EFCom GPRS/GSM Shield

From Wiki

Contents

- 1 Introduction
- 2 Feature
- 3 Cautions
- 4 Schematic
- 5 Pin definition and Rating
- 6 Mechanic Dimensions
- 7 Usage
  - 7.1 Hardware Installation
  - 7.2 Light Status
  - 7.3 Getting Started - Fun with AT Commands
    - 7.3.1 Step 1: Creating a test setup for the GPRS Shield
    - 7.3.2 Step 2: Sending a text message (SMS)
    - 7.3.3 Step 3: Exploring further
  - 7.4 Connection Notes
  - 7.5 Example
- 8 Source Code Examples
  - 8.1 Sending SMS: using Hardware UART
  - 8.2 Making a call & Controlling GPIO and PWM pins
  - 8.3 GSM Library
- 9 Version Tracker
- 10 Bug Tracker
- 11 Additional Idea
- 12 Resources
- 13 How to buy
- 14 See Also
- 15 Licensing
- 16 External Links

Introduction

GPRS Shield - EFCom is an **ultra** compact and reliable wireless module. This GPRS Shield is compatible with all boards which have the same form factor (and pinout) as a standard Arduino Board. EFCom is base on **SIM900 4 (http://www.elecfreaks.com/store/4-frequency-ultra-gsmgprs-module-sim900-p-109.html) Frequency GPRS module**, which delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. SIM900 is a complete Quad-band GSM/GPRS module in a SMT type and designed with a very powerful single-chip processor integrating AMR926EJ-S core, allowing you to benefit from small dimensions and cost-effective solutions.

The GPRS Shield is configured and controlled via its UART using simple **AT commands (http://elecfreaks.com/store/download/datasheet/rf/SIM900/SIM900_AT%20Command%20Manual_V1.03.pdf) .** Just plug this shield on the Arduino/ Freaduino board, you could easy to use AT command control EFCom Shild. You can use the **2 jumper block** to connect the SIM900 URAT post to any pins within D0-D3(for Hardware/Software serial port). There is a switch on board , you can use it to select the connection of the UART port or Debug port , even be set on on Arduino, but by the switch and jumper block, the SIM900 can be connect to PC via FT233RL.

The shield allows you to achieve this via any of the three methods:

- Short Message Service
- Audio
- GPRS

We have layout of **all of** the SIM900' pins to the 2.54 standard pitch. Not just the UART port and debug port be layout. **The super capacitor** power supply for the RTC. The RTC can work more than 1 days by the power supply of super capacitor. So the SIM900 can keep the time and day when power off.

EFCom not only can use the S_PWR button for power on , but also can use the digital pin(D6) of Arduino to power on and reset(D5) the SIM900 module.

Model:EFCOM (http://www.elecfreaks.com/store/gprsgsm-shield-efcom-p-415.html)

Feature

- Fully compatible with Arduino / Freaduino and Mega.
- Free serial port connecting，you can select Hardware Serial port(D0/D1) control or Software Serial port(D2/D3) control it.
- SIM900 all pins breakout. Not just the UART port and debug port be layout, but also all pins on SIM900 be layout to the 2.54 standard pitch.
- Super capacitor power supply for the RTC.
- EFCom not only can use the button for power on , but also can use the digital pin of Arduino to power on and reset the SIM900 module.
- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+

```
- Class 4 (2 W @850/ 900 MHz)
- Class 1 (1 W @ 1800/1900MHz)
```

- Control via AT commands (GSM 07.07 ,07.05 and EFCOM enhanced AT Commands)
- SIM application toolkit
- Supply voltage range : 3.1 … 4.8V
- Low power consumption: 1.5mA(sleep mode)
- Operation temperature: -40°C to +85 °C
- Dimension:68.33x53.09mm(Same dimension of Arduino main board)

Cautions

- **Make sure add using 9V Charger for power supply for your Arduino board and EFCom, The 9V Charger we will provide to you with EFCom Shild. Because of the power supply range of SIM900 is from 3.2V to 4.8V. The transmitting burst will cause voltage drop and the power supply must be able to provide sufficient current up to 2A. The USB port can not supply such a large current.**

- **Make sure your SIM card is unlocked.**
- **The product is provided as is without an insulating enclosure. Please observe ESD precautions specially in dry (low humidity) weather.**
- **The factory default setting for the GPRS Shield UART is 19200 bps 8-N-1. (Can be changed using AT commands).**

Schematic

- PDF Schematics EFcomv1.2.pdf (http://elecfreaks.com/store/download/datasheet/wireless/EFcom_v1.2.pdf)

- PDF Schematics EFcomv1.0.pdf (http://elecfreaks.com/store/download/datasheet/wireless/EFComGPRS_v1.0.pdf)

Pin definition and Rating

Top-view:

Bottom-view

Mechanic Dimensions

Usage

Hardware Installation

- **Details**

LDO

Super RTC Cap and Line In

LCD5100 interface and Software SerialPort Jumper

SIM900

SMA and Antenna

SMA Bottom

SIM Card Connector

Light Status

LED	State	Function
Status	Off	Power Off
	On	Power On
Netlight	Off	SIM900 is not working
	64ms On/800ms Off	SIM900 does not find the network
	64ms On/3000ms Off	SIM900 find the network
	64ms On/300ms Off	GPRS communication

Block diagram for UART relay program running inside Arduino for patching a direct serial connection (19200bps 8-N-1) between GPRS Shield and a computer via FT232RL and ATmega328P

Getting Started - Fun with AT Commands

The GPRSShield comes with all the accessories that you would need to get started with sending data over the GSM network except for of course an Arduino board and a GSM SIM Card with a data plan subscription active on it. If you want to make voice calls, you would also require a headset with microphone.

Step 1: Creating a test setup for the GPRS Shield

So you received your GPRS Shield, what would be the first thing you would want to do with it? Send out a text (SMS) to your cell phone? or call up someone (headset required)? You can do all this by talking to the GPRS Shield using AT Commands - which is a special language that it understands. AT Commands are simple textual commands sent to the GPRS modem over its serial interface (UART), so you can pretty much use any serial terminal software to communicate with it - Window's built-in HyperTerminal, Arduino IDE's Serial Monitor or Serial Tool (http://elecfreaks.com/store/download/datasheet/Bluetooth/Serial_Port_Test.zip) . On a Mac you may use the command line application "screen" available through Terminal.
To experiment with AT commands, you would require a way to power up and communicate with your GPRS Shield. The best way to do this using an Arduino board is described below. Arduino One boards are confirmed to work as well (January 2012)

1. Install the SIM card in your GPRS Shield. You only require a data plan active on it if you want to use GPRS. If you are only looking to send text messages (SMSes) or make voice calls then it is not required.
2. Connect the antenna to the GPRS Shield.
3. Take an Arduino board and install the GPRS Shield over it.
4. Make sure the GPRS_TX & GPRS_RX jumpers on the GPRS Shield are mounted in SWSerial position - that is we want GPRS_TX to be connected to D2(RX) and GPRS_RX to D3(TX)
5. Connect the Arduino to your computer using a USB cable.
6. The ATmega328P microcontroller on board Duemilanove has only one UART which is used for communicating with the PC. What we need is an Arduino Sketch running inside the ATmega328P that would emulate a second serial port (UART) using software on the digital pins D2 and D3 and patch through all the communication between this second software serial port and the actual hardware serial port. By doing this, all the data coming from the computer (connected to the actual hardware UART) would be relayed as is to the GPRS Shield (connected to software UART) and we would be able to issue AT commands to control the GPRS

Shield. The block diagram outlining this scheme is shown below.

For developing such a program, we require installation of a new Arduino library - NewSoftSerial (Notice: From Arduino 1.0, use SoftwareSerial instead. See example code below). This uses the on-chip TIMER of ATmega328P in interrupt mode to emulate a second serial port. The library can be obtained from here (http://arduiniana.org/libraries/newsoftserial/) . It would be a ZIP file having a name like NewSoftSerial10c.zip. Extract the folder NewSoftSerial from within this .zip file and install it by placing it in your Arduino Installation folder which would be something like "C:\arduino-0023\libraries". Make sure to restart the Arduino IDE after the library. For users with Arduino 1.0, you can skip this step; the library SoftwareSerial is included in the installation already.

7. Once the library is installed, create a new sketch with the following code and download it into your Arduino board (For NewSoftSerial library):

```
//Serial Relay - Arduino will patch a
//serial link between the computer and the GPRS Shield
//at 19200 bps 8-N-1
//Computer is connected to Hardware UART
//GPRS Shield is connected to the Software UART

#include <NewSoftSerial.h>

NewSoftSerial mySerial(2, 3);

void setup()
{
  mySerial.begin(19200);               // the GPRS baud rate
  Serial.begin(19200);                 // the GPRS baud rate
}

void loop()
{
    if(Serial.available())
    {
       mySerial.print((unsigned char)Serial.read());
     }
    else  if(mySerial.available())
    {
       Serial.print((unsigned char)mySerial.read());
     }

}
```

8. If you are using Arduino 1.0 or later, use this sketch instead:

```
//Serial Relay - Arduino will patch a
//serial link between the computer and the GPRS Shield
//at 19200 bps 8-N-1
//Computer is connected to Hardware UART
//GPRS Shield is connected to the Software UART

#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3);

void setup()
{
  mySerial.begin(19200);               // the GPRS baud rate
  Serial.begin(19200);                 // the GPRS baud rate
}

void loop()
{
  if (mySerial.available())
    Serial.write(mySerial.read());
  if (Serial.available())
    mySerial.write(Serial.read());

}
```

Note : With Arduino 1.0 you should be able to use the SoftwareSerial library included with the distribution (instead of NewSoftSerial). However, you must be aware that the buffer reserved for incoming messages are hardcoded to 64 bytes in the library header, "SoftwareSerial.h": define _SS_MAX_RX_BUFF 64 // RX buffer size

This means that if the GPRS module responds with more data than that, you are likely to loose it with a buffer overflow! For instance, reading out an SMS from the module with "AT+CMGR=xx" (xx is the message index), you might not even see the message part because the preceding header information (like telephone number and time) takes up a lot of space. The fix seems to be to manually change _SS_MAX_RX_BUFF to a higher value (but reasonable so you don't use all you precious memory!)

9. Upload the sketch to the Arduino board.
10. Fire up your favorite serial terminal software, choose the COM port for Arduino, set it to operate at 19200 8-N-1
11. Press and hold the power button **a short while(Over 3 seconds)** on the GPRS Shield to turn it on. Wait half a minute for the GPRS Shield to connect to the network (Led Net will start blinking every 3 seconds or so). But there will not any information back in the monitor. If you want to see messages from the shield in the serial monitor such as, you need disable auto-bauding mode, using "AT+IPR=19200" . (Factory setting is AT+IPR=0 auto-bauding)

```
RDY

+CFUN: 1
```

```
+CPIN: READY

Call Ready
```

12. Now, type and send "AT" (without the quotes) followed by carriage return (enter key) to the Arduino board. The GPRS Shield should respond by sending back an "OK". This would mean that you have been able to successfully setup your GPRS Shield can can now play around with various AT Commands. (If you are using the readily available Serial Monitor in the Arduino IDE, you should set the line ending to "Carriage return" along with a baud rate of 19200).

Step 2: Sending a text message (SMS)

Now that our test setup is ready, let's play around with some AT Commands before moving on to programming the Arduino to do it by itself instead of doing it manually. To start off let's try sending an SMS

1. Create the setup as described in Step 1 above.
2. Through your serial terminal software, send **AT+CMGF=1** and press the Enter key. The GPRS Shield can send SMSes in two modes: Text mode and PDU (or binary) mode. Since we want to send out a human readable message, we will select the text mode. The GPRS Shield will respond with an **OK**.
3. Send **AT+CMGS="+918446043032"** and press the Enter key (include the quotes). This will instruct the GPRS Shield to start accepting text for a new message meant for the phone number specified (replace the number with the phone number of the target phone). The GPRS Shield will send a **>** signaling you to start typing the message. Please note that phone numbers specified as parameters in any AT Command must be in E.123 format (http://en.wikipedia.org/wiki/E.123) .
4. Start typing your message and when you are done, press **Ctrl + Z** keys on your keyboard(or sent a hex: 0x1a) when your cursor at the end of the message. The modem will accept the message and respond with an **OK**. A few moments later, the message should be received on the handset whose number you had specified. (There seems to be no way to send CTRL+Z in Arduino IDEs Serial Monitor - at least not on Mac. This is why a terminal program such as "screen" might work better for these tests.)

NOTE: If, in spite of following the steps as specified above, you aren't able to receive the message on the target handset, then it might be that you need to set the SMS Message Center number. Send the command **AT+CSCA="+919032055002"** and press the Enter Key. Send this command in between the AT+CMGF and AT+CMGS commands. Replace the phone number specified in the command above with the SMS Center number of your GSM Service Provider. The message center number is specific to each service provider (for example +919032055002 is the message center number for *Tata DoCoMo, Pune, India*). You can get the message center number by calling up the customer care center of the GSM Service Provider and asking them for it.

Step 3: Exploring further

Now that you have gotten a taste of how the AT Commands work, you can try out some more of them before moving on to developing sketches for Arduino to use the GPRS Shield. This involves creating a sketch for sending out these same sequence of AT Commands (on your behalf) out the serial port to the GPRS Shield to perform the same task of sending and SMS, making a call or sending data over a GPRS connection. You can go through the AT Commands reference manual (http://elecfreaks.com/store/download/datasheet/rf/SIM900/SIM900_AT%20Command%20Manual_V1.03.pdf) to figure out the sequence of commands required to do a particular task. If while developing an Arduino sketch, you find that the GPRS Shield isn't what you expected it to do, then you will need to check your AT Commands and their sequence. To do this, reload the serial relay sketch attached above in the getting started section into ATmega328P and type out the AT

Commands manually and check the output. The responses sent by the GPRS Shield will help you debug the AT Command sequence.

NOTE: A C program to perfom the same task as the serial relay sketch present above has also been developed and attached: File:Softuart relay atmega328p.zip. The program was developed on a Windows PC. AVRStudio4 (http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725) was used as the IDE and WinAVR (http://winavr.sourceforge.net/) was used as the compiler. The ZIP file contains an AVRStudio4 Project. The C compiler (WinAVR) will generate an Intel Hex (.hex). To upload this .hex file into an Arduino board outside of Arduino IDE would require a program which is able to communicate with the Arduino boards bootloader. XLoader (http://xloader.russemotto.com/) is such a program which runs on Windows can upload .hex files generated by various compiler into an Arduino Board.

Connection Notes

This section is a work in progress
Here are the list of connectors and pin headers present on-board the GPRS Shield:

- **SIM Card Holder** - 6 Pin Holder for SIM Cards. Both 1.8 volts and 3.0 volts SIM Cards are supported by SIM900 - the SIM card voltage type is automatically detected. Out of the 6 pins present on the SIM card holder, one is life unconnected and the other 5 are: SIM_VDD,SIM_RST,SIM_CLK,SIM_DATA,GND

Example

The projects and application examples.

We have added LCD5110 (http://www.elecfreaks.com/store/nokia-5110-lcd-black-on-blue-p-157.html) to GPRS Shield. There are two type of LCD5110 for you choosing, the Blue backlight LCM's pin compare with White backligt LCM's on the opposite.

- Nokia 5110 LCD Black on Blue (http://www.elecfreaks.com/store/nokia-5110-lcd-black-on-blue-p-157.html) (Default)

- Nokia 5110 LCD Black on White (http://www.elecfreaks.com/store/nokia-5110-lcd-black-on-white-p-114.html)

Demo code such as:

```
/*********************************************************************
**  Device: EFCom GPRS Module                                       **
**  File:   EF_GRPS.c                                               **
**  Description:                                                    **
**  This file is a sample code for your reference.                  **
**  Serial Relay - Arduino will patch a serial link between the     **
**  computer and the GPRS Shield at 19200 bps 8-N-1 Computer is     **
**  connected to Hardware UART GPRS Shield is connected to the      **
**  Software UART                                                   **
**                                                                  **
**  Copyright (C) 2011 ElecFreaks Corp.                             **
**  Created by ElecFreaks Robi.W /15 May 2012                       **
**  http://www.elecfreaks.com                                       **
*********************************************************************/

#include <LCD5110_Graph.h>
#include <NewSoftSerial.h>

LCD5110 myGLCD(9,10,11,12,13);

extern uint8_t SmallFont[];

#define rxPin 2
#define txPin 3

NewSoftSerial mySerial(rxPin, txPin);

void setup()
{
  myGLCD.InitLCD();
  myGLCD.setFont(SmallFont);

  mySerial.begin(19200);                 // the GPRS baud rate
  Serial.begin(19200);                  // the Serial port baud rate
  Serial.println("ElecFreaks EFComs Start");

  myGLCD.clrScr();
  myGLCD.print("ElecFreaks.com", CENTER, 0);
  myGLCD.print("EFCom GPRS", LEFT, 12);
  myGLCD.print("GSM Beta V1.0", LEFT, 30);
  myGLCD.update();
}

void loop()
{
    int i = 0;
    char someChar[32] = {0};
    if(Serial.available())
    {
        do{
          someChar[i++] = Serial.read();
        }while (Serial.available());
        mySerial.print(someChar);
     }
    else if(mySerial.available())
    {
       do{
          someChar[i++] = mySerial.read();
        }while (Serial.available());
        Serial.print(someChar);
     }
}
```

Source Code Examples

Here are some source code examples.

Sending SMS: using Hardware UART

The Arduino sketch for sending an SMS is presented below. It has been tested on Arduino Duemilanove but will work on any compatible variant. Please note that this sketch uses the hardware UART of ATmega328P. Please follow the following steps for running this sketch.

1. With the GPRS Shield removed, download this sketch into your Arduino. The GPRS Shield must be removed so that it doesn't interfere with the programming of Arduino which takes place over the Hardware UART (using FT232RL).
2. Disconnect the Arduino from USB port to remove power to it.
3. Set the Serial Port jumpers on the GPRS Shield in Xduino position (i.e. Arduino's RX connected to GPRS_TX and TX of Arduino connected to GPRS_RX)
4. Connect the antenna to the GPRS Shield and insert the SIM Card.
5. Mount the GPRS Shield on Arduino.
6. Apply power to the Arduino using USB port or via external power supply.
7. Switch on the GPRS Shield by using the power switch. Wait till the Network LED (D1) starts blinking.
8. Using a pen or a plastic tweezer access the reset switch on the Arduino Board and reset the microcontroller to run the sketch from the start. Do not try resetting the Arduino by removing and applying power to it as this will turn off the GPRS Shield.
9. If nothing goes wrong, the SMS will be received on receiver's handset.

```
void setup()
{
  Serial.begin(19200);  //Default serial port setting for the GPRS modem is 19200bps 8-N-1
  Serial.print("\r");
  delay(1000);                    //Wait for a second while the modem sends an "OK"
  Serial.print("AT+CMGF=1\r");    //Because we want to send the SMS in text mode
  delay(1000);

  //Serial.print("AT+CSCA=\"+919032055002\"\r");  //Setting for the SMS Message center number,
  //delay(1000);                                  //uncomment only if required and replace with
                                                  //the message center number obtained from
                                                  //your GSM service provider.
                                                  //Note that when specifying a tring of characters
                                                  // " is entered as \"

  Serial.print("AT+CMGS=\"+918446043032\"\r");    //Start accepting the text for the message
                                                  //to be sent to the number specified.
                                                  //Replace this number with the target mobile number.
  delay(1000);
  Serial.print("SIM900 and Arduino say Hi!\r");   //The text for the message
  delay(1000);
  Serial.print(26,BYTE);  //Equivalent to sending Ctrl+Z
}

void loop()
{
      //We just want to send the SMS only once, so there is nothing in this loop.
      //If we put the code for SMS here, it will be sent again and again and cost us a lot.
}
```

Making a call & Controlling GPIO and PWM pins

The sample source code below shows how one can make a call and control the GPRS Shield's General Purpose I/O pins as well as the PWM pins

```
//Source code sample showing how to make a call (use headphones and mic to talk),
//and how to control the General Purpose I/O pins and PWM

#include <NewSoftSerial.h>

NewSoftSerial mySerial(7, 8);

void setup()
{
  mySerial.begin(19200);               // the GPRS baud rate
  Serial.begin(19200);               // the GPRS baud rate
  delay(2000);
}

void loop()
{
  int count=0;

  mySerial.println("ATDxxxxxxxxx;"); // xxxxxxxxx is the number you want to dial, Noice the ";" in the end
  delay(2000);
     while(1)
     {
       mySerial.println("AT+SPWM=2,63,100");// set PWM 2 PIN
       delay(100);
       mySerial.println("AT+SPWM=1,63,100");
       delay(100);
       mySerial.println("AT+SGPIO=0,1,1,1");// set GPIO 1 PIN to 1
```

```
        delay(100);
        mySerial.println("AT+SGPIO=0,2,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,3,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,4,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,5,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,6,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,7,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,8,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,9,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,10,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,11,1,1");
        delay(100);
        mySerial.println("AT+SGPIO=0,12,1,1");

        delay(500);

        mySerial.println("AT+SPWM=1,63,0");
        delay(100);
        mySerial.println("AT+SPWM=2,63,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,1,1,0"); // set GPIO 1 PIN to 0
        delay(100);
        mySerial.println("AT+SGPIO=0,2,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,3,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,4,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,5,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,6,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,7,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,8,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,9,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,10,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,11,1,0");
        delay(100);
        mySerial.println("AT+SGPIO=0,12,1,0");
        delay(500);

        count++;

        if(count==5)
        {
          mySerial.println("ATH"); //end the call.
          if(mySerial.available())
         {
            Serial.print((unsigned char)mySerial.read());

          }
        }
      }

}
```

GSM Library

The Library (http://elecfreaks.com/store/download/datasheet/wireless/GSM_Shield2.zip) of GSM Shield example by Boris Landoni. But the library used

```
NewSoftSerial mySerial(4, 5);  //rx, tx
```

D4,D5 for Software Serial port, We need change it to

```
NewSoftSerial mySerial(2, 3);  //rx, tx
```

```
/* GSM Shield example

 created 2011
 by Boris Landoni

 This example code is in the public domain.


 http://www.open-electronics.org
```

```
 http://www.futurashop.it
 */

#include <GSM_Shield.h>
//for enable disable debug rem or not the string       #define DEBUG_PRINT
// definition of instance of GSM class
GSM gsm;



void setup() {
  Serial.begin(9600);
  Serial.println("system startup");
  //gsm.InitSerLine(9600);   //initialize serial 1
  gsm.TurnOn(9600);              //module power on
  //gsm.InitSerLine(9600);   //initialize serial 1
  gsm.InitParam(PARAM_SET_1);//configure the module
  gsm.Echo(1);               //enable AT echo
}


void loop()
{

    int call;
    call=gsm.CallStatus();
    switch (call){
      case CALL_NONE:
        Serial.println("no call");
        break;
      case CALL_INCOM_VOICE:
        Serial.println("incoming voice call");
        delay(5000);
        gsm.PickUp();
        break;
      case CALL_ACTIVE_VOICE:
        Serial.println("active voice call");
        delay(5000);
        gsm.HangUp();
        break;
      case CALL_NO_RESPONSE:
        Serial.println("no response");
        break;
    }
    delay(1000);

}
```

Version Tracker

Revision	Descriptions	Release
v1.2	Present Version	date
v1.0	Beta / Initial public release	date

Bug Tracker

Bug Tracker is the place you can publish any bugs you think you might have found during use. Please write down what you have to say, your answers will help us improve our

products.

Contrasts with Beta

Additional Idea

The Additional Idea is the place to write your project ideas about this product, or other usages you've found. Or you can write them on Projects page.

Resources

- EFcom_v1.2.pdf (http://elecfreaks.com/store/download/datasheet/wireless/EFcom_v1.2.pdf)
- EFcom_v1.0.pdf (http://elecfreaks.com/store/download/datasheet/wireless/EFComGPRS_v1.0.pdf)
- GSM Shield Library (http://elecfreaks.com/store/download/datasheet/wireless/GSM_Shield2.zip)
- SIM900 Reference Design Guide (http://elecfreaks.com/store/download/datasheet/rf/SIM900/SIM900_EVB%20kit_User%20Guide_V1.03.pdf)
- SIM900_AT Command Manual_V1.03 (http://elecfreaks.com/store/download/datasheet/rf/SIM900/SIM900_AT%20Command%20Manual_V1.03.pdf)
- SIM900_Hardware Design_V2.00 (http://elecfreaks.com/store/download/datasheet/rf/SIM900/SIM900_Hardware%20Design_V2.00.pdf)
- SMI900_Materia (http://elecfreaks.com/store/download/datasheet/rf/SIM900/SMI900_Material.zip)

How to buy

Click here to buy: http://www.elecfreaks.com/store/gprs-shield-efcom-p-415.html

See Also

Other related products and resources.

Licensing

This documentation is licensed under the Creative Commons Attribution-ShareAlike License 3.0 (http://creativecommons.org/licenses/by-sa/3.0/) Source code and libraries are

licensed under GPL/LGPL (http://www.gnu.org/licenses/gpl.html) , see source code files for details.

External Links

Links to external webpages which provide more application ideas, documents/datasheet or software libraries

Retrieved from "http://www.elecfreaks.com/wiki/index.php?title=EFCom_GPRS/GSM_Shield"
Categories: EFCom | Wireless | All Products

- This page was last modified on 29 October 2012, at 01:24.
- This page has been accessed 6,345 times.

- About Wiki
- Disclaimers