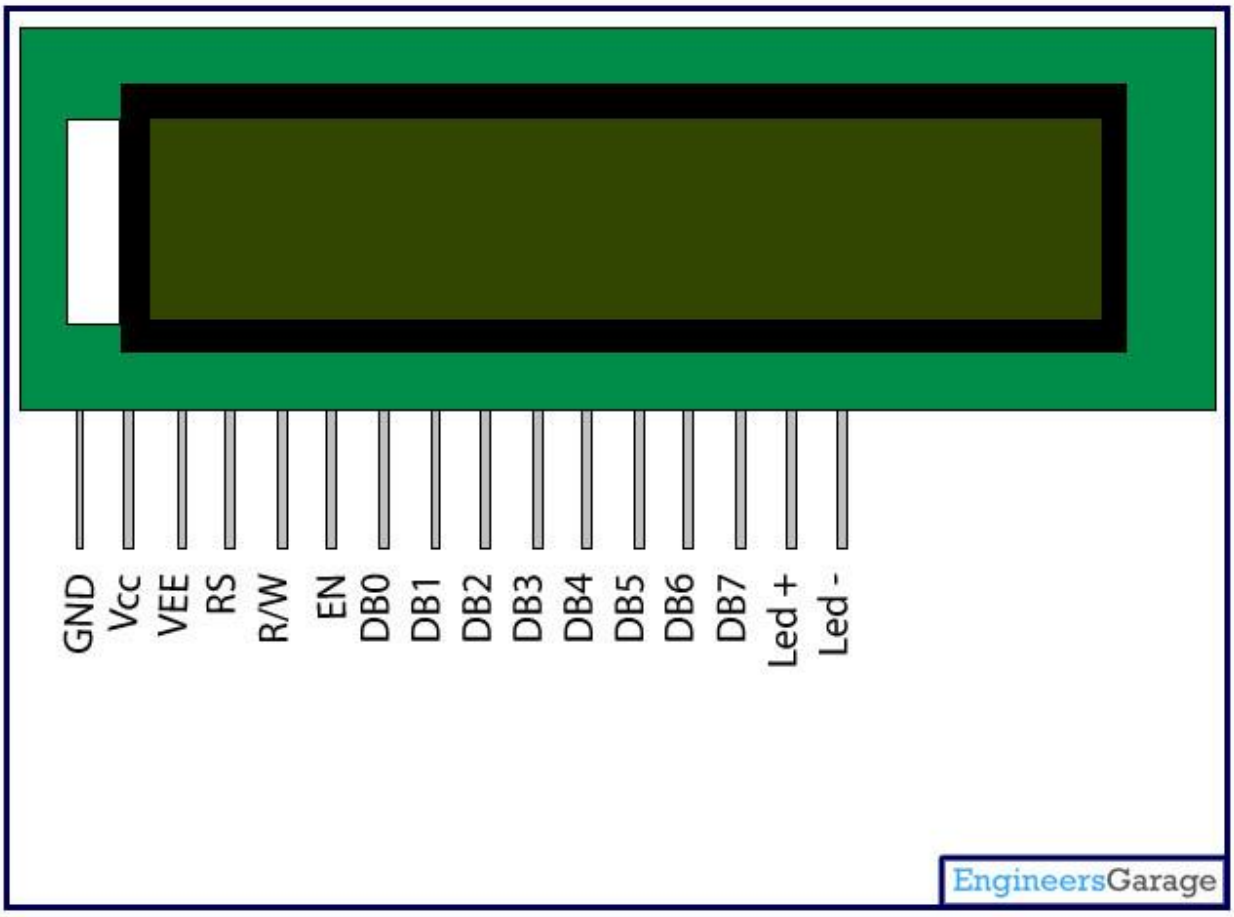
LCD

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

Pin Diagram:



Pin Description:

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-