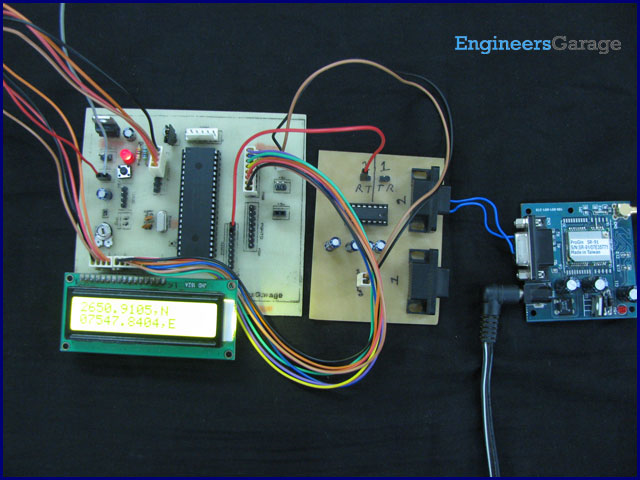
**[How to interface GPS with PIC18F4550 Microcontroller](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/gps-interface-circuit" \o "How to interface GPS with PIC18F4550 Microcontroller)**



[Global Positioning System](http://www.engineersgarage.com/articles/global-positioning-system-gps) is based on satellite navigation technology. A GPS Receiver provides the accurate location of an object in terms of latitude and longitude. Accurate time calculation with respect to GMT can also be done by using GPS. For more information on different data obtained through GPS, refer [GPS Receivers](http://www.engineersgarage.com/tutorials/gps-receivers-nmea-standards). Here a [PIC microcontroller](http://www.engineersgarage.com/articles/pic-microcontroller-tutorial) has been interfaced with a GPS module to extract its position information (location).

[GPS](http://www.engineersgarage.com/articles/global-positioning-system-gps) provides a lot of geographical information for a particular object like its latitude, longitude, direction of travel, GMT etc. This information are assembled in a particular string format which are to be decoded by GPS modems. A GPS modem gives the output data in a following string format called as [NMEA Format](http://www.engineersgarage.com/tutorials/gps-receivers-nmea-standards). A common GPS sentence ($GPGGA) has been explained below.

**$GPGGA,100156.000,2650.9416,N,07547.8441,E,1,08,1.0,442.8,M,-42.5,M,,0000\*71**

1.               A string always starts with a ‘**$**’ sign

2.               **GPGGA** : Global Positioning System Fix Data

3.                ‘**,**’ Comma indicates the separation between two values

4.               **100156.000** : GMT time as 10(hr):01(min):56(sec):000(ms)

5.               **2650.9416,N**: Latitude 26(degree) 50(minutes) 9416(sec) North

6.               **07547.8441,E**: Longitude 075(degree) 47(minutes) 8441(sec) East

7.               **1** : Fix Quantity 0= invalid data, 1= valid data, 2=DGPS fix

8.               **08** :  Number of satellites currently viewed.

9.               **1.0**: HDOP

10.           **442.8,M** : Altitude (Height above sea level in meter)

11.           **-42.5,M** :         Geoids height

12.           **\_\_** , DGPS data

13.           **0000** : DGPS data

14.           **\*71** : checksum

The main objective here is to find the location of the GPS Receiver in terms of latitude and longitude. The GPS module gives output data in RS232 logic level format. To convert the RS232 logic level into TTL, a line converter [MAX232](http://www.engineersgarage.com/electronic-components/max232-datasheet) has been connected between GPS module and [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller). (Also refer [PIC USART](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/eusart-circuit)) The circuit connection of GPS module with microcontroller is shown in the circuit diagram tab. The latitude and longitude data has been displayed on a [16x2 LCD interfaced to PIC](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/lcd-interfacing-text-circuit).

**Programming steps:**

1. Set the baud rate of [PIC’s USART](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/eusart-circuit) to 4800 bps.

2. Enable the SPEN and CREN bits (RCSTA register).

3. Receive the Serial data and compare with the string ‘$GPGGA,’ byte by byte.

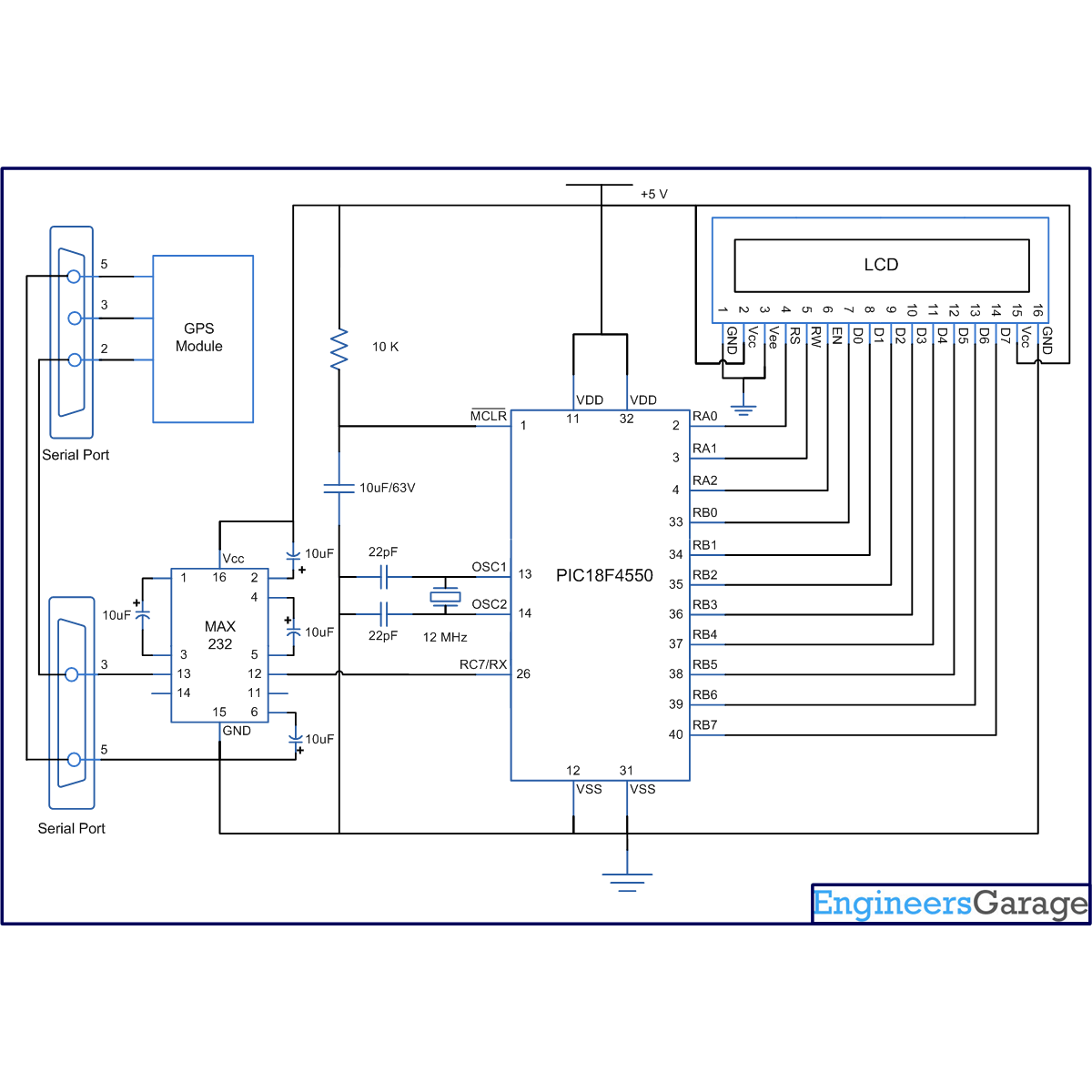
4. Wait for comma (,) as string gets matched.

5. Store the data which appears after the above comma into a string which will be the Latitude.

6. After another comma (,), store the data into another string which will be the Longitude.

7. Display both Latitude and Longitude data on LCD.

8. Repeat the steps 3 to 7 to update the GPS module’s positions on LCD.



// Program to Interface GPS with PIC18F4550 Microcontroller  
#define FREQ 12000000  
#define baud 4800  
#define spbrg\_value (((FREQ/64)/baud)-1)  
#define rs LATA.F0  
#define rw LATA.F1  
#define en LATA.F2  
#define lcdport LATB

unsigned char rx\_data();  
void lcd\_ini();  
void lcdcmd(unsigned char);  
void lcddata(unsigned char);

unsigned char longi\_data[12];  
unsigned char lati\_data[12];  
unsigned char data,value=0;  
unsigned int i=0,pos;

void main()  
{  
 TRISB=0; // Set Port B as output port  
 LATB=0;  
 TRISA=0;  
 LATA=0;  
 SPBRG=spbrg\_value; // Fill SPBRG register to set the baud rate  
 RCSTA.SPEN=1; // To activate serial port (Tx and Rx pins)   
 RCSTA.CREN=1; // To enable continuous reception  
 lcd\_ini();  
 while(1)  
 {  
 data=rx\_data(); // Check the string '$GPGGA,'  
 if(data=='$')  
 {  
 data=rx\_data();  
 if(data=='G')  
 {  
 data=rx\_data();  
 if(data=='P');  
 {  
 data=rx\_data();  
 if(data=='G');  
 {  
 data=rx\_data();  
 if(data=='G')  
 {  
 data=rx\_data();  
 if(data=='A')  
 {  
 data=rx\_data();  
 if(data==',')  
 {  
 data=rx\_data();  
 while(data!=',')  
 data=rx\_data();  
 for(i=0;data!='N';i++)  
 data=rx\_data();  
 lati\_data[i]=data; // Store the Latitude data  
 }  
 data=rx\_data();  
 if(data==',')  
 {  
 for(i=0;data!='E';i++)  
 {  
 data=rx\_data();  
 longi\_data[i]=data; // Store the Longitude data  
 }  
 }  
 i=0;  
 lcdcmd(0x80);  
 while(i<11)  
 {  
 lcddata(lati\_data[i]); // Print the Latitude data  
 i++;  
 }  
 i=0;  
 lcdcmd(0xC0);  
 while(i<12)  
 {  
 lcddata(longi\_data[i]); // Print the Longitude data  
 i++;  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 Delay\_ms(1000);  
 for(i=0;i<12;i++)  
 {  
 data=0;  
 lati\_data[i]=0;  
 longi\_data[i]=0;  
 }  
 }  
}

unsigned char rx\_data(void)  
{  
 while(PIR1.RCIF==0); // Wait until RCIF gets low  
 return RCREG; // Store data in Reception register  
}

void lcd\_ini()  
{  
 lcdcmd(0x38); // Configure the LCD in 8-bit mode, 2 line and 5x7 font  
 lcdcmd(0x0C); // Display On and Cursor Off  
 lcdcmd(0x01); // Clear display screen  
 lcdcmd(0x06); // Increment cursor  
 lcdcmd(0x80); // Set cursor position to 1st line, 1st column  
}

void lcdcmd(unsigned char cmdout)  
{  
 lcdport=cmdout; //Send command to lcdport=PORTB  
 rs=0;   
 rw=0;  
 en=1;  
 Delay\_ms(10);  
 en=0;  
}

void lcddata(unsigned char dataout)  
{  
 lcdport=dataout; //Send data to lcdport=PORTB  
 rs=1;  
 rw=0;  
 en=1;  
 Delay\_ms(10);  
 en=0;  
}