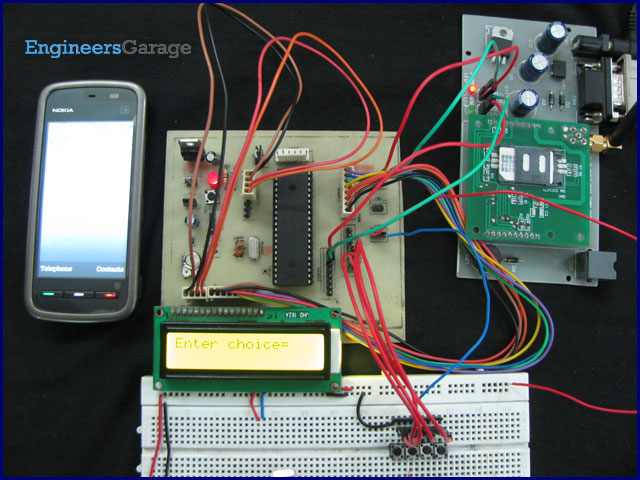
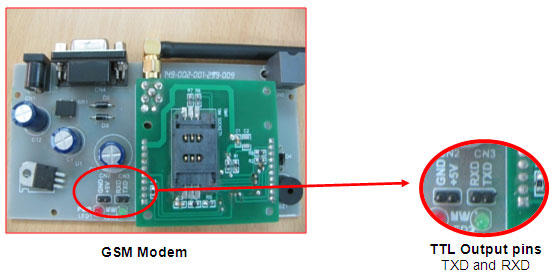
**[How to interface GSM Module with PIC18F4550 Microcontroller](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/interface-gsm-module-call-message-circuit" \o "How to interface GSM Module with PIC18F4550 Microcontroller)**



The Global System for Mobile ([GSM](http://www.engineersgarage.com/articles/gsm-gprs-modules)) communication is the Second Generation of mobile technology. Although the world is moving towards Third and Fourth generation but GSM has been the most successful and widespread technology in the communication sector. GSM technology paved a new way for mobile communication.

This project explains the interfacing of a [GSM Module](http://www.engineersgarage.com/articles/gsm-gprs-modules) with a [PIC microcontroller](http://www.engineersgarage.com/articles/pic-microcontroller-tutorial). It also covers a way **to dial a particular GSM mobile number as well as send a message to it using** [**AT Commands**](http://www.engineersgarage.com/tutorials/at-commands) with the help of [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller).

As explained earlier (refer [GSM interfacing with 8051](http://www.engineersgarage.com/microcontroller/8051projects/gsm-interface-8051-microcontroller-circuit-code)), a line converter [MAX232](http://www.engineersgarage.com/electronic-components/max232-datasheet) is employed to convert the RS232 logic data of [GSM Module](http://www.engineersgarage.com/articles/gsm-gprs-modules) to TTL logic so that it can be processed by the [microcontroller](http://www.engineersgarage.com/microcontroller). In this project, instead of RS232 logic data, TTL logic output has been taken and thus PIC18F4550 has been directly connected with GSM Modem without any line converter in between. The following diagram shows the TTL input and output of GSM modem used.

****

**Objectives:**

This project has following objectives which are fulfilled using [AT Commands](http://www.engineersgarage.com/tutorials/at-commands):

1.      Test the simple AT Command.

2.      Find out the IMEI number of the GSM Modem.

3.      **Connect a call to a GSM mobile number** (Dial a number).

4.      **Send a text message to a mobile number**.

The provision of these four operations has been provided by means of four tactile switches. Each switch corresponds to each of the above functions. AT Commands used to perform the above operations have been given below along with their output. (For complete list of AT Commands supported by GSM Modem, refer the list in tutorial on [AT Commands](http://www.engineersgarage.com/tutorials/at-commands).

**1.** Test the simple AT Command.

**AT** (Enter)

**OK**

**2.** Find out the IMEI number of the GSM modem.

**AT+GSN** (Enter)

**xxxxxxxxxxxxxxx**(15- digit unique IMEI number)

**OK**

**3.** Dial a number.

**ATDXXXXXXXXXX;** (Enter)(10-digit mobile number)

**OK**

**4.** Send a text message.

**AT+CMGS= “XXXXXXXXXX”** (Enter) (10-digit mobile number)

**>Hello ^z** (Enter message after ‘>’ and use Ctrl+z to terminate the message)

The program for the controller is written in a manner so that when a particular switch is pressed, its corresponding command will be called to execute. LCD is also interfaced with [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller) to display the results. The circuit connections of GSM module, LCD and switches with controller are shown in the circuit diagram tab. *Check the video for the execution of this project*.

**Programming Steps:**

1. Store all AT commands into strings.

2. Set the baud rate of microcontroller to 9600bps.

3. Enable serial port. (Refer [USART with PIC](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/eusart-circuit))

4. Enable the global and peripheral interrupt bits of the INTCON register.

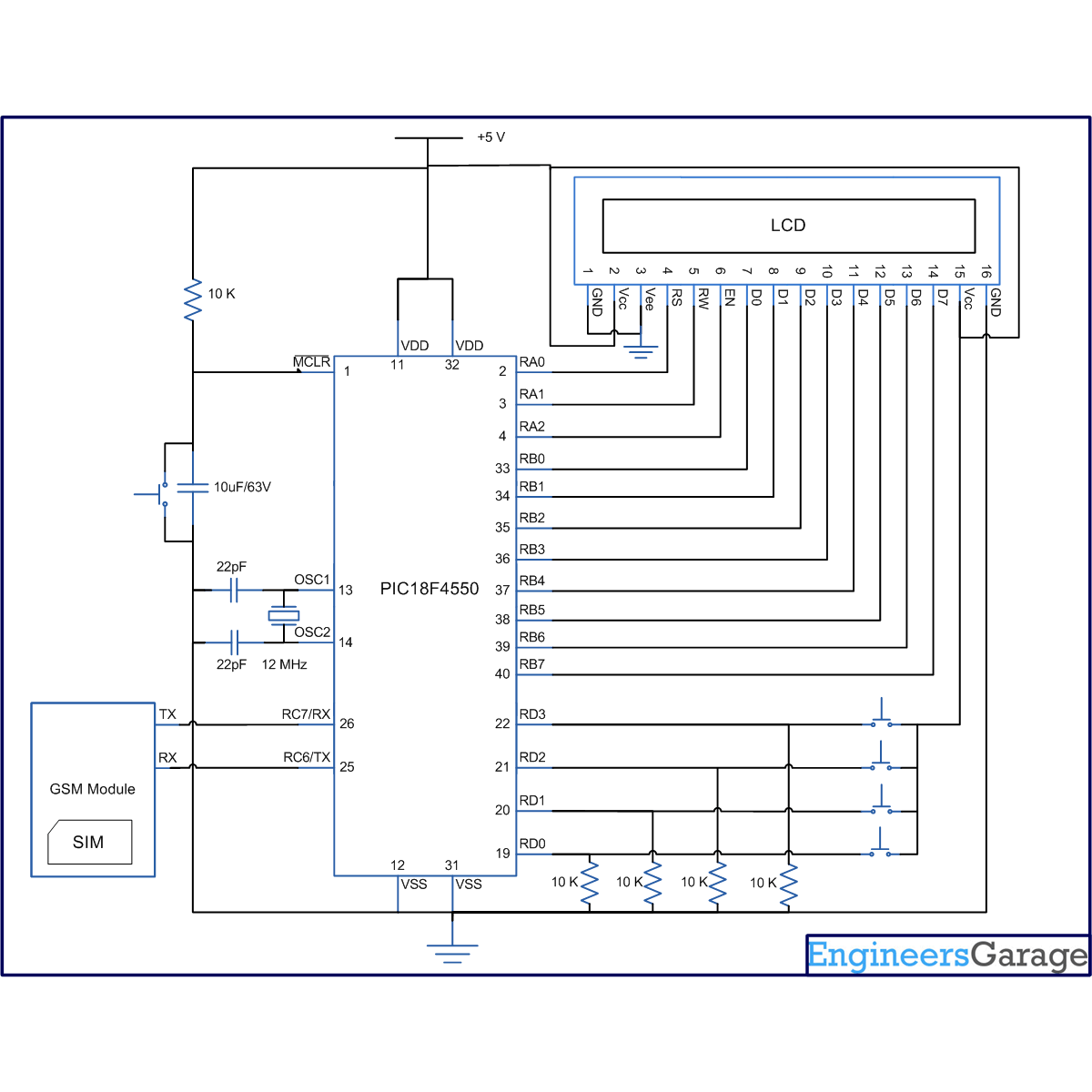
5. Configure Port D as input port.

6. Use *if* condition to detect the pressed switch.

7. As switch is pressed, process the corresponding AT command and transmit via USART.

8. Reception interrupt method is used to store the GSM output data into an array.

9. Display the stored value on the LCD. (Refer [displaying text on LCD using PIC](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/lcd-interfacing-text-circuit))



//Program to interface GSM Modem with PIC18F4550 Microcontroller  
//This code takes four choices as four inputs

//Choice 1 : Test the simple AT Command.  
//Choice 2 : Find out the IMEI number of the GSM Modem.  
//Choice 3 : Connect a call to a GSM mobile number.  
//Choice 4 : Send a text message to a mobile number.

#define FREQ 12000000  
#define baud 9600  
#define spbrg\_value (((FREQ/64)/baud)-1)  
#define rs LATA.F0  
#define rw LATA.F1  
#define en LATA.F2  
#define lcdport LATB

void tx\_data(unsigned char);  
unsigned char rx\_data();  
void lcd\_ini();  
void lcdcmd(unsigned char);  
void lcddata(unsigned char);  
void gsm\_cmd(unsigned char \*);  
void output(void);

unsigned char value=0;  
int i=0,j,k,temp,flag,choice;  
unsigned char \*starting\_text="Enter choice=";  
unsigned char \*dial\_text="Dialing...";  
unsigned char \*at\_cmd="AT";  
unsigned char \*imei\_cmd="AT+GSN";  
unsigned char \*call\_cmd="ATD9xxxxxxxxx;"; // Provide a 10-Digit Mobile Number  
unsigned char \*sms\_format="AT+CMGF=1";  
unsigned char \*sms\_write="AT+CMGS=\"xxxxxxxxxx\""; // 10-Digit Mobile Number  
unsigned char \*sms="Hello";  
unsigned char \*sms\_report="SMS Sent...";  
unsigned char sms\_terminate=0x1A;  
unsigned char enter=0x0D;  
unsigned char \*data;

void main()  
{  
 TRISB=0; // Set Port B as output port  
 LATB=0;  
 TRISA=0;  
 LATA=0;  
 TRISD=0xFF;  
 LATD=0;  
 SPBRG=spbrg\_value; // Fill SPBRG register to set the baud rate  
 RCSTA.SPEN=1; // To activate serial port (Tx and Rx pins)  
 TXSTA.TXEN=1; // Activate Transmissiom  
 RCSTA.CREN=1; // Activate Reception  
 PIE1.RCIE=1; // Enable Reception interrupt  
 INTCON.GIE=1; // Enable Global interrupt  
 INTCON.PEIE=1; // Enable Peripheral interrupt  
   
 lcd\_ini();  
 while(1)  
 {  
 k=0;  
 lcdcmd(0x80);  
 while(starting\_text[k]!='\0')  
 {  
 lcddata(starting\_text[k]);  
 k++;  
 }  
   
 //Check inputs

//Choice 1  
 if(PORTD.F0)  
 {  
 gsm\_cmd(at\_cmd);  
 output();  
 Delay\_ms(1000);  
 }

//Choice 2  
 if(PORTD.F1)  
 {  
 gsm\_cmd(imei\_cmd);  
 output();  
 Delay\_ms(1000);  
 }  
   
 //Choice 3   
 if(PORTD.F2)  
 {  
 gsm\_cmd(call\_cmd);  
 output();  
 Delay\_ms(1000);  
 }

//Choice 4  
 if(PORTD.F3)  
 {  
 gsm\_cmd(sms\_format);  
 output();  
 Delay\_ms(1000);

gsm\_cmd(sms\_write);  
 output();  
 Delay\_ms(1000);

gsm\_cmd(sms);  
 output();  
 tx\_data(0x1A);  
 Delay\_ms(1000);  
 }

}

}

void gsm\_cmd(unsigned char \*string)  
{  
 i=0;j=0;  
 while(string[i]!='\0')  
 {  
 temp=0;  
 if(string[i]==0x5C) // Not to send '\' cahracter  
 i++;  
 tx\_data(string[i]); // Send by serial communication  
 i++;  
 while(temp!=1);  
 }  
 temp=0;  
 tx\_data(enter); // Send ASCII code for 'Enter' key  
 while(temp!=1);  
}

void output(void) // To print data on LCD  
{  
 lcdcmd(0x01);  
 i=-1;flag=0;  
 while(i<j)  
 {  
 if(flag>1)  
 {  
 flag=0;  
 Delay\_ms(500);  
 lcdcmd(0x01);  
 lcdcmd(0x80);  
 }  
 if(data[i]==0x0A) // This condition is to avoid double Enter

// during execution of a command  
 {  
 flag++;  
 lcdcmd(0xc0);  
 }  
 if(data[i]=='>'||data[i]=='"') // Not to print this character  
 {  
 i++;  
 lcdcmd(0xc0);  
 }  
 if(data[i]!=0x0D&&data[i]!=0x0A&&data[i]!=0x1A) // Condition to print the data   
 // except 'Enter','New line' and 'Submit'  
   
 {  
 lcddata(data[i]);  
 i++;  
 }  
 else  
 i++;  
 Delay\_ms(300);  
 }  
 lcdcmd(0x01);  
}

void tx\_data(unsigned char serial\_data) // Transmit data function  
{  
 TXREG=serial\_data;  
 while(PIR1.TXIF==0);  
}

void interrupt()  
{  
 data[j]=RCREG; // Store the data into array when Reception interrupt occurs  
 value=RCREG;  
 j++;  
 temp=1;  
}

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;  
}