

# AN1604E ATK-SIM800C GSM/GPRS 模块使用说明

本应用文档（AN1604E）将教大家如何在 **ALIENTEK 阿波罗 STM32F429 开发板** 上使用 ATK-SIM800C GSM/GPRS 模块。

本文档分为如下几部分：

- 1, ATK-SIM800C GSM/GPRS 模块简介
- 2, 硬件连接
- 3, 软件实现
- 4, 验证

## 1、ATK-SIM800C GSM/GPRS 模块简介

ATK-SIM800C-V15(V15 是版本号, 下面均以 ATK-SIM800C 表示该产品)是 ALIENTEK 推出的一款高性能工业级 GSM/GPRS 模块（开发板）。ATK-SIM800C 模块板载 SIMCOM 公司的工业级四频 GSM/GPRS 模块：SIM800C, 工作频段四频：850/900/1800/1900MHz, 可以低功耗实现语音、SMS(短信)、MMS(彩信)、蓝牙数据信息的传输。

ATK-SIM800C 模块支持 RS232 串口和 LVTTTL 串口(即支持 3.3V/5V 系统), 并带硬件流控制, 支持 5V~24V 的超宽工作范围, 使得本模块可以非常方便的与您的产品进行连接, 从而给您的产品提供包括语音、短信、彩信、蓝牙和 GPRS 数据传输等功能。

### 1.1 模块资源简介

ATK-SIM800C 模块是 ALIENTEK 开发的一款高性能工业级 GSM/GPRS 模块（开发板），功能完善, 尤其适用于需要语音/短信/GPRS 数据/蓝牙通信服务的各种领域, 其资源图如图 1.1.1 所示：

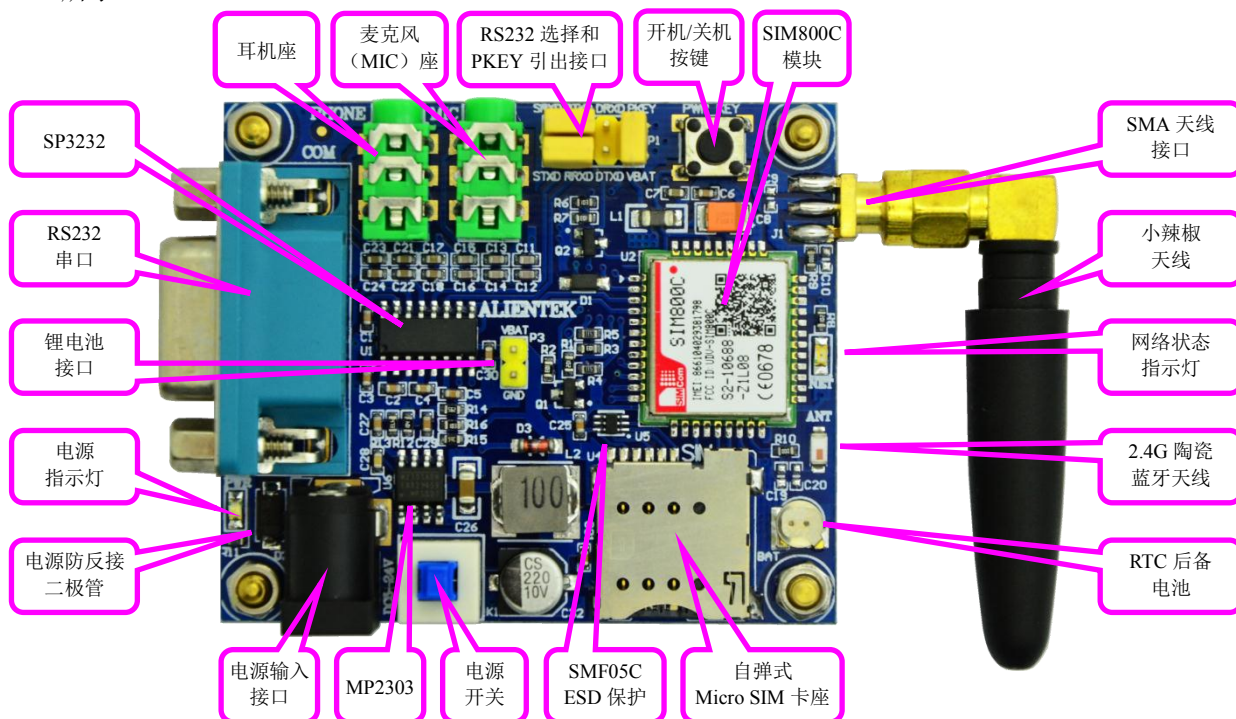


图 1.1.1 ATK-SIM800C 模块资源图

从图 1.1.1 可以看出，ATK-SIM800C 模块不但外观漂亮，而且功能齐全，模块尺寸（不

算天线部分)为 62mm\*52.5mm,并带有安装孔位,非常小巧,并且利于安装,可方便应用于各种产品设计。

ALIENTEK ATK-SIM800C 模块(开发板)板载资源如下:

- ◆ GSM 模块: SIM800C
- ◆ 1 个 RTC 后备电池
- ◆ 1 个麦克风接口
- ◆ 1 个耳机接口
- ◆ 1 个 RS232 选择和 PKEY 引出接口
- ◆ 1 个 RS232 串口
- ◆ 1 个锂电池接口
- ◆ 1 个电源输入接口
- ◆ 1 个电源指示灯(蓝色)
- ◆ 1 个电源开关
- ◆ 1 个自弹式 Micro SIM 卡座
- ◆ 1 个 SMA 天线接口并配套小辣椒天线
- ◆ 1 个 2.4G 蓝牙陶瓷天线
- ◆ 1 个开机/关机按键
- ◆ 1 个网络状态指示灯(红色)

ATK-SIM800C 模块(开发板)采用工业级标准设计,特点包括:

- 板载 RS232 串口(支持硬件流控制),方便与 PC/工控机等设备连接;
- 板载 3.5mm 耳机和麦克风座,方便进行语音通信开发;
- 板载高效同步降压电路,转换效率高达 90%,支持超宽电压工作范围(5~24V),非常适合工业应用;
- 板载电源防反接保护, SIM 卡 ESD 保护,保护功能完善;
- 板载 RTC 后备电池(XH414H-IV01E),无需担心掉电问题;
- 板载小辣椒天线和陶瓷天线,能有效提高信号接收能力;
- 采用国际 A 级 PCB 料,沉金工艺加工,稳定可靠;
- 采用全新元器件加工,纯铜镀金排针,坚固耐用;
- 人性化设计,各个接口都有丝印标注,使用起来一目了然;接口位置设计安排合理,方便顺手。
- PCB 尺寸为 62mm\*52.5mm,并带有安装孔位,小巧精致;

ATK-SIM800C 模块的资源介绍,我们就介绍到这里,详细的介绍,请看《ATK-SIM800C GSM(GPRS)模块用户手册\_V1.0》相关章节。

## 1.2 模块使用

本文档,我们将介绍大家如何通过 ALIENTEK 阿波罗 STM32F429 开发板连接 ATK-SIM800C 模块,实现:拨号测试(电话的拨打和接听)、短信测试(读短信和写短信)和 GPRS 测试(TCP 通信和 UDP 通信)、蓝牙测试(SPP 通信)等 4 大功能,本节我们将介绍要实现这些功能所需要的相关知识。

### 1.2.1 AT 指令简介

AT 即 Attention, AT 指令集是从终端设备(Terminal Equipment, TE)或数据终端设备(Data Terminal Equipment, DTE)向终端适配器(Terminal Adapter, TA)或数据电路终端设备(Data Circuit Terminal Equipment, DCE)发送的。通过 TA, TE 发送 AT 指令来控制移动台(Mobile Station, MS)的功能,与 GSM 网络业务进行交互。用户可以通过 AT 指令进行呼叫、短信、电话本、

数据业务、传真等方面的控制。

AT 指令必须以"AT"或"at"开头，以回车 (<CR>) 结尾。模块的响应通常紧随其后，格式为：<回车><换行><响应内容><回车><换行>。

我们通过串口调试助手 XCOM 来测试一下，打开：ATK-SIM800C 模块配套资料\3，配套软件\串口调试助手\XCOM V2.0.exe，选择正确的 COM 号（连接到 ATK-SIM800C 模块的 COM 端口，我电脑是 COM8），然后设置波特率为 115200，**勾选发送新行（必选！即 XCOM 自动添加发送回车换行功能）**，然后发送 AT 到 ATK-SIM800C 模块，如图 1.2.1.1 所示：

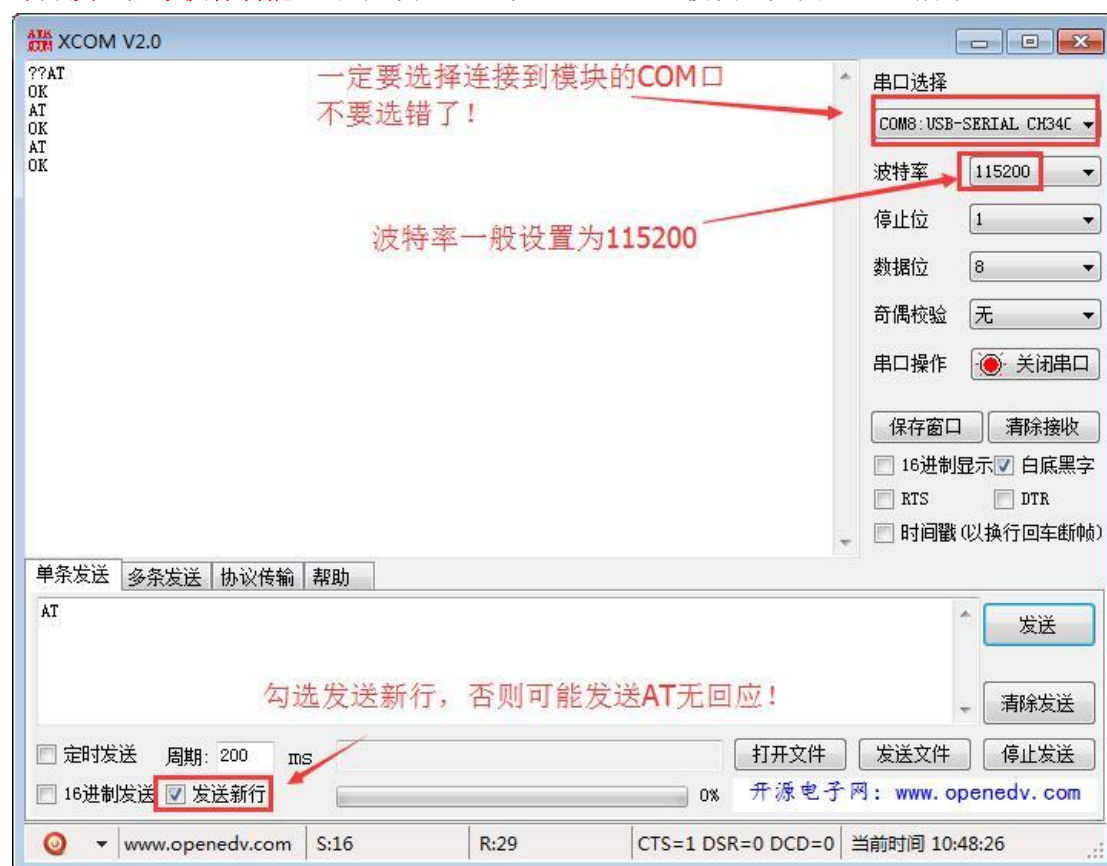


图 1.2.1.1 AT 指令测试

图 1.2.1.1 中，我们发送了 3 次 AT 指令，第一次看到有乱码，这是因为模块上电后，还没有实现串口同步，在收到第一次数据（不一定要 AT 指令）后，模块会自动实现串口同步（即自动识别出了通信波特率），后续通信就不会出现乱码了。因为 SIM800C 具有自动串口波特率识别功能（识别范围：1200-115200），所以我们的电脑（或设备）可以随便选择一个波特率（不超过识别范围即可），来和模块进行通信，这里我们选择最快的 115200。

从图 1.2.1.1 可以看出，我们现在已经和 SIM800C 模块进行通信了，我们通过发送不同的 AT 指令，就可以实现对 SIM800C 的各种控制了。

SIM800C 模块提供的 AT 命令包含符合 3GPP TS 27.005、3GPP TS 27.007 和 ITU-T Recommendation V.25ter 的指令，以及 SIMCOM 自己开发的指令。接下来我们介绍几个常用的 AT 指令：

#### 1, AT+CPIN?

该指令用于查询 SIM 卡的状态，主要是 PIN 码，如果该指令返回：+CPIN:READY,则表明 SIM 卡状态正常，返回其他值，则有可能是没有 SIM 卡。

#### 2, AT+CSQ

该指令用于查询信号质量，返回 SIM800C 模块的接收信号强度，如返回：+CSQ: 24,0，

表示信号强度是 24（最大的有效值是 31）。如果信号强度过低，则要检查天线是否接好了？

### 3, AT+COPS?

该指令用于查询当前运营商，该指令只有在连上网络后，才返回运营商，否则返回空，如返回：+COPS:0,0,"CHINA MOBILE",表示当前选择的运营商是中国移动。

### 4, AT+CGMI

该指令用于查询模块制造商，如返回：SIMCOM\_Ltd，说明 SIM800C 模块是 SIMCOM 公司生产的。

### 5, AT+CGMM

该指令用于查询模块型号，如返回：SIMCOM\_SIM800C，说明模块型号是 SIM800C。

### 6, AT+CGSN

该指令用于查询产品序列号（集 IMEI 号），每个模块的 IMEI 号都是不一样的，具有全球唯一性，如返回：866104023267696，说明模块的产品序列号是：866104023267696。

### 7, AT+CNUM

该指令用于查询本机号码，必须在 SIM 卡在位的时候才可以查询，如返回：+CNUM:"";"136\*\*\*\*\*";"129",7,4，则表明本机号码为：136\*\*\*\*\*。另外，不是所有的 SIM 卡都支持这个指令，有个别 SIM 卡无法通过此指令得到其号码。

### 8, ATE1

该指令用于设置回显模式（默认开启），即模块将收到的 AT 指令完整的返回给发送端，启用该功能，有利于调试模块。如果不需要开启回显模式，则发送 ATE0 指令即可关闭，这样收到的指令将不再返回给发送端，这样方便程序控制。

### 9, AT+CGMR

该指令用于查询固件版本序列号，如返回：Revision:1418B02SIM800C32\_BT,说明模块的固件版本序列号是 1418B02SIM800C32\_BT，flash 大小是 32Mbit、支持蓝牙通信功能。

以上就是我们介绍的几个常用的 AT 指令，当然还有其他一些常用的 AT 指令，比如 ATD/ATA/ATH 等，我们在后面的章节会慢慢介绍。关于 SIM800C 详细的 AT 指令介绍，请参考：[ATK-SIM800C 模块配套资料\4, SIM800C 模块资料\SIM800 Series\\_AT Command Manual\\_V1.09.pdf](#) 这个文档。

发送给模块的指令，如果执行成功，则会返回对应信息和“OK”，如果执行失败/指令无效，则会返回“ERROR”。

## 1.2.2 拨打/接听电话

使用 ATK-SIM800C 模块可以非常方便的进行拨打和接听电话。实现拨号和接听电话，常用的指令有：ATE0/ATD/ATA/ATH/AT+CLIP/AT+VTS 等 6 条 AT 指令。

ATE0，用于关闭回显，在通过电脑串口调试助手调试的时候，我们发送：ATE1，开启回显，可以方便调试，但是我们通过单片机程序控制的时候，用不到回显功能，所以发送：ATE0，将其关闭。

ATD，用于拨打任意电话号码，格式为：ATD+号码+；，末尾的‘；’一定要加上，否则不能成功拨号，如发送：ATD10086；，即可实现拨打 10086。

ATA，用于应答电话，当收到来电的时候，给模块发送：ATA，即可接听来电。

ATH，用于挂断电话，要想结束正在进行的通话，只需给模块发送：ATH，即可挂断。

AT+COLP，用于设置被叫号码显示，这里我们通过发送：AT+COLP=1，开启被叫号码显示，当成功拨通的时候（被叫接听电话），模块会返回被叫号码。

AT+CLIP，用于设置来电显示，通过发送：AT+CLIP=1，可以实现设置来电显示功能，模块接收到来电的时候，会返回来电号码。

AT+VTS, 产生 DTMF 音, 该指令只有在通话进行中才有效, 用于向对方发送 DTMF 音, 比如在拨打 10086 查询的时候, 我们可以通过发送: AT+VTS=1, 模拟发送按键 1。

以上就是在拨打/接听电话时经常用到的几条指令, 通过这几条指令, 就可以实现电话的拨打和接听了, 不过首先要保证模块成功接入到 GSM 网络, 通过发送: AT+COPS?, 如果返回: +COPS: 0,0,"CHINA MOBILE", 则说明模块成功连接到了 GSM 网络, 可以正常使用了, 网络运营商为"CHINA MOBILE" (中国移动)。

这些指令的使用示例可以参考《ATK-SIM800C GSM (GPRS) 模块用户手册\_V1.0》2.3.3 节。

### 1.2.3 短信的读取与发送

使用 ATK-SIM800C 模块, 我们可以很方便的进行中英文短信的读取与发送。短信的读取与发送将用到的指令有: AT+CNMI/ AT+CMGF / AT+CSCS / AT+CSMP / AT+CMGR/AT+CMGS/ AT+CPMS 等 7 条 AT 指令。

AT+CNMI, 用于设置新消息指示。发送: AT+CNMI=2,1, 设置新消息提示, 当收到新消息, 且 SIM 卡未满的时候, SIM800C 模块会通过串口输出数据, 如: +CMTI: "SM",2, 表示收到接收到新消息, 存储在 SIM 卡的位置 2。

AT+CMGF, 用于设置短消息模式, SIM800C 支持 PDU 模式和文本 (TEXT) 模式等 2 种模式, 发送: AT+CMGF=1, 即可设置为文本模式。

AT+CSCS, 用于设置 TE 字符集, 默认的为 IRA, 国际标准字符集, 在发送纯英文短信的时候, 发送: AT+CSCS="GSM", 设置为缺省字符集即可。在发送中英文短信的时候, 需要发送: AT+CSCS="UCS2", 设置为 16 位通用 8 字节倍数编码字符集。

AT+CSMP, 用于设置短消息文本模式参数, 在使用 UCS2 方式发送中文短信的时候, 需要发送: AT+CSMP=17,167,2,25, 设置文本模式参数。

AT+CMGR, 用于读取短信, 比如发送: AT+CMGR=1, 则可以读取 SIM 卡存储在位置 1 的短信。

AT+CMGS, 用于发送短信, 在"GSM"字符集下, 最大可以发送 180 个字节的英文字符, 在"UCS2"字符集下, 最大可以发送 70 个汉字 (包括字符/数字)。

AT+CPMS, 用于查询/设置优选消息存储器, 通过发送: AT+CPMS?, 可以查询当前 SIM 卡最大支持多少条短信存储, 以及当前存储了多少条短信等信息。如返回: +CPMS:"SM\_P",1,50,"SM\_P",1,50,"SM\_P",1,50, 表示当前 SIM 卡最大存储 50 条信息, 目前已经有 1 条存储的信息。

以上就是短信读取与发送需要用到的一些 AT 指令, 这些指令的使用示例可以参考《ATK-SIM800C GSM(GPRS)模块用户手册\_V1.0》2.3.4 节。

为方便实现中英文短信的读取与发送, 本文档例程采用文本模式 (AT+CMGF=1)、UCS2 编码字符集 (AT+CSCS="UCS2"), 这样电话号码和短信内容, 全部是采用 UNICODE 编码的字符串。在读取短信的时候, 需要将模块返回的 UNICODE 编码字符串转换为 GBK/ASCII 码, 以便显示 (我们的例程只支持 GBK/ASCII 编码的汉字/字符显示)。而在发送短信的时候, 需要将 GBK/ASCII 编码的电话号码和短信内容转换为 UNICODE 编码的字符串, 发送给 ATK-SIM800C 模块, 实现中英文短信的发送。

在《ATK-SIM800C GSM(GPRS)模块用户手册\_V1.0》2.3.4 节里面, 我们使用了一个汉字 Unicode 互换工具的软件来实现汉字和 UNICODE 的互换, 而在本文档例程里面, 我们要在开发板液晶上面显示短信内容, 而液晶只支持 GBK 编码的汉字显示, 所以我们需要一个 GBK/UNICODE 互换编码表, 通过查表来实现 UNICDOE 和 GBK 的互换。这里我们利用 FATFS 提供的 cc936.c 里面的数组 uni2oem 来实现, 不过为了节省空间, 我们将该码表转换为: UNIGBK.BIN, 并存放到了外部 FLASH 芯片 (这部分实现请参考《STM32 开发指南》第 45 章 汉



字显示实验), 通过 `ff_convert` 函数, 我们可以实现 UNICODE 码和 GBK 码的互换, 不过都是十六进制格式的, 但是 ATK-SIM800C 模块接受的 UNICODE 编码, 都是采用字符串格式的形式, 所以需要做一些字符串/十六进制格式转换。

比如汉字“好”的 GBK 编码是 0XBAC3, 我们需要先将其转换为 UNICODE 编码: 0X597D, 然后再转换为 UNICODE 字符串“597D”, 最后再发送给 ATK-SIM800C 模块, 才可以正常使用。而相反的, 我们的程序在收到模块发过来的 UNICODE 字符串“597D”后, 必须先将其转换为 16 进制的 UNICODE 编码: 0X597D, 然后再将其转换为 GBK 编码: 0XBAC3, 最后送给汉字显示函数, 才能在 LCD 上面显示出“好”这个汉字。

### 1.2.4 GPRS 通信

ATK-SIM800C 模块内嵌了 TCP/IP 协议, 通过该模块, 我们可以很方便的进行 GPRS 数据通信。本文档例程我们将实现模块与电脑的 TCP 和 UDP 数据传输。将要用到的指令有: `AT+CGCLASS/AT+CGDCONT/ AT+CGATT/AT+CIPCSGP/AT+CIPHEAD /AT+CLPORT/AT+CIPSTART/AT+CIPSEN/AT+CIPSTATUS/AT+CIPCLOSE/AT+CIPSHUT` 等 11 条 AT 指令。

`AT+CGCLASS`, 用于设置移动台类别。SIM800C 模块支持类别“B”、“CG”和“CC”, 发送: `AT+CGCLASS="B"`, 设置移动台类别为 B。即, 模块支持包交换和电路交换模式, 但不能同时支持。

`AT+CGDCONT`, 用于设置 PDP 上下文。发送: `AT+CGDCONT=1,"IP","CMNET"`, 设置 PDP 上下文标志为 1, 采用互联网协议 (IP), 接入点为“CMNET”。

`AT+CGATT`, 用于设置附着和分离 GPRS 业务。发送: `AT+CGATT=1`, 附着 GPRS 业务。

`AT+CIPCSGP`, 用于设置 CSD 或 GPRS 链接模式。发送: `AT+CIPCSGP=1,"CMNET"`, 设置为 GPRS 连接, 接入点为“CMNET”。

`AT+ CIPHEAD`, 用于设置接收数据是否显示 IP 头。发送: `AT+CIPHEAD=1`, 即设置显示 IP 头, 在收到 TCP/UDP 数据的时候, 会在数据之前添加如: `+IPD:28`, 表示是 TCP/UDP 数据, 数据长度为 28 字节。通过这个头, 可以方便我们在程序上区分数据来源。

`AT+CLPORT`, 用于设置本地端口号。发送: `AT+CLPORT="TCP","8888"`, 即设置 TCP 连接本地端口号为 8888。

`AT+CIPSTART`, 用于建立 TCP 连接或注册 UDP 端口号。发送: `AT+CIPSTART="TCP","219.137.88.114","8086"`, 模块将建立一个 TCP 连接, 连接目标地址为: 219.137.88.114, 连接端口为 8086, 连接成功会返回: `CONNECT OK`。

`AT+CIPSEND`, 用于发送数据。在连接成功以后发送: `AT+CIPSEND`, 模块返回: `>`, 此时可以输入要发送的数据, 最大可以一次发送 1352 字节, 数据输入完后, 同发短信一样, 输入十六进制的: `1A (0X1A)`, 启动发送数据。在数据发送完成后, 模块返回: `SEND OK`, 表示发送成功。

`AT+CIPSTATUS`, 用于查询当前连接状态。发送: `AT+CIPSTATUS`, 模块即返回当前连接状态。

`AT+CIPCLOSE`, 用于关闭 TCP/UDP 连接。发送: `AT+CIPCLOSE=1`, 即可快速关闭当前 TCP/UDP 连接。

`AT+CIPSHUT`, 用于关闭移动场景。发送: `AT+SHUT`, 则可以关闭移动场景, 关闭场景后连接状态为: `IP INITIAL`, 可以通过发送: `AT+CIPSTATUS`, 查询。另外, 在连接建立后, 如果收到: `+PDP: DEACT`, 则必须发送: `AT+CIPSHUT`, 关闭场景后, 才能实现重连。

以上就是 GPRS 通信 (TCP/UDP) 将要用到的一些 AT 指令的简介, 这些指令的使用示例可以参考《ATK-SIM800C GSM(GPRS)模块用户手册\_V1.0》2.3.5 节。

另外, 要实现模块与电脑的 GPRS 通信, 需要**确保所用电脑具有公网 IP, 否则无法实现**

**通信**，推荐在 ADSL 网络下进行测试，并最好关闭防火墙/杀毒软件。

对于 ADSL 用户（没用路由器），直接拥有 1 个公网 IP，你可以通过百度，搜索：IP，第一个条目，就是本机 IP，如图 1.2.4.1 所示：



图 1.2.4.1 百度得到的本机公网 IP

该 IP 将与你的电脑 IP（双击本地连接图标→支持选项卡，即可查看）是一致的。

对与使用了路由器的 ADSL 用户，那么电脑 IP 与你百度到的公网 IP 是不一样的，如图 1.2.4.2 所示：



图 1.2.4.2 经过路由器后的电脑 IP

可以看到，我们电脑 IP 为 192.168.1.140，与公网 IP 不一致，此时我们需要对路由器进行一下转发规则设置：登录路由器控制页面，然后选择→LAN 接口配置→DMZ 配置，如图 1.2.4.3 所示：



图 1.2.4.3 DMZ 设置值

然后设置启用 DMZ 主机，并设置 DMZ 主机 IP 地址为所用电脑的 IP 地址，本机 IP 为：192.168.1.140，如图 1.2.4.4 所示：

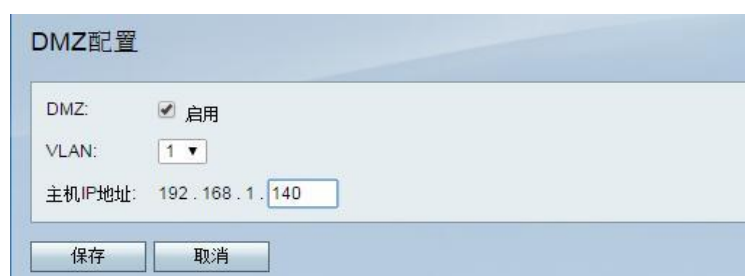


图 1.2.4.4 设置 DMZ 主机

然后保存。这样，我们就把内网 IP（192.168.1.140）映射到了外网，相当于经过路由器的电脑，拥有了一个公网 IP。

最后，我们在电脑上，还需要用到一个软件：网络调试助手，来协助验证 GPRS 通信，该软件启动界面如图 1.2.4.5 所示：



图 1.2.4.5 网络调试助手启动界面



该软件的使用非常简单，我们将在第四节配合我们的例程向大家介绍该软件的使用。

### 1.2.5 蓝牙通信

ATK-SIM800C 模块集成了蓝牙 3.0，通过该模块，我们可以很方便的进行蓝牙数据通信。本文档例程我们将实现模块与手机蓝牙数据传输。将要用到的指令有：AT+BTPOWER/AT+BTHOST/AT+BTSCAN/AT+BTUNPAIR/AT+BTPAIR/AT+BTACPT/AT+BTSPSEND/AT+BTDISCONN 等 8 条 AT 指令。

AT+BTPOWER,用于设置开启或关闭蓝牙电源，当发送 AT+BTPOWER=1，返回 OK，表示开启蓝牙电源；发送 AT+BTPOWER=0,返回 OK，表示关闭蓝牙电源。

AT+BTHOST，用于查询和设置当前模块蓝牙设备名，当发送 AT+BTHOST? 时，返回该设备的蓝牙名字和地址，设置当前模块蓝牙设备名时，命令格式为 AT+BTHOST=<name>,name 为你要设置的设备名。

AT+BTSCAN,用于设置蓝牙搜索参数，发送 AT+BTSCAN=1,10，开启扫描设备，时间为 10s，搜索到设备返回例如：+BTSCAN: 0,1,"Meizu MX4 Pro",22:22:5f:b8:e9:af,-79，表示设备 1，名称：Meizu MX4 Pro，地址：22:22:5f:b8:e9:af，信号：-79。

AT+BTUNPAIR 用于删除蓝牙设备配对信息，发送 AT+BTUNPAIR=0，删除所有已配对的蓝牙设备信息。（**注意：上次配对过的设备，下次进行配对前必须删除配对信息**）

AT+BTPAIR 用于实现蓝牙配对，发送:AT+BTPAIR=0,1，向设备 1 发起配对请求。

AT+BTACPT 用于接收配对的蓝牙设备的连接请求，发送 AT+BTACPT=1，接收连接请求，发送 AT+BTACPT=0，拒绝连接请求。

AT+BTSPSEND 用于蓝牙串口发送数据，发送数据有两种方式，定长与非定长。在连接成功以后发送：AT+CIPSEND，模块返回：>，即非定长模式下，此时可以输入要发送的数据，最大可以一次发送 1024 字节，数据输入完后，同发短信一样，输入十六进制的：1A(0X1A)，启动发送数据。在数据发送完成后，模块返回：SEND OK，表示发送成功。**关于定长模式，请参考文档《ATK-SIM800C 蓝牙功能\_AN1603C.pdf》，文件路径：ATK-SIM800C 模块资料\ATK-SIM800C 蓝牙功能\_AN1603C.pdf。**

AT+BTDISCONN 用于断开已连接的蓝牙设备服务，发送 AT+BTDISCONN=1，断开与设备 1 服务的连接。

以上就是蓝牙通信将要用到的一些 AT 指令的简介，这些指令的使用示例可以参考《ATK-SIM800C 蓝牙功能\_AN1603C.pdf》或《ATK-SIM800C GSM(GPRS)模块用户手册\_V1.0》2.3.6 节。

另外，要实现模块与手机的蓝牙通信，需要在手机端安装一个软件：蓝牙串口助手增强版\_Bluetooth\_spp\_pro,该软件你可以在豌豆荚市场搜索到，也可以在我们提供的 ATK-SIM800C 配套软件资料中找到。该软件启动界面如图 1.2.5.1 所示：

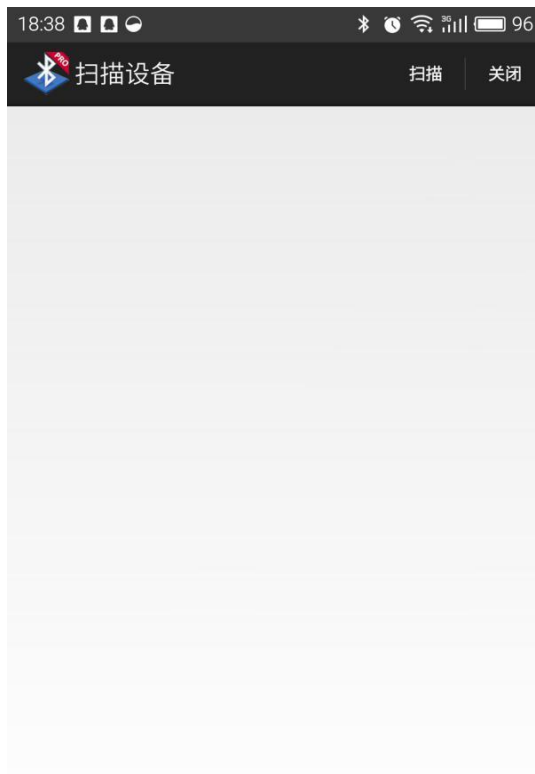


图 1.2.5.1 蓝牙串口助手增强版启动界面

该软件的使用非常简单，我们将在第四节配合我们的例程向大家介绍该软件的使用。

### 1.2.6 TTS 文本转语音

关于 TTS 文本转语音的使用说明请查看《[ATK-SIM800C TTS 功能\\_AN1603B.pdf](#)》，文件路径：[ATK-SIM800C 模块资料\ATK-SIM800C TTS 功能\\_AN1603B.pdf](#)，在这里我们就不做出介绍了。

## 2、硬件连接

本实验功能简介：本实验用于测试 ATK-SIM800C GSM/GPRS 模块，总共包括四大项测试：

1，拨号测试—通过按 KEY0 按键进入此项测试。进入测试后，屏幕将虚拟一个键盘，通过键盘输入电话号码，即可进行拨号。如果有电话打进来，则会显示来电号码，并可以通过键盘实现来电接听。

2，短信测试—通过按 KEY1 按键进入此项测试。此项测试包含 2 个子项：读短信测试和发短信测试。按 KEY0 进入读短信测试，屏幕将显示 SIM 卡当前存储的信息条数以及总共可以存储的信息条数，并在屏幕上虚拟一个键盘，通过键盘输入，即可读取指定条目的短信，并且语音报读，其内容将显示在 LCD 上面。按 KEY1 进入发短信测试，屏幕将显示一条固定的短信内容，并虚拟一个键盘，通过键盘输入目标手机号码，即可执行发送，将固定内容的短信发送给目标手机，并带状态提示。

3，GPRS 测试—通过按 WK\_UP 按键进入此项测试。此项测试又包含 2 个子项：TCP 测试和 UDP 测试。默认为 TCP 连接，通过按 WK\_UP 按键，可以在 TCP/UDP 之间切换。此项测试需要输入 IP 地址（要连接的目标 IP 地址，必须为公网 IP），端口号固定为：8086。在设定好连接方式和 IP 地址之后，即可进行连接，连接成功后，则可以和目标进行 GPRS 数据通

信。本测试，我们在电脑和 ATK-SIM800C 模块之间实验，电脑端需要一个软件：网络调试助手，来实现和模块的 TCP/UDP 数据通信测试。

4，蓝牙测试—通过先按 KEY\_2 按键，然后按 KEY0 进入此项测试。此项测试又包含 2 个子项：发起配对请求和接收配对请求模式的通信，按 KEY2 进入发起配对请求，然后通过扫描搜索到手机设备，建立连接后，手机端打开蓝牙调试助手与模块再一次进行 spp 连接，然后手机看到模块发送的数据，屏幕也显示手机端发送过来的数据。按 KEY0 进入接收配对请求模式，手机端连接搜索到模块设备，然后进行连接，spp 的连接和数据通信和前面的效果一样。

本实验所需的硬件资源如下：

- 1，ALIENTEK 阿波罗 STM32F429 开发板 1 个
- 2，ATK-SIM800C GSM/GPRS 模块一个
- 3，直流稳压电源 1 个（推荐 12V 1A 电源）
- 4，中国移动/联通 GSM SIM 卡一张（未停机，开通 GPRS 业务）
- 5，耳机一副（带麦克风功能，用于通话测试）
- 6，一台支持蓝牙的手机设备（安卓系统）

要完成本文档例程的所有功能测试，请大家务必准备好以上硬件，否则有些功能可能无法完成。

ATK-SIM800C 所有的控制与数据，都是通过串口来传输的，所以我们的开发板与模块连接，只需要连接串口即可（当然还需要共地）。接下来，我们看看 ALIENTEK 阿波罗 STM32F429 开发板与 ATK-SIM800C 模块的连接方式，本例程通过开发板的串口 3 连接 ATK-SIM800C 模块，有两种连接方式：

1，通过杜邦线连接。

这种方式通过杜邦线连接，需要将阿波罗开发板 P9 上面的两个跳线帽拔了，同时将 ATK-SIM800C 模块 P3 的两个跳线帽拔了。然后，用 3 根杜邦线，按表 2.1 所示关系连接：

ATK-SIM800C GSM 模块与开发板连接关系			
ATK-SIM800C 模块	GND	STXD	SRXD
阿波罗 STM32F429 开发板	GND	PB11(RX)	PB10(TX)

表 2.1 ATK-SIM800C 模块和阿波罗 STM32F429 开发板连接关系表

注意，表中的 GND，大家可以在开发板和 ATK-SIM800C 模块上面，随便找一个 GND 标号的排针，连接在一起即可。

2，通过 RS232 串口线连接。

这种方式比较简单，因为阿波罗 STM32F429 开发板自带了一个公头 RS232 接口(COM3)并且配备了一根 RS232 串口线，所以，可以直接用 RS232 串口线，将开发板和 GSM 模块连接起来即可，如图 2.1 所示：

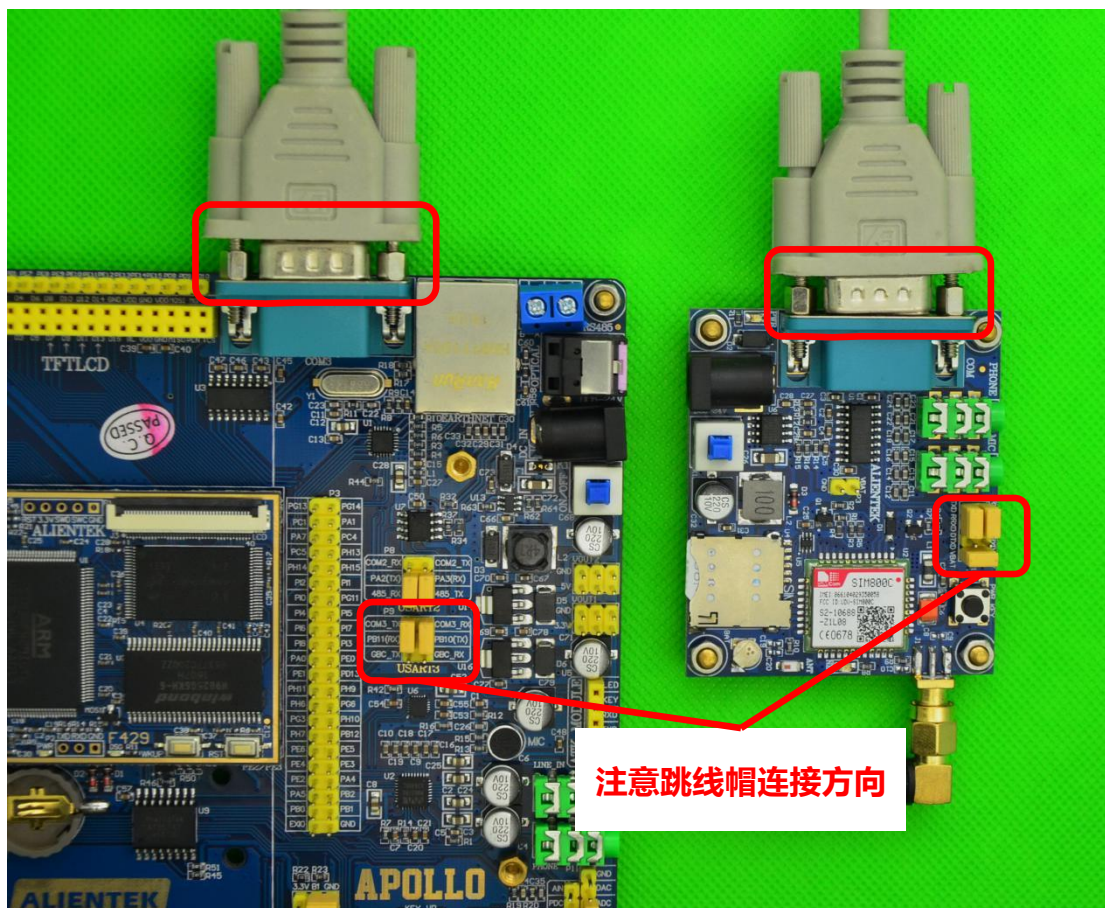


图 2.1 ATK-SIM800C 模块与阿波罗 STM32F429 开发板连接

如图 2.1 所示，通过 RS232 串口线连接的时候，必须保证 P9 和 P3 的跳线帽都在位，不能拔了，也不能插错方向!! 跳线帽插法如图所示。

我们推荐采用第二种方法(图 2.1 所示)来连接阿波罗 STM32F429 开发板和 ATK-SIM800C 模块，最后，特别提醒：ATK-SIM800C 模块必须由单独的电源供电（推荐 12V1A 电源），开发板则可以通过 USB 插电脑供电，不过切记要共地哦!!

### 3、软件实现

本实验，在阿波罗 STM32F429 开发板的汉字显示实验基础上进行修改，在 HARDWARE 文件夹里面新建 USART3 文件夹，存放 usart3.c 和 usart3.h 两个文件。并在工程 HARDWARE 组里面添加 usart3.c，并添加 USART3 文件夹到头文件包含路径。

在工程目录添加 SIM800C 文件夹，并在工程里面再添加 SIM800C 分组，新建 sim800c.c 和 sim800c.h 两个文件，存放在 SIM800C 文件夹内，将 sim800c.c 加入 SIM800C 分组，并添加 SIM800C 文件夹到头文件包含路径。

我们去掉原工程的一些未用到的.c 文件，最终的工程如图 3.1 所示：

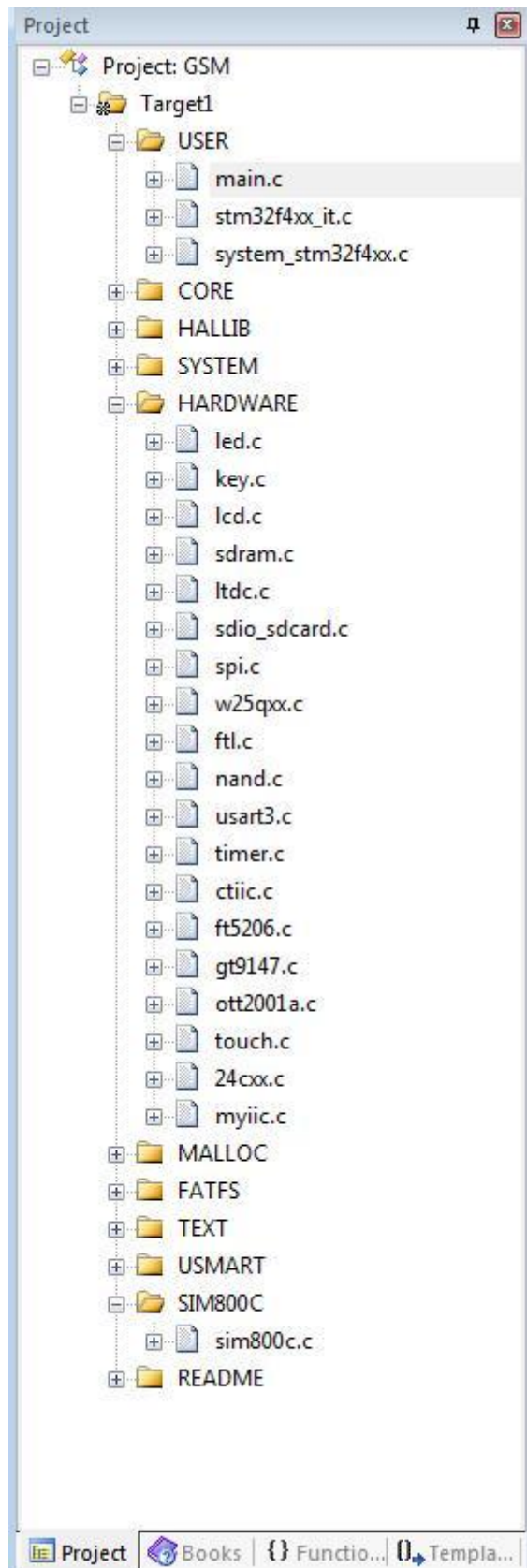


图 3.1 ATK-SIM800C 模块测试实验工程截图

usart3.c 在之前的例程（详见：AN1610 ATK-HC05 蓝牙串口模块使用说明）已经有介绍过，这里，我们主要看 sim800c.c 和 main.c 的代码，首先是 sim800c.c，该文件是 ATK-SIM800C



模块的驱动代码，sim800c.c 里面的代码如下：

```
////////////////////////////////////
u8 Scan_Wtime = 0;//保存蓝牙扫描需要的时间
u8 BT_Scan_mode=0;//蓝牙扫描设备模式标志

//usmart 支持部分
//将收到的 AT 指令应答数据返回给电脑串口
//mode:0,不清零 USART3_RX_STA;
//      1,清零 USART3_RX_STA;
void sim_at_response(u8 mode)
{
    if(USART3_RX_STA&0X8000)          //接收到一次数据
    {
        USART3_RX_BUF[USART3_RX_STA&0X7FFF]=0;//添加结束符
        printf("%s",USART3_RX_BUF);    //发送到串口
        if(mode)USART3_RX_STA=0;
    }
}
////////////////////////////////////
//ATK-SIM800C 各项测试(拨号测试、短信测试、GPRS 测试、蓝牙测试)共用代码
//SIM800C 发送命令后,检测接收到的应答
//str:期待的应答结果
//返回值:0,没有得到期待的应答结果
//      其他,期待应答结果的位置(str 的位置)
u8* sim800c_check_cmd(u8 *str)
{
    char *strx=0;
    if(USART3_RX_STA&0X8000) //接收到一次数据了
    {
        USART3_RX_BUF[USART3_RX_STA&0X7FFF]=0;//添加结束符
        strx=strstr((const char*)USART3_RX_BUF,(const char*)str);
    }
    return (u8*)strx;
}
//向 SIM800C 发送命令
//cmd:发送的命令字符串(不需要添加回车了),当 cmd<0XFF 的时候,发送数字(比如发送
//0X1A),大于的时候发送字符串.
//ack:期待的应答结果,如果为空,则表示不需要等待应答
//waittime:等待时间(单位:10ms)
//返回值:0,发送成功(得到了期待的应答结果)
//      1,发送失败
u8 sim800c_send_cmd(u8 *cmd,u8 *ack,u16 waittime)
{
    u8 res=0;
```

```

    USART3_RX_STA=0;
    if((u32)cmd<=0xFF)
    {
        while((USART3->SR&0X40)==0);//等待上一次数据发送完成
        USART3->DR=(u32)cmd;
    }else u3_printf("%s\r\n",cmd);//发送命令
    if(waittime==1100)//11s 后返回读口数据（蓝牙扫描模式）
    {
        Scan_Wtime = 11;//需要定时的时间
        TIM7_SetARR(10000-1);//产生 1S 定时中断
    }
    if(ack&&waittime)    //需要等待应答
    {
        while(--waittime) //等待倒计时
        {
            if(BT_Scan_mode)//蓝牙扫描模式
            {
                res=KEY_Scan(0);//返回上一级
                if(res==KEY1) return 2;
            }
            delay_ms(10);
            if(USART3_RX_STA&0X8000)//接收到期待的应答结果
            {
                if(sim800c_check_cmd(ack))break;//得到有效数据
                USART3_RX_STA=0;
            }
        }
        if(waittime==0)res=1;
    }
    return res;
}
//接收 SIM800C 返回数据（蓝牙测试模式下使用）
//request:期待接收命令字符串
//waitting:等待时间(单位： 10ms)
//返回值:0,发送失败(得到了期待的应答结果)
//      1,发送失败
u8 sim800c_wait_request(u8 *request ,u16 waittime)
{
    u8 res = 1;
    u8 key;
    if(request && waittime)
    {
        while(--waittime)
        {

```

```

        key=KEY_Scan(0);
        if(key==KEY1) return 2;//返回上一级
        delay_ms(10);
        if(USART3_RX_STA &0x8000)//接收到期待的应答结果
        {
            if(sim800c_check_cmd(request)) break;//得到有效数据
            USART3_RX_STA=0;
        }
    }
    if(waittime==0)res=0;
}
return res;
}
//将 1 个字符转换为 16 进制数字
//chr:字符,0~9/A~F/a~f
//返回值:chr 对应的 16 进制数值
u8 sim800c_chr2hex(u8 chr)
{
    if(chr>='0'&&chr<='9')return chr-'0';
    if(chr>='A'&&chr<='F')return (chr-'A'+10);
    if(chr>='a'&&chr<='f')return (chr-'a'+10);
    return 0;
}
//将 1 个 16 进制数字转换为字符
//hex:16 进制数字,0~15;
//返回值:字符
u8 sim800c_hex2chr(u8 hex)
{
    if(hex<=9)return hex+'0';
    if(hex>=10&&hex<=15)return (hex-10+'A');
    return '0';
}
//unicode gbk 转换函数
//src:输入字符串
//dst:输出(uni2gbk 时为 gbk 内码,gbk2uni 时,为 unicode 字符串)
//mode:0,unicode 到 gbk 转换;
//      1,gbk 到 unicode 转换;
void sim800c_unigbk_exchange(u8 *src,u8 *dst,u8 mode)
{
    u16 temp;
    u8 buf[2];
    if(mode)//gbk 2 unicode
    {
        while(*src!=0)

```

```

    {
        if(*src<0X81) //非汉字
        {
            temp=(u16)ff_convert((WCHAR)*src,1);
            src++;
        }else //汉字,占 2 个字节
        {
            buf[1]=*src++;
            buf[0]=*src++;
            temp=(u16)ff_convert((WCHAR)*(u16*)buf,1);
        }
        *dst++=sim800c_hex2chr((temp>>12)&0X0F);
        *dst++=sim800c_hex2chr((temp>>8)&0X0F);
        *dst++=sim800c_hex2chr((temp>>4)&0X0F);
        *dst++=sim800c_hex2chr(temp&0X0F);
    }
}else//unicode 2 gbk
{
    while(*src!=0)
    {
        buf[1]=sim800c_chr2hex(*src++)*16;
        buf[1]+=sim800c_chr2hex(*src++);
        buf[0]=sim800c_chr2hex(*src++)*16;
        buf[0]+=sim800c_chr2hex(*src++);
        temp=(u16)ff_convert((WCHAR)*(u16*)buf,0);
        if(temp<0X80){*dst=temp;dst++;}
        else {(u16*)dst=swap16(temp);dst+=2;}
    }
}
*dst=0;//添加结束符
}
//键盘码表
const u8* kbd_tbl1[13]={"1","2","3","4","5","6","7","8","9","*","0","#","DEL"};
const u8* kbd_tbl2[13]={"1","2","3","4","5","6","7","8","9",".","0","#","DEL"};
u8** kbd_tbl;
u8* kbd_fn_tbl[2];
//加载键盘界面（尺寸为 240*140）
//x,y:界面起始坐标（320*240 分辨率的时候，x 必须为 0）
void sim800c_load_keyboard(u16 x,u16 y,u8 **kbtbl)
{
    u16 i;
    POINT_COLOR=RED;
    kbd_tbl=kbtbl;
    LCD_Fill(x,y,x+240,y+140,WHITE);

```

```

LCD_DrawRectangle(x,y,x+240,y+140);
LCD_DrawRectangle(x+80,y,x+160,y+140);
LCD_DrawRectangle(x,y+28,x+240,y+56);
LCD_DrawRectangle(x,y+84,x+240,y+112);
POINT_COLOR=BLUE;
for(i=0;i<15;i++)
{
    if(i<13)Show_Str_Mid(x+(i%3)*80,y+6+28*(i/3),(u8*)kbd_tbl[i],16,80);
    else Show_Str_Mid(x+(i%3)*80,y+6+28*(i/3),kbd_fn_tbl[i-13],16,80);
}
}
//按键状态设置
//x,y:键盘坐标
//key:键值（0~8）
//sta:状态，0，松开；1，按下；
void sim800c_key_staset(u16 x,u16 y,u8 keyx,u8 sta)
{
    u16 i=keyx/3,j=keyx%3;
    if(keyx>15)return;
    if(sta)LCD_Fill(x+j*80+1,y+i*28+1,x+j*80+78,y+i*28+26,GREEN);
    else LCD_Fill(x+j*80+1,y+i*28+1,x+j*80+78,y+i*28+26,WHITE);
    if(j&&(i>3))Show_Str_Mid(x+j*80,y+6+28*i,(u8*)kbd_fn_tbl[keyx-13],16,80);
    else Show_Str_Mid(x+j*80,y+6+28*i,(u8*)kbd_tbl[keyx],16,80);
}
//得到触摸屏的输入
//x,y:键盘坐标
//返回值：按键键值（1~15 有效；0,无效）
u8 sim800c_get_keynum(u16 x,u16 y)
{
    u16 i,j;
    static u8 key_x=0;//0,没有任何按键按下；1~15，1~15 号按键按下
    u8 key=0;
    tp_dev.scan(0);
    if(tp_dev.sta&TP_PRES_DOWN) //触摸屏被按下
    {
        for(i=0;i<5;i++)
        {
            for(j=0;j<3;j++)
            {
                if(tp_dev.x[0]<(x+j*80+80)&&tp_dev.x[0]>(x+j*80)&&tp_dev.y[0]
                <(y+i*28+28)&&tp_dev.y[0]>(y+i*28))
                {
                    key=i*3+j+1;
                    break;
                }
            }
        }
    }
}

```





```

if(sim800c_send_cmd("AT+CTTSPARAM=20,0,50,70,0","OK",200))return 1;
//设置 TTS 声音大小、语调配置
if(sim800c_send_cmd("AT+CLIP=1","OK",200))return 1;    //设置来电显示
if(sim800c_send_cmd("AT+COLP=1","OK",200))return 2;    //设置被叫号码显示
p1=mymalloc(SRAMIN,20); //申请 20 直接用于存放号码
if(p1==NULL)return 2;
POINT_COLOR=RED;
Show_Str_Mid(0,30,"ATK-SIM800C 拨号测试",16,240);
Show_Str(40,70,200,16,"请拨号:",16,0);
kbd_fn_tbl[0]="拨号";
kbd_fn_tbl[1]="返回";
sim800c_load_keyboard(0,180,(u8**)kbd_tbl1);
POINT_COLOR=BLUE;
while(1)
{
    delay_ms(10);
    if(USART3_RX_STA&0X8000)    //接收到数据
    {
        sim_at_response(0);
        if(cmode==1 | cmode==2)
        {
            if(cmode==1)if(sim800c_check_cmd("+COLP:"))cmode=2;//拨号成功
            if(sim800c_check_cmd("NO CARRIER"))cmode=0;    //拨号失败
            if(sim800c_check_cmd("NO ANSWER"))cmode=0;    //拨号失败
            if(sim800c_check_cmd("ERROR"))cmode=0;    //拨号失败
        }
        if(sim800c_check_cmd("+CLIP:"))//接收到来电
        {
            cmode=3;
            p=sim800c_check_cmd("+CLIP:");
            p+=8;
            p2=(u8*)strstr((const char *)p,"");
            p2[0]=0;//添加结束符
            strcpy((char*)p1,(char*)p);
        }
        USART3_RX_STA=0;
    }
    key=sim800c_get_keynum(0,180);
    if(key)
    {
        if(key<13)
        {
            if(cmode==0&&pohnumlen<15)
            {

```

```

        callbuf[pohnenumlen++]=kbd_tbl[key-1][0];
        u3_printf("AT+CLDTMF=2,\"%c\"\\r\\n",kbd_tbl[key-1][0]);
        delay_ms(55);//延时
        u3_printf("AT+CTTS=2,\"%c\"\\r\\n",kbd_tbl[key-1][0]); //TTS 语音

    }else if(cmode==2)//通话中
    {
        u3_printf("AT+CLDTMF=2,\"%c\"\\r\\n",kbd_tbl[key-1][0]);
        delay_ms(100);
        u3_printf("AT+VTS=%c\\r\\n",kbd_tbl[key-1][0]);
        LCD_ShowChar(40+56,90,kbd_tbl[key-1][0],16,0);
    }
}
else
{
    if(key==13)if(pohnenumlen&&cmode==0)pohnenumlen--;//删除
    if(key==14)//执行拨号
    {
        if(cmode==0)//拨号模式
        {
            callbuf[pohnenumlen]=0;        //最后加入结束符
            delay_ms(200);
            u3_printf("ATD%s;\\r\\n",callbuf);//拨号
            cmode=1;                        //拨号中模式
        }else
        {
            sim800c_send_cmd("ATH","OK",200);//挂机
            cmode=0;
        }
    }
    if(key==15)
    {
        if(cmode==3//接收到来电
        {
            sim800c_send_cmd("ATA","OK",200);//发送应答指令
            Show_Str(40+56,70,200,16,callbuf,16,0);
            cmode=2;
        }else
        {
            sim800c_send_cmd("ATH",0,0);//结束通话
            break;//退出循环
        }
    }
}
}
if(cmode==0)//只有在等待拨号模式有效

```

```

        {
            callbuf[pohnumlen]=0;
            LCD_Fill(40+56,70,239,70+16,WHITE);
            Show_Str(40+56,70,200,16,callbuf,16,0);
        }
    }
    if(oldmode!=cmode)//模式变化了
    {
        switch(cmode)
        {
            case 0:
                kbd_fn_tbl[0]="拨号";
                kbd_fn_tbl[1]="返回";
                POINT_COLOR=RED;
                Show_Str(40,70,200,16,"请拨号:",16,0);
                LCD_Fill(40+56,70,239,70+16,WHITE);
                if(pohnumlen)
                {
                    POINT_COLOR=BLUE;
                    Show_Str(40+56,70,200,16,callbuf,16,0);
                }
                break;
            case 1:
                POINT_COLOR=RED;
                Show_Str(40,70,200,16,"拨号中:",16,0);
                pohnumlen=0;
            case 2:
                POINT_COLOR=RED;
                if(cmode==2)Show_Str(40,70,200,16,"通话中:",16,0);
                kbd_fn_tbl[0]="挂断";
                kbd_fn_tbl[1]="返回";
                break;
            case 3:
                POINT_COLOR=RED;
                Show_Str(40,70,200,16,"ÐÀ´ µç:",16,0);
                POINT_COLOR=BLUE;
                Show_Str(40+56,70,200,16,p1,16,0);
                kbd_fn_tbl[0]="Ò¶Ï";
                kbd_fn_tbl[1]="%ÓÏý";
                break;
        }
        if(cmode==2)Show_Str(40,90,200,16,"DTMF 音:",16,0);
        //通话中,可以通过键盘输入 DTMF 音
        else LCD_Fill(40,90,120,90+16,WHITE);
    }

```

```

        sim800c_load_keyboard(0,180,(u8**)kbd_tbl1); //显示键盘
        oldmode=cmode;
    }
    if((lenx%50)==0)LED0=!LED0;
    lenx++;
}
myfree(SRAMIN,p1);
return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//短信测试部分代码
//SIM800C 读短信测试
void sim800c_sms_read_test(void)
{
    u8 *p,*p1,*p2;
    u8 timex=0;
    u8 msgindex[3];
    u8 msglen=0;
    u8 msgmaxnum=0;    //短信最大条数
    u8 key=0;
    u8 smsreadsta=0; //是否在短信显示状态
    p=mymalloc(SRAMIN,200);//申请 200 个字节的内存
    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    Show_Str_Mid(0,30,"ATK-SIM800C 读短信测试",16,240);
    Show_Str(30,50,200,16,"读取:    总信息:",16,0);
    kbd_fn_tbl[0]="读取";
    kbd_fn_tbl[1]="返回";
    sim800c_load_keyboard(0,180,(u8**)kbd_tbl1);//显示键盘
    while(1)
    {
        key=sim800c_get_keynum(0,180);
        if(key)
        {
            if(smsreadsta)
            {
                LCD_Fill(30,75,239,179,WHITE);//清除显示的短信内容
                u3_printf("AT+CTTS=0\r\n");
                smsreadsta=0;
            }
            if(key<10 || key==11)
            {
                if(msglen<2)
                {

```



```

        msgindex[msglen++]=kbd_tbl[key-1][0];
        u3_printf("AT+CLDTMF=2,\"%c\"\\r\\n",kbd_tbl[key-1][0]);
    }
    if(msglen==2)
    {
        key=(msgindex[0]-'0')*10+msgindex[1]-'0';
        if(key>msgmaxnum)
        {
            msgindex[0]=msgmaxnum/10+'0';
            msgindex[1]=msgmaxnum%10+'0';
        }
    }
}
else
{
    if(key==10 || key==12) u3_printf("AT+CTTS=0\\r\\n");
    if(key==13)
    {u3_printf("AT+CTTS=0\\r\\n");if(msglen)msglen--;}//删除
    if(key==14&&msglen)//执行读取短信
    {
        LCD_Fill(30,75,239,179,WHITE);//清除之前的显示
        sprintf((char*)p,"AT+CMGR=%s",msgindex);
        if(sim800c_send_cmd(p,"+CMGR:",200)==0)//读取短信
        {
            POINT_COLOR=RED;
            Show_Str(30,75,200,12,"状态:",12,0);
            Show_Str(30+75,75,200,12,"来自:",12,0);
            Show_Str(30,90,200,12,"接收时间:",12,0);
            Show_Str(30,105,200,12,"内容:",12,0);
            POINT_COLOR=BLUE;
            if(strstr((constchar*)(USART3_RX_BUF),"UNREAD")==0)
            Show_Str(30+30,75,200,12,"已读",12,0);
            else Show_Str(30+30,75,200,12,"未读",12,0);
            p1=(u8*)strstr((const char*)(USART3_RX_BUF),",");
            p2=(u8*)strstr((const char*)(p1+2),"\");
            p2[0]=0;//加入结束符
            sim800c_unigbk_exchange(p1+2,p,0);//unicode 转为 gbk 码
            Show_Str(30+75+30,75,200,12,p,12,0);    //显示电话号码
            p1=(u8*)strstr((const char*)(p2+1),"/");
            p2=(u8*)strstr((const char*)(p1),"+");
            p2[0]=0;//加入结束符
            Show_Str(30+54,90,200,12,p1-2,12,0);    //显示接收时间
            p1=(u8*)strstr((const char*)(p2+1),"\r");//寻找回车符
            sim800c_unigbk_exchange(p1+2,p,0);//unicode 转为 gbk 码
            Show_Str(30+30,105,180,75,p,12,0);//显示短信内容
        }
    }
}

```

```

        u3_printf("AT+CTTS=2,\"%s\"\\r\\n",p);
        //TTS 读取短信(ASCII 模式)
        smsreadsta=1;    //标记有显示短信内容
    }else
    {
        Show_Str(30,75,200,12,"无短信内容!!!请检查!!",12,0);
        u3_printf("AT+CTTS=2,\"无短信内容请检查\"\\r\\n");
        delay_ms(1000);
        delay_ms(1000);
        LCD_Fill(30,75,239,75+12,WHITE);//清除显示
    }
    USART3_RX_STA=0;
}
if(key==15){u3_printf("AT+CTTS=0\\r\\n"); break;}
}
msgindex[msglen]=0;
LCD_Fill(30+40,50,86,50+16,WHITE);
Show_Str(30+40,50,86,16,msgindex,16,0);
}
if(timex==0)    //2.5 秒左右更新一次
{
    if(sim800c_send_cmd("AT+CPMS?","+CPMS:",200)==0)
        //查询优选消息存储器
    {
        p1=(u8*)strstr((const char*)(USART3_RX_BUF),",");
        p2=(u8*)strstr((const char*)(p1+1),",");
        p2[0]='/';
        if(p2[3]==',')//小于 64K SIM 卡，最多存储几十条短信
        {
            msgmaxnum=(p2[1]-'0')*10+p2[2]-'0';//获取最大存储短信条数
            p2[3]=0;
        }else //如果是 64K SIM 卡，则能存储 100 条以上的信息
        {
            msgmaxnum=(p2[1]-'0')*100+(p2[2]-'0')*10+p2[3]-'0';//最大条数
            p2[4]=0;
        }
        sprintf((char*)p,"%s",p1+1);
        Show_Str(30+17*8,50,200,16,p,16,0);
        USART3_RX_STA=0;
    }
}
if((timex%20)==0)LED0=!LED0;//200msÉÄË,
timex++;
delay_ms(10);

```

```

        if(USART3_RX_STA&0X8000)sim_at_response(1);//检查从模块接收到的数据
    }
    myfree(SRAMIN,p);
}
//测试短信发送内容(70 个字[UCS2 的时候,1 个字符/数字都算 1 个字])
const u8* sim800c_test_msg="您好，这是一条测试短信，由 ATK-SIM800C GSM 模块发送，模块购买地址:http://eboard.taobao.com，谢谢支持！";
//SIM800C 发短信测试
void sim800c_sms_send_test(void)
{
    u8 *p,*p1,*p2;
    u8 phonebuf[20];        //号码缓存
    u8 pohnenumlen=0;       //号码长度,最大 15 个数
    u8 timex=0;
    u8 key=0;
    u8 smssendsta=0;        //短信发送状态,0,等待发送;1,发送失败;2,发送成功
    p=mymalloc(SRAMIN,100); //申请内存,用于存放电话号码的 unicode 字符串
    p1=mymalloc(SRAMIN,300);//申请内存,用于存放短信的 unicode 字符串
    p2=mymalloc(SRAMIN,100);//申请 100 个字节的内存 存放：AT+CMGS=p1
    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    Show_Str_Mid(0,30,"ATK-SIM800C 发短信测试,16,240);
    Show_Str(30,50,200,16,"发送给:",16,0);
    Show_Str(30,70,200,16,"状态:",16,0);
    Show_Str(30,90,200,16,"内容:",16,0);
    POINT_COLOR=BLUE;
    Show_Str(30+40,70,170,90,"等待发送",16,0);//显示状态
    Show_Str(30+40,90,170,90,(u8*)sim800c_test_msg,16,0);//显示短信内容
    kbd_fn_tbl[0]="发送";
    kbd_fn_tbl[1]="返回";
    sim800c_load_keyboard(0,180,(u8**)kbd_tbl1);//显示键盘
    while(1)
    {
        key=sim800c_get_keynum(0,180);
        if(key)
        {
            if(smssendsta)
            {
                smssendsta=0;
                Show_Str(30+40,70,170,90,"等待发送",16,0);//显示状态
            }
            if(key<10 || key==11)
            {
                if(pohnenumlen<15)

```

```

        {
            phonebuf[pohnenumlen++]=kbd_tbl[key-1][0];
            u3_printf("AT+CLDTMF=2,\"%c\"\\r\\n",kbd_tbl[key-1][0]);
        }
    }else
    {
        if(key==13){if(pohnenumlen)pohnenumlen--;//删除
        if(key==14&&pohnenumlen)                //执行发送短信
        {
            Show_Str(30+40,70,170,90,"正在发送",16,0);//显示正在发送
            smssendsta=1;
            sim800c_unigbk_exchange(phonebuf,p,1);//号码转 unicode 字符串
            sim800c_unigbk_exchange((u8*)sim800c_test_msg,p1,1);
            //将短信内容转换为 unicode 字符串.
            sprintf((char*)p2,"AT+CMGS=\"%s\"",p);
            if(sim800c_send_cmd(p2,">",200)==0) //发送短信命令+电话号码
            {
                u3_printf("%s",p1); //发送短信内容到 GSM 模块
                if(sim800c_send_cmd((u8*)0X1A,"+CMGS:",1000)==0)
                smssendsta=2;
                //发送结束符,等待发送完成(最长等待 10 秒钟,因为短信长
                //了的话,等待时间会长一些)
            }
            if(smssendsta==1)Show_Str(30+40,70,170,90,"发送失败",16,0);
            else Show_Str(30+40,70,170,90,"发送成功",16,0);//显示状态
            USART3_RX_STA=0;
        }
        if(key==15)break;
    }
    phonebuf[pohnenumlen]=0;
    LCD_Fill(30+54,50,239,50+16,WHITE);
    Show_Str(30+54,50,156,16,phonebuf,16,0);
}
if((timex%20)==0)LED0=!LED0;//200msÉÄË,
timex++;
delay_ms(10);
if(USART3_RX_STA&0X8000)sim_at_response(1);//检查 GSM 接收到的数据
}
myfree(SRAMIN,p);
myfree(SRAMIN,p1);
myfree(SRAMIN,p2);
}
//sms 测试主界面
void sim800c_sms_ui(u16 x,u16 y)

```

```

{
    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    Show_Str_Mid(0,y,"ATK-SIM800C 短信测试","",16,240);
    Show_Str(x,y+40,200,16,"请选择:",16,0);
    Show_Str(x,y+60,200,16,"KEY0:读短信测试",16,0);
    Show_Str(x,y+80,200,16,"KEY1:发短信测试",16,0);
    Show_Str(x,y+100,200,16,"KEY_UP:返回上级菜单",16,0);
}

//SIM800C 短信测试
//用于读短信或者发短信
//返回值:0,正常
//其他,错误代码
u8 sim800c_sms_test(void)
{
    u8 key;
    u8 timex=0;
    if(sim800c_send_cmd("AT+CMGF=1","OK",200))return 1;
    //设置文本模式
    if(sim800c_send_cmd("AT+CSCS=\"UCS2\"", "OK",200))return 2;
    //设置 TE 字符集为 UCS2
    if(sim800c_send_cmd("AT+CSMP=17,0,2,25","OK",200))return 3;
    //设置短消息文本模式参数
    if(sim800c_send_cmd("AT+CTTSPARAM=5,0,51,61,0","OK",200))return 1;
    //设置 TTS 声音大小、语调配置
    sim800c_sms_ui(40,30);
    while(1)
    {
        key=KEY_Scan(0);
        if(key==KEY0_PRES)
        {
            sim800c_sms_read_test();
            sim800c_sms_ui(40,30);
            timex=0;
        }else if(key==KEY1_PRES)
        {
            sim800c_sms_send_test();
            sim800c_sms_ui(40,30);
            timex=0;
        }else if(key==WKUP_PRES)break;
        timex++;
        if(timex==20)
        {
            timex=0;
        }
    }
}

```



```

        LED0=!LED0;
    }
    delay_ms(10);
    sim_at_response(1);//检查 GSM 模块发送过来的数据,及时上传给电脑
}

    sim800c_send_cmd("AT+CSCS=\"GSM\"", "OK", 200);
    //设置默认的 GSM 7 位缺省字符集
    return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//GPRS 测试部分代码
const u8 *modetbl[2]={"TCP","UDP"};//连接模式
//tcp/udp 测试
//带心跳功能,以维持连接
//mode:0:TCP 测试;1,UDP 测试)
//ipaddr:ip 地址
//port:端口
void sim800c_tcpudp_test(u8 mode,u8* ipaddr,u8* port)
{
    u8 *p,*p1,*p2,*p3;
    u8 key;
    u16 timex=0;
    u8 count=0;
    const u8* cnttbl[3]={"正在连接","连接成功","连接关闭"};
    u8 connectsta=0; //0,正在连接;1,连接成功;2,连接关闭;
    u8 hbeaterrcnt=0; //心跳错误计数器,连续 5 次心跳信号无应答,则重新连接
    u8 oldsta=0xFF;
    p=mymalloc(SRAMIN,100); //申请 100 字节内存
    p1=mymalloc(SRAMIN,100); //申请 100 字节内存
    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    if(mode)Show_Str_Mid(0,30,"ATK-SIM800C UDP 连接测试",16,240);
    else Show_Str_Mid(0,30,"ATK-SIM800C TCP 连接测试",16,240);
    Show_Str(30,50,200,16,"KEY_UP:退出测试 KEY0:发送数据",12,0);
    sprintf((char*)p,"IP 地址:%s 端口:%s",ipaddr,port);
    Show_Str(30,65,200,12,p,12,0); //显示 IP 地址和端口
    Show_Str(30,80,200,12,"状态:",12,0); //连接状态
    Show_Str(30,100,200,12,"发送数据:",12,0); //连接状态
    Show_Str(30,115,200,12,"接收数据:",12,0); //端口固定为 8086
    POINT_COLOR=BLUE;
    USART3_RX_STA=0;
    sprintf((char*)p,"AT+CIPSTART=\"%s\", \"%s\", \"%s\"", modetbl[mode], ipaddr, port);
    if(sim800c_send_cmd(p, "OK", 500))return; //发起连接
    while(1)

```

```

{
    key=KEY_Scan(0);
    if(key==WKUP_PRES)//退出测试
    {
        sim800c_send_cmd("AT+CIPCLOSE=1","CLOSE OK",500); //关闭连接
        sim800c_send_cmd("AT+CIPSHUT","SHUT OK",500);      //关闭移动场景
        break;
    }else if(key==KEY0_PRES&(hbeaterrcnt==0)) //发送数据(心跳正常时发送)
    {
        Show_Str(30+30,80,200,12,"数据发送中...",12,0); //提示数据发送中
        if(sim800c_send_cmd("AT+CIPSEND",>",500)==0)//发送数据
        {
            printf("CIPSEND DATA:%s\r\n",p1); //发送数据打印到串口
            u3_printf("%s\r\n",p1);
            delay_ms(10);
            if(sim800c_send_cmd((u8*)0X1A,"SEND OK",1000)==0)
                Show_Str(30+30,80,200,12,"数据发送成功!",12,0); //最长等待 10s
            else Show_Str(30+30,80,200,12,"数据发送失败!",12,0);
            delay_ms(1000);
        }else sim800c_send_cmd((u8*)0X1B,0,0); //ESC,取消发送
        oldsta=0XFF;
    }
    if((timex%20)==0)
    {
        LED0=!LED0;
        count++;
        if(connectsta==2||hbeaterrcnt>8)
            //连接中断了,或者连续 8 次心跳没有正确发送成功,则重新连接
            {
                sim800c_send_cmd("AT+CIPCLOSE=1","CLOSE OK",500);
                //关闭连接
                sim800c_send_cmd("AT+CIPSHUT","SHUT OK",500);
                //关闭移动场景
                sim800c_send_cmd(p,"OK",500);
                //尝试重新连接
                connectsta=0;
                hbeaterrcnt=0;
            }
        sprintf((char*)p1,"ATK-SIM800C %s 测试 %d ",modetbl[mode],count);
        Show_Str(30+54,100,200,12,p1,12,0);
    }
    if(connectsta==0&&(timex%200)==0)
        //连接还没建立的时候,每 2 秒查询一次 CIPSTATUS.
    {

```

```

sim800c_send_cmd("AT+CIPSTATUS","OK",500); //查询连接状态
if(strstr((const char*)USART3_RX_BUF,"CLOSED"))connectsta=2;
if(strstr((const char*)USART3_RX_BUF,"CONNECT OK"))connectsta=1;
}
if(connectsta==1&&timex>=600)//连接正常的时候,每 6 秒发送一次心跳
{
    timex=0;
    if(sim800c_send_cmd("AT+CIPSEND",>",200)==0)//发送数据
    {
        sim800c_send_cmd((u8*)0X00,0,0); //发送数据:0X00
        delay_ms(20); //必须加延时
        sim800c_send_cmd((u8*)0X1A,0,0);
        //CTRL+Z,结束数据发送,启动一次传输
    }else sim800c_send_cmd((u8*)0X1B,0,0); //ESC,取消发送

    hbeaterrcnt++;
    printf("hbeaterrcnt:%d\r\n",hbeaterrcnt);//方便调试代码
}
delay_ms(10);
if(USART3_RX_STA&0X8000) //接收到一次数据了
{
    USART3_RX_BUF[USART3_RX_STA&0X7FFF]=0; //添加结束符
    printf("%s",USART3_RX_BUF); //发送到串口
    if(hbeaterrcnt) //需要检测心跳应答
    {
        if(strstr((const char*)USART3_RX_BUF,"SEND OK"))hbeaterrcnt=0;
        //心跳正常
    }
    p2=(u8*)strstr((const char*)USART3_RX_BUF,"+IPD");
    if(p2)//接收到 TCP/UDP 数据
    {
        p3=(u8*)strstr((const char*)p2,",");
        p2=(u8*)strstr((const char*)p2,":");
        p2[0]=0;//加入结束符
        sprintf((char*)p1,"收到%s 字节,内容如下",p3+1);//接收到的字节数
        LCD_Fill(30+54,115,239,130,WHITE);
        POINT_COLOR=BRED;
        Show_Str(30+54,115,156,12,p1,12,0); //显示接收到的数据长度
        POINT_COLOR=BLUE;
        LCD_Fill(30,130,210,319,WHITE);
        Show_Str(30,130,180,190,p2+1,12,0); //显示接收到的数据
    }
    USART3_RX_STA=0;
}
}

```

```

        if(oldsta!=connectsta)
        {
            oldsta=connectsta;
            LCD_Fill(30+30,80,239,80+12,WHITE);
            Show_Str(30+30,80,200,12,(u8*)cnttbl[connectsta],12,0); //更新状态
        }
        timex++;
    }
    myfree(SRAMIN,p);
    myfree(SRAMIN,p1);
}
//gprs 测试主界面
void sim800c_gprs_ui(void)
{
    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    Show_Str_Mid(0,30,"ATK-SIM800C GPRS 通信测试",16,240);
    Show_Str(30,50,200,16,"KEY_UP:连接方式切换",16,0);
    Show_Str(30,90,200,16,"连接方式:",16,0); //连接方式通过 KEY_UP 设置(TCP/UDP)
    Show_Str(30,110,200,16,"IP 地址:",16,0); //IP 地址可以键盘设置
    Show_Str(30,130,200,16,"端口:",16,0); //端口固定为 8086
    kbd_fn_tbl[0]="连接";
    kbd_fn_tbl[1]="返回";
    sim800c_load_keyboard(0,180,(u8**)kbd_tbl2); //显示键盘
}
//sim800c GPRS 测试
//用于测试 TCP/UDP 连接
//返回值:0,正常
// 其他,错误代码
u8 sim800c_gprs_test(void)
{
    const u8 *port="8086"; //端口固定为 8086,当你的电脑 8086 端口被其他程序占用的
                           //时候,请修改为其他空闲端口
    u8 mode=0; //0,TCP 连接;1,UDP 连接
    u8 key;
    u8 timex=0;
    u8 ipbuf[16]; //IP 缓存
    u8 iplen=0; //IP 长度
    sim800c_gprs_ui(); //加载主界面
    Show_Str(30+72,90,200,16,(u8*)modetbl[mode],16,0); //显示连接方式
    Show_Str(30+40,130,200,16,(u8*)port,16,0); //显示端口
    sim800c_send_cmd("AT+CIPCLOSE=1","CLOSE OK",100); //关闭连接
    sim800c_send_cmd("AT+CIPSHUT","SHUT OK",100); //关闭移动场景
    if(sim800c_send_cmd("AT+CGCLASS=\"B\"", "OK",1000))return 1;
}

```

```

//设置 GPRS 移动台类别为 B,支持包交换和数据交换
if(sim800c_send_cmd("AT+CGDCONT=1,\"IP\", \"CMNET\", \"OK\",1000))return 2;
//设置 PDP 上下文,互联网接协议,接入点等信息
if(sim800c_send_cmd("AT+CGATT=1,\"OK\",500))return 3; //附着 GPRS 业务
if(sim800c_send_cmd("AT+CIPCSGP=1,\"CMNET\", \"OK\",500))return 4;
//设置为 GPRS 连接模式
if(sim800c_send_cmd("AT+CIPHEAD=1,\"OK\",500))return 5;
//设置接收数据显示 IP 头(方便判断数据来源)
ipbuf[0]=0;
while(1)
{
    key=KEY_Scan(0);
    if(key==WKUP_PRES)
    {
        mode=!mode;    //连接模式切换
        Show_Str(30+72,90,200,16,(u8*)modetbl[mode],16,0); //显示连接模式
    }
    key=sim800c_get_keynum(0,180);
    if(key)
    {
        if(key<12)
        {
            if(iplen<15)
            {
                ipbuf[iplen++]=kbd_tbl[key-1][0];
                u3_printf("AT+CLDTMF=2,\"%c\"\\r\\n\",kbd_tbl[key-1][0]);
            }
        }else
        {
            if(key==13)if(iplen)iplen--;    //删除
            if(key==14&&iplen)    //执行 GPRS 连接
            {
                sim800c_tcpudp_test(mode,ipbuf,(u8*)port);
                sim800c_gprs_ui();    //加载主界面
                Show_Str(30+72,90,200,16,(u8*)modetbl[mode],16,0);
                //显示连接模式
                Show_Str(30+40,130,200,16,(u8*)port,16,0);//显示端口
                USART3_RX_STA=0;
            }
            if(key==15)break;
        }
        ipbuf[iplen]=0;
        LCD_Fill(30+56,110,239,110+16,WHITE);
        Show_Str(30+56,110,200,16,ipbuf,16,0);    //显示 IP 地址
    }
}

```

```

    }
    timex++;
    if(timex==20)
    {
        timex=0;
        LED0=!LED0;
    }
    delay_ms(10);
    sim_at_response(1);//检查 GSM 模块发送过来的数据,及时上传给电脑
}

return 0;
}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//蓝牙测试部分代码

//蓝牙 SPP 测试主界面
void sim800c_spp_ui(u16 x,u16 y)
{
    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    Show_Str_Mid(0,y,"ATK-SIM800C 蓝牙测试",16,240);
    Show_Str(x,y+40,200,16,"请选择:",16,0);
    Show_Str(x,y+60,200,16,"KEY2:发起配对请求模式",16,0);
    Show_Str(x,y+80,200,16,"KEYY0:接收配对请求模式",16,0);
    Show_Str(x,y+100,200,16,"KEY_DOWN:返回上一级",16,0);
}

//SIM800C 蓝牙连接模式测试
//用于模式选择和串口透传
//mode(模式): 0,接收配对模式
//          1,寻找设备模式
//返回值: 0,正常
//      其他,错误代码
u8 sim800c_spp_mode(u8 mode)
{
    u8 *p1,*p2;
    u8 key;
    u8 timex=1;
    u8 sendcnt=0;
    u8 sendbuf[20];
    u8 res;
    LCD_Clear(WHITE);
    POINT_COLOR=RED;

```

```

Show_Str_Mid(0,20,"ATK-SIM800C 蓝牙测试",16,240);
Show_Str(40,60,200,16,"KEYDOWN:返回上一级",16,0);

if(mode==0)//接收配对模式
{
    do
    {
        Show_Str(40,30+60,200,16,"等待连接请求...",16,0);
        res = sim800c_wait_request("+BTPAIRING:",600);
        //等待手机端蓝牙连接请求 6s
        if(res==1)//手机端连接请求
        {
            delay_ms(10);
            sim800c_send_cmd("AT+BTPAIR=1,1","BTPAIR:",500);//响应连接
        }
        else if(res==2) return 0;//按键返回上一级
        Show_Str(40,30+60,200,16,"",16,0);
        delay_ms(50);
        LED0=!LED0;
    }while(strstr((const char*)USART3_RX_BUF,"+BTPAIR: 1")!=NULL);
    //判断是否匹配成功
    USART3_RX_STA=0;
}
else if(mode==1)//寻找设备模式
{
    BT_Scan_mode=1;//设置蓝牙设备扫描标志
    do
    {
        Show_Str(40,30+60,200,16,"搜索设备中...",16,0);
        res=sim800c_send_cmd("AT+BTSCAN=1,11","BTSCAN: 1",1100);
        //搜索附近的蓝牙设备,搜索 11s
        //(重新配置定时器分频系数,定时为 1S 中断,蓝牙扫描
        // 结束后重新为 10ms 定时中断)
        if(res==2){BT_Scan_mode=0;return 0;}//按键返回上一级
        Show_Str(40,30+60,200,16,"",16,0);
        delay_ms(100);
        LED0=!LED0;
    }while(strstr((const char*)USART3_RX_BUF,"+BTSCAN: 0")!=NULL);
    //判断是否扫描到设备
    USART3_RX_STA=0;
    do
    {
        Show_Str(40,30+60,200,16,"发现设备", 16,0);
        res= sim800c_send_cmd("AT+BTPAIR=0,1","+BTPAIRING:",400);
    }
}

```

```

//连接搜索到的第一个设备
if(res==2){BT_Scan_mode=0;return 0;}//按键返回上一级
delay_ms(100);
Show_Str(40,30+60,200,16,"正在连接中.....",16,0);
sim800c_send_cmd("AT+BTPAIR=1,1","BTPAIR:",500);
//响应连接，如果手机长期不确认匹配，SIM800C 蓝牙端 30S 后才会
//上报配对失败
}while(strstr((const char*)USART3_RX_BUF,"+BTPAIR: 1")!=NULL);
//判断是否匹配成功
USART3_RX_STA=0;
BT_Scan_mode=0;//清除蓝牙设备扫描标志
}
Show_Str(40,30+60,200,16,"蓝牙连接成功",16,0);
do
{
if(!LED0)Show_Str(40,120,200,16,"等待 SPP 连接",16,0);
else Show_Str(40,120,200,16,"",16,0);
res = sim800c_wait_request("SPP",120);//等待手机端 SPP 连接请求
if(res==2)return 0; //按键返回上一级
else if(res==1) break;//SPP 连接成功
LED0=!LED0;
}while(1);
if(!sim800c_send_cmd("AT+BTACPT=1","+BTCONNECT:",300))
//应答手机端 spp 连接请求 3S
{
Show_Str(40,120,200,16,"SPP 连接成功",16,0);
delay_ms(1000);
Show_Str(40,120,200,16,"",16,0);
Show_Str(10,140,200,16,"发送数据:",16,0);
Show_Str(10,200,200,16,"接收数据:",16,0);
}
else
{
Show_Str(40,120,200,16,"SPP 连接失败",16,0);
return 0;//若一段时间内没有应答，则重新连接蓝牙！
}
while(1)
{
key = KEY_Scan(0);
if(key == KEY1_PRES) break;//按键返回上一级
if(USART3_RX_STA&0x8000)
{
USART3_RX_BUF[USART3_RX_STA&0X7FFF]=0;//添加结束符
USART3_RX_STA=0;
}
}

```



```

p1=(u8*)strstr((const char*)USART3_RX_BUF,"DATA: ");
if(p1!=NULL)
{
    p2=(u8*)strstr((const char *)p1,"\x0d\x0a");
    if(p2!= NULL)
    {
        p2=(u8*)strstr((const char *)p1,",");
        p1=(u8*)strstr((const char *)p2+1,",");
        // printf("接收到的数据是: ");
        // printf("%s\r\n",p1+1);//打印到串口
        LCD_Fill(90,200,320,480,WHITE);//清除显示
        LCD_ShowString(90,200,150,119,16,(u8*)(p1+1));
        //显示接收到的数据

    }
}
else
{
    p1=(u8*)strstr((const char*)USART3_RX_BUF,"BTDISCONN: ");
    //判断是否断开连接
    if(p1!=NULL)
    {
        Show_Str(40,30+60,200,16,"蓝牙连接断开",16,0);
        LCD_Fill(10,140,240,320,WHITE);//清屏
        delay_ms(1000);
        delay_ms(1000);
        delay_ms(1000);
        break;//退出
    }
}
}
timex++;
if(timex%50==0)
{
    timex=0;
    sim800c_send_cmd("AT+BTSPSEND", ">",100);//发送数据
    sprintf((char*)sendbuf,"Bluetooth test %d \r\n\32",sendcnt);
    sendcnt++;
    if(sendcnt>99) sendcnt = 0;
    res = sim800c_send_cmd((u8*)sendbuf,"OK",100);//发送数据
    if(res==0)
    {
        LCD_ShowString(90,140,209,119,16,(u8*)sendbuf);//显示发送的数据
        LED0=!LED0;
    }
}

```

```

        }
        else
        {
            Show_Str(40,30+60,200,16,"蓝牙连接断开",16,0);
            LCD_Fill(10,140,240,320,WHITE);//清屏
            delay_ms(1000);
            delay_ms(1000);
            delay_ms(1000);
            break;//退出
        }
    }
    delay_ms(10);
}
return 0;

}
//SIM800C 蓝牙连接模式选择
//返回值:0,正常
//      其他,错误代码
u8 sim800c_spp_test(void)
{
    u8 key;
    u8 timex=0;
    if(sim800c_send_cmd("ATE1","OK",200))//打开回显失败
    {
        //printf("打开回显失败");
        return 1;
    }
    delay_ms(10);
    if(sim800c_send_cmd("AT+BTPOWER=1","AT",300))
    //打开蓝牙电源 不判断 OK，因为电源原本开启再发送打开的话会返回 error
    {
        sim800c_send_cmd("ATE0","OK",200);//关闭回显功能
        return 1;
    }
    delay_ms(10);
    sim800c_spp_ui(40,30);
    while(1)
    {
        key=KEY_Scan(0);
        if(key==KEY1_PRES)
        {
            sim800c_send_cmd("ATE0","OK",300);//关闭回显功能
            break;
        }
    }
}

```

```

    }
    else if(key==KEY2_PRES)
    {
        sim800c_spp_mode(1);//寻找设备模式
        sim800c_spp_ui(40,30);
        sim800c_send_cmd("AT+BTUNPAIR=0","AT",120);//删除配对信息
        timex=0;
    }
    else if(key==KEY0_PRES)
    {
        sim800c_spp_mode(0);//接收配对模式
        sim800c_spp_ui(40,30);
        sim800c_send_cmd("AT+BTUNPAIR=0","AT",120);//删除配对信息
        timex=0;
    }
    timex++;
    if(timex==20)
    {
        timex=0;
        LED0=!LED0;
    }
    delay_ms(10);
    //sim_at_response(1); //检查 GSM 模块发送过来的数据,及时上传给电脑
}
return 0;
}

//ATK-SIM800C GSM/GPRS 主测试控制部分
//测试界面主 UI
void sim800c_mtest_ui(u16 x,u16 y)
{
    u8 *p,*p1,*p2;
    p=mymalloc(SRAMIN,50);//申请 50 个字节的内存
    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    Show_Str_Mid(0,y,"ATK-SIM800C 测试程序",16,240);
    Show_Str(x,y+25,200,16,"请选择:",16,0);
    Show_Str(x,y+45,200,16,"KEY0:拨号测试",16,0);
    Show_Str(x,y+65,200,16,"KEY1:短信测试",16,0);
    Show_Str(x,y+85,200,16,"KEY2:蓝牙测试",16,0);
    Show_Str(x,y+105,200,16,"WK_UP:GPRS 测试",16,0);
    POINT_COLOR=BLUE;
    USART3_RX_STA=0;
    if(sim800c_send_cmd("AT+CGMI","OK",200)==0) //查询制造商名称
    {

```

```

        p1=(u8*)strstr((const char*)(USART3_RX_BUF+2),"\r\n");
        p1[0]=0;//加入结束符
        sprintf((char*)p,"制造商:%s",USART3_RX_BUF+2);
        Show_Str(x,y+110+25,200,16,p,16,0);
        USART3_RX_STA=0;
    }
    if(sim800c_send_cmd("AT+CGMM","OK",200)==0)//查询模块名字
    {
        p1=(u8*)strstr((const char*)(USART3_RX_BUF+2),"\r\n");
        p1[0]=0;//加入结束符
        sprintf((char*)p,"模块型号:%s",USART3_RX_BUF+2);
        Show_Str(x,y+130+25,200,16,p,16,0);
        USART3_RX_STA=0;
    }
    if(sim800c_send_cmd("AT+CGSN","OK",200)==0)//查询产品序列号
    {
        p1=(u8*)strstr((const char*)(USART3_RX_BUF+2),"\r\n");//查找回车
        p1[0]=0;//加入结束符
        sprintf((char*)p,"序列号:%s",USART3_RX_BUF+2);
        Show_Str(x,y+150+25,200,16,p,16,0);
        USART3_RX_STA=0;
    }
    if(sim800c_send_cmd("AT+CNUM","+CNUM",200)==0)    //查询本机号码
    {
        p1=(u8*)strstr((const char*)(USART3_RX_BUF),",");
        p2=(u8*)strstr((const char*)(p1+2),"\");
        p2[0]=0;//加入结束符
        sprintf((char*)p,"本机号码:%s",p1+2);
        Show_Str(x,y+170+25,200,16,p,16,0);
        USART3_RX_STA=0;
    }
    myfree(SRAMIN,p);
}
//GSM 信息显示(信号质量,电池电量,日期时间)
//返回值:0,正常
//    其他,错误代码
u8 sim800c_gsminfo_show(u16 x,u16 y)
{
    u8 *p,*p1,*p2;
    u8 res=0;
    p=mymalloc(SRAMIN,50);//申请 50 个字节的内存
    POINT_COLOR=BLUE;
    USART3_RX_STA=0;
    if(sim800c_send_cmd("AT+CPIN?", "OK",200))res|=1<<0;    //查询 SIM 卡是否在位

```

```

USART3_RX_STA=0;
if(sim800c_send_cmd("AT+COPS?", "OK", 200)==0)           //查询运营商名字
{
    p1=(u8*)strstr((const char*)(USART3_RX_BUF), "\"");
    if(p1)//有有效数据
    {
        p2=(u8*)strstr((const char*)(p1+1), "\"");
        p2[0]=0;//加入结束符
        sprintf((char*)p, "运营商:%s", p1+1);
        Show_Str(x, y, 200, 16, p, 16, 0);
    }
    USART3_RX_STA=0;
}else res|=1<<1;
if(sim800c_send_cmd("AT+CSQ", "+CSQ:", 200)==0)           //查询信号质量
{
    p1=(u8*)strstr((const char*)(USART3_RX_BUF), ":");
    p2=(u8*)strstr((const char*)(p1), ",");
    p2[0]=0;//加入结束符
    sprintf((char*)p, "信号质量:%s", p1+2);
    Show_Str(x, y+20, 200, 16, p, 16, 0);
    USART3_RX_STA=0;
}else res|=1<<2;
if(sim800c_send_cmd("AT+CBC", "+CBC:", 200)==0)           //查询电池电量
{
    p1=(u8*)strstr((const char*)(USART3_RX_BUF), ",");
    p2=(u8*)strstr((const char*)(p1+1), ",");
    p2[0]=0; p2[5]=0;//加入结束符
    sprintf((char*)p, "电池电量:%s%%  %smV", p1+1, p2+1);
    Show_Str(x, y+40, 200, 16, p, 16, 0);
    USART3_RX_STA=0;
}else res|=1<<3;
if(sim800c_send_cmd("AT+CCLK?", "+CCLK:", 200)==0)       //查询电池电量
{
    p1=(u8*)strstr((const char*)(USART3_RX_BUF), "\"");
    p2=(u8*)strstr((const char*)(p1+1), ":");
    p2[3]=0;//加入结束符
    sprintf((char*)p, "日期时间:%s", p1+1);
    Show_Str(x, y+60, 200, 16, p, 16, 0);
    USART3_RX_STA=0;
}else res|=1<<4;
myfree(SRAMIN, p);
return res;
}
//NTP 网络同步时间

```

```

void ntp_update(void)
{
    sim800c_send_cmd("AT+SAPBR=3,1,\"Contype\\\", \"GPRS\\\", \"OK\",200); //配置承载场景 1
    sim800c_send_cmd("AT+SAPBR=3,1,\"APN\\\", \"CMNET\\\", \"OK\",200);
    sim800c_send_cmd("AT+SAPBR=1,1,0,200); //激活一个 GPRS 上下文
    delay_ms(5);
    sim800c_send_cmd("AT+CNTPCID=1\", \"OK\",200); //设置 CNTP 使用的 CID
    sim800c_send_cmd("AT+CNTP=\\\"202.120.2.101\\\",32\", \"OK\",200);
    //设置 NTP 服务器和本地时区(32 时区 时间最准确)
    sim800c_send_cmd("AT+CNTP\", \"+CNTP: 1\",600); //同步网络时间
}
//SIM800C 主测试程序
void sim800c_test(void)
{
    u8 key=0;
    u8 timex=0;
    u8 sim_ready=0;
    POINT_COLOR=RED;
    Show_Str_Mid(0,30,\"ATK-SIM800C 测试程序\",16,240);
    while(sim800c_send_cmd(\"AT\", \"OK\",100)) //检测是否应答 AT 指令
    {
        Show_Str(40,55,200,16,\"未检测到模块!!!\",16,0);
        delay_ms(800);
        LCD_Fill(40,55,200,55+16,WHITE);
        Show_Str(40,55,200,16,\"尝试连接模块...\",16,0);
        delay_ms(400);
    }
    LCD_Fill(40,55,200,55+16,WHITE);
    key+=sim800c_send_cmd(\"ATE0\", \"OK\",200); //不回显
    sim800c_mtest_ui(40,20);
    ntp_update(); //网络同步时间
    while(1)
    {
        delay_ms(10);
        sim_at_response(1); //检查 GSM 模块发送过来的数据,及时上传给电脑
        if(sim_ready) //SIM 卡就绪.
        {
            key=KEY_Scan(0);
            if(key)
            {
                switch(key)
                {
                    case KEY0_PRES:
                        sim800c_call_test(); //拨号测试

```

```

        break;
    case KEY1_PRES:
        sim800c_sms_test();//短信测试
        break;
    case WKUP_PRES:
        sim800c_gprs_test();//GPRS 测试
        break;
    case KEY2_PRES:
        sim800c_spp_test();//蓝牙 spp 测试
        break;

    }
    sim800c_mtest_ui(40,30);
    timex=0;
}
}
if(timex==0) //2.5 秒左右更新一次
{
    if(sim800c_gsminfo_show(40,225)==0)sim_ready=1;
    else sim_ready=0;
}
if((timex%20)==0)LED0=!LED0;//200ms 闪烁
timex++;
}
}

```

此部分代码比较多，我们挑几个重要的函数进行讲解一下。

首先，是检测模块应答函数：`u8* sim800c_check_cmd(u8 *str)`，该函数用于检测 ATK-SIM800C 模块发送回来的应答/数据，其中 `str` 为期待应答字符串，返回值如果为 0，则表示没有收到期待应答字符串，否则为期待应答字符串所在的位置。

第二个函数是：`u8 sim800c_send_cmd(u8 *cmd,u8 *ack,u16 waittime)`，该函数用于向 ATK-SIM800C 模块发送命令。`cmd` 为命令字符串，当 `cmd<=0xFF` 的时候，则直接发送 `cmd`，比如短信发送结束的时候，需要发送 `0x1A`，也可以通过该函数发送。`ack` 为期待应答字符串，`waittime` 为等待时间（单位：10ms）。由于在蓝牙 SPP 测试，寻找设备模式下，扫描时需要 11S 时间后才会有返回的设备信息，所以在这函数中，我们做了 `waittime` 判断，当设置 `waittime` 等待时间等于 1100（即 11S）时，会调用 `TIM7_SetARR()` 函数，重新设置定时器的预装载周期值，设置为 1S 的更新中断，当时间累加到 11S 时，定时器才重新设置为默认的 10ms 更新中断，继续实现两个字符接收间隔以 10ms 为标准。（具体的请查看 `timer.c` 中 `HAL_TIM_PeriodElapsedCallback` 定时器的回调函数和 `usart3.c` 中 `USART3_IRQHandler` 串口 3 的中断服务函数）

第三个函数是：`u8 sim800c_wait_request(u8 *request,u16 waittime)`，该函数用于等待 ATK-SIM800C 模块命令返回。`ack` 为期待应答字符串，`waittime` 为等待时间（单位：10ms）。该函数在蓝牙 SPP 测试中使用到。

第四个函数是：`void sim800c_unigbk_exchange(u8 *src,u8 *dst,u8 mode)`，该函数用于将输入字符串 `src` 转换为 UNICODE 编码字符串或者 GBK 内码，通过 `dst` 输出转换结果。`mode`

用于控制是 unicode 转换为 gbk (mode=0), 还是 gbk 转换为 unicode (mode=1)。该函数通过调用 FATFS 提供的 ff\_convert 函数实现 UNICODE 码与 GBK 码转换。

第五个函数是: u8 sim800c\_call\_test(void), 该函数用于拨号测试。通过虚拟键盘 (在 LCD 上触摸屏输入, 下同), 可以输入任意电话号码, 实现拨打电话功能, 并且在收到来电的时候, 可以通过虚拟键盘接听/挂断来电。

第六个函数是: void sim800c\_sms\_read\_test(void), 该函数用于读短信测试。该函数可以读取中英文短信, 通过虚拟键盘输入要读的短信编号, 即可读取短信, 并且语音报读, 其内容并显示在 LCD 上, 还可以显示短信状态 (已读/未读)、短信发送方号码、接收时间等信息。

第七个函数是: void sim800c\_sms\_send\_test(void), 该函数用于发短信测试。该函数可以向任意号码, 发送一条固定内容 (存放在 sim800c\_test\_msg) 的中英文短信。通过虚拟键盘输入电话号码, 即可实现短信发送。

第八个函数是: void sim800c\_tcpudp\_test(u8 mode,u8\* ipaddr,u8\* port), 该函数用于 TCP/UDP 通信测试。ipaddr 和 port 分别是目标 IP 地址及其端口号, mode 为 0 的时候, 进行 TCP 测试, mode 为 1 的时候进行 UDP 测试。该函数在连接成功后, 就可以实现和目标 IP 地址进行 TCP/UDP 数据通信, 收到的数据会显示在 LCD 上, 另外也可以通过按键 KEY0 向目标 IP 地址发送数据。该函数还带有心跳和自动重连功能, 可以实现长时间维持连接, 具有很高的实用价值。

第九个函数是: u8 sim800c\_gprs\_test(void), 该函数用于 GPRS 测试。通过 KEY\_UP 按键, 可以设置连接方式 (TCP/UDP), 通过虚拟键盘可以输入需要连接的目标 IP 地址, 端口号固定为: 8086。该函数通过调用 sim800c\_tcpudp\_test 函数实现 TCP/UDP 连接测试。

第十个函数是: u8 sim800c\_spp\_mode(u8 mode), 该函数用于蓝牙不同模式连接下的 SPP 通信, mode 为 0 的时候进行接收配对连接, mode 为 1 的时候进行寻找设备连接, 连接成功后, 就可以实现和手机端蓝牙 app 进行数据通信了, 收到的数据会显示在 LCD 上, 另外模块会自动发送数据到手机端 app 上。

第十一个函数是: u8 sim800c\_spp\_test(void), 该函数用于蓝牙连接模式选择。通过按 KEY2 按键, 设置寻找设备模式, 按 KEY0 按键, 设置为接收配对模式, 选择好模式后通过调用 u8 sim800c\_spp\_mode(u8 mode)函数实现蓝牙 SPP 数据通信。

最后要介绍的函数是: void sim800c\_test(void), 该函数是本 ATK-SIM800C 模块测试的主函数, 该函数将在 LCD 上面显示: 制造商、模块型号、序列号、本机号码、运营商、信号质量、电池电量和日期时间等参数。通过按键 KEY0, 可以进入拨号测试功能; 按键 KEY1, 可以进入短信测试功能; 按键 KEY\_UP, 可以进入 GPRS 测试功能; 按键 KEY2, 可以进入蓝牙 SPP 测试功能。

sim800c.c 我们就介绍到这里, 我们再来看看 main.c, 该文件里面就一个 main 函数, main 函数代码如下:

```
int main(void)
{
    u8 key,fontok=0;
    HAL_Init();                //初始化 HAL 库
    Stm32_Clock_Init(360,25,2,8); //设置时钟,180Mhz
    delay_init(180);           //初始化延时函数
    uart_init(115200);          //初始化 USART
    uart3_init(115200);         //初始化 USART3
    usmart_init(90);            //初始化 USMART
```



```

LED_Init();           //初始化 LED
KEY_Init();           //初始化按键
SDRAM_Init();         //SDRAM 初始化
LCD_Init();           //LCD 初始化
W25QXX_Init();        //初始化 W25Q256
tp_dev.init();        //初始化触摸屏
my_mem_init(SRAMIN);  //初始化内部内存池
my_mem_init(SRAMEX);  //初始化外部 SDRAM 内存池
my_mem_init(SRAMCCM); //初始化内部 CCM 内存池
exfuns_init();        //为 fatfs 相关变量申请内存
f_mount(fs[0], "0:", 1); //挂载 SD 卡
f_mount(fs[1], "1:", 1); //挂载 SPI FLASH.
key=KEY_Scan(0);
if(key==KEY0_PRES)    //强制校准
{
    LCD_Clear(WHITE); //清屏
    TP_Adjust();      //屏幕校准
    TP_Save_Adjdata();
    LCD_Clear(WHITE); //清屏
}
fontok=font_init();   //检查字库是否 OK
if(fontok || key==KEY1_PRES) //需要更新字库
{
    LCD_Clear(WHITE); //清屏
    POINT_COLOR=RED;  //设置字体为红色
    LCD_ShowString(60,50,200,16,16,"ALIENTEK STM32");
    while(SD_Init())  //检测 SD 卡
    {
        LCD_ShowString(60,70,200,16,16,"SD Card Failed!");
        delay_ms(200);
        LCD_Fill(60,70,200+60,70+16,WHITE);
        delay_ms(200);
    }
    LCD_ShowString(60,70,200,16,16,"SD Card OK");
    LCD_ShowString(60,90,200,16,16,"Font Updating...");
    key=update_font(20,110,16,"0:"); //从 SD 卡更新
    while(key) //更新失败
    {
        LCD_ShowString(60,110,200,16,16,"Font Update Failed!");
        delay_ms(200);
        LCD_Fill(20,110,200+20,110+16,WHITE);
        delay_ms(200);
    }
    LCD_ShowString(60,110,200,16,16,"Font Update Success!");
}

```

```

        delay_ms(1500);
        LCD_Clear(WHITE);          //清屏
    }
    sim800c_test();                //GSM 测试
}

```

此部分代码比较简单，由于本例程我们用到了触摸屏、12\*12 字体、16\*16 字体以及 UNICODE 与 GBK 转换码表，所以我们在 main 函数里面加入了触摸屏校准以及字库更新的代码。在启动的时候，按下 KEY0，可以进入触摸屏强制校准；在启动的时候，按下 KEY1，可以强制进行字库更新。

最后，通过调用 sim800c\_test()，进入 ATK-SIM800C 模块的主测试程序，开始对 ATK-SIM800C 的各项功能（拨号测试、短信测试、GPRS 测试、蓝牙测试）进行测试。

另外，为了方便大家调试，我们在本例程的 USMART 添加了 sim800c\_send\_cmd 这个函数，通过该函数，电脑可以通过串口 1，间接控制 ATK-SIM800c 模块。

比如（假设硬件已经准备好），我们通过串口调试助手可以将模块默认的字符集（IRA），设置为 UCS2 字符集，只需通过串口发送：sim800c\_send\_cmd("AT+CSCS=\"UCS2\"",0,0)，如图 3.2 所示（SIM 卡必须就位，否则该指令无效）：



图 3.2 通过 USMART 设置 ATK-SIM800C 字符集

至此，软件实现部分就介绍完了，我们接下来看代码验证。

## 4、验证

首先，请先确保硬件都已经连接好了：

- 1，给 ATK-SIM800C 模块装上 SIM 卡，并插好耳机和麦克风。

- 2, 连接 ATK-SIM800C 模块与阿波罗 STM32F429 开发板（连接方式见表：2.1）
- 3, 给 ATK-SIM800C 模块上电（按 K1, 蓝色电源指示灯亮）
- 4, ATK-SIM800C 模块开机（**长按 PWR\_KEY 键开机** 或 **用跳线帽短接 P1 口的 PKEY 与 VBAT 实现上电自动开机**, 红色 NET\_STA 指示灯闪烁）

本文档以 ALIENTEK 阿波罗 STM32F429 开发板为平台进行试验,在代码编译成功之后,我们下载代码到我们的 STM32 开发板上, LCD 显示如图 4.1 所示界面:



图 4.1 本测试实验界面

可以看到, LCD 上面显示了: 制造商、模块型号、序列号、本机号码、运营商、信号质量、电池电量以及日期时间等信息。**注意: 必须等到屏幕显示运营商后, 才可以通过 KEY0/KEY1/KEY2/KEY\_UP 这四个按键, 即可选择不同的测试项目, 进行测试。**

#### 4.1 拨号测试

在主界面, 按 KEY0, 则可以进入此项测试, 此项测试我们可以测试 ATK-SIM800C 模块的拨打电话或者接听来电等功能。拨号测试主界面如图 4.1.1 所示:



图 4.1.1 拨号测试主界面

在此界面，我们可以输入您要拨打的电话号码进行拨号。比如拨打 10086，输入 10086，然后点击“拨号”，就可以进行拨号了，如图 4.1.2 所示：

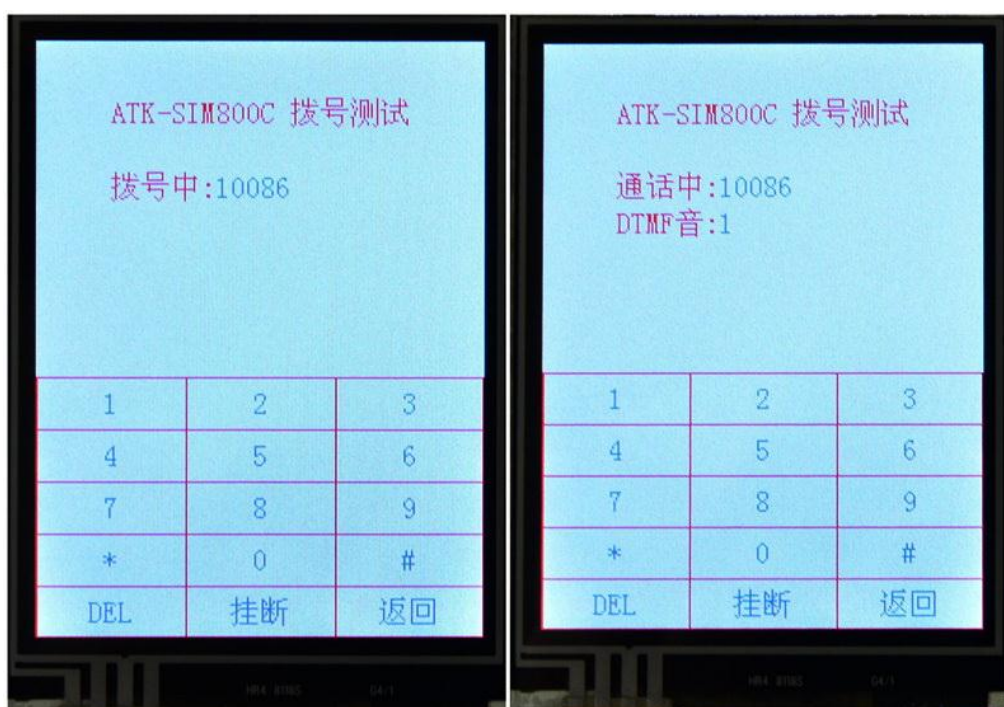


图 4.1.2 拨号测试

图 4.1.2 中，左侧图片为正在拨号中的界面，在拨号接通后，界面如图 4.3 右侧图片所示，此时，我们可以通过键盘输入数字，来产生 DTMF 音，实现数字输入。比如图中我们点击数字 1，可以查询话费余额。

在拨号测试主界面，如果有电话接入，则会提示有来电，并显示来电电话号码，如图

4.1.3 所示:

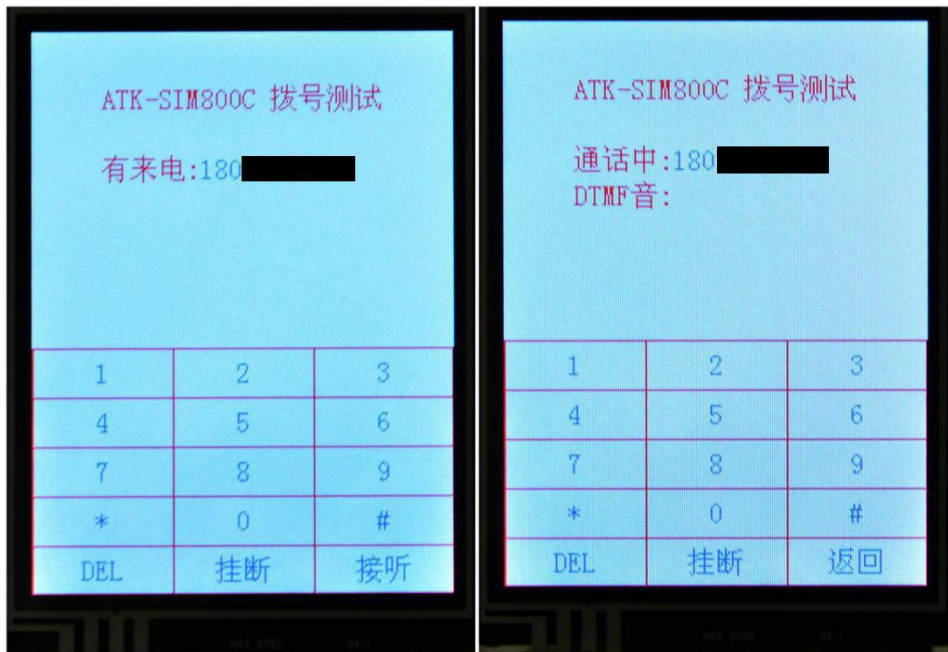


图 4.1.3 接听来电

图 4.1.3 中，左侧图片为来电提醒图片，此时可以在耳机听到来电铃声，我们通过点击“接听”即可接听来电，或者点击“挂断”，拒绝接听。在接通来电后，我们就可以和对方进行通话了，界面如图 4.4 右侧图片所示。

最后，按“返回”键，可以返回主界面。

## 4.2 短信测试

在主界面，按 KEY1，则可以进入此项测试，此项测试我们可以测试 ATK-SIM800C 模块的短信读取与短信发送功能。短信测试主界面如图 4.2.1 所示：

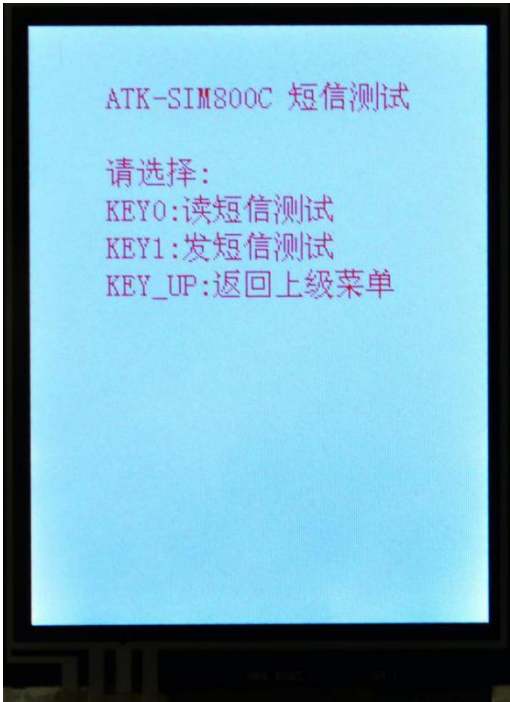


图 4.2.1 短信测试主界面



在此界面，我们按 KEY0 可以进入读短信测试，如图 4.2.2 所示：

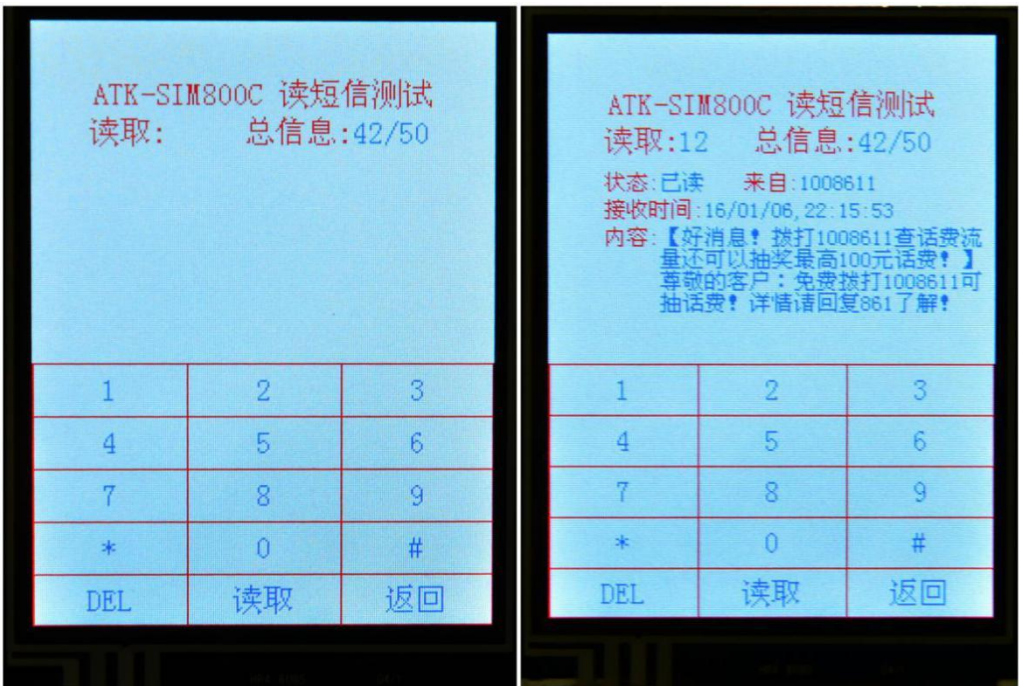


图 4.2.2 读短信测试

图 4.2.2 中，左侧图片为刚进入读短信测试时候的界面，此时可以看到总信息提示，当前 SIM 卡中有 42 条短信，最多存储 50 条。我们通过键盘输入 12，点击“读取”，即可中文语音读取第 12 条短信，如图 4.2.2 中，右侧图片所示，图中不仅显示了读取到的短信内容，还显示了当前短信的状态为：已读，来自：1008611，接收时间为：2016 年 1 月 6 号，22:15:53 等信息。

回到短信测试主界面，按 KEY1，可以进入短信发送测试，输入对方手机号码，我们就可以将一条固定内容的短信，发送到对方手机，如图 4.2.3 所示：

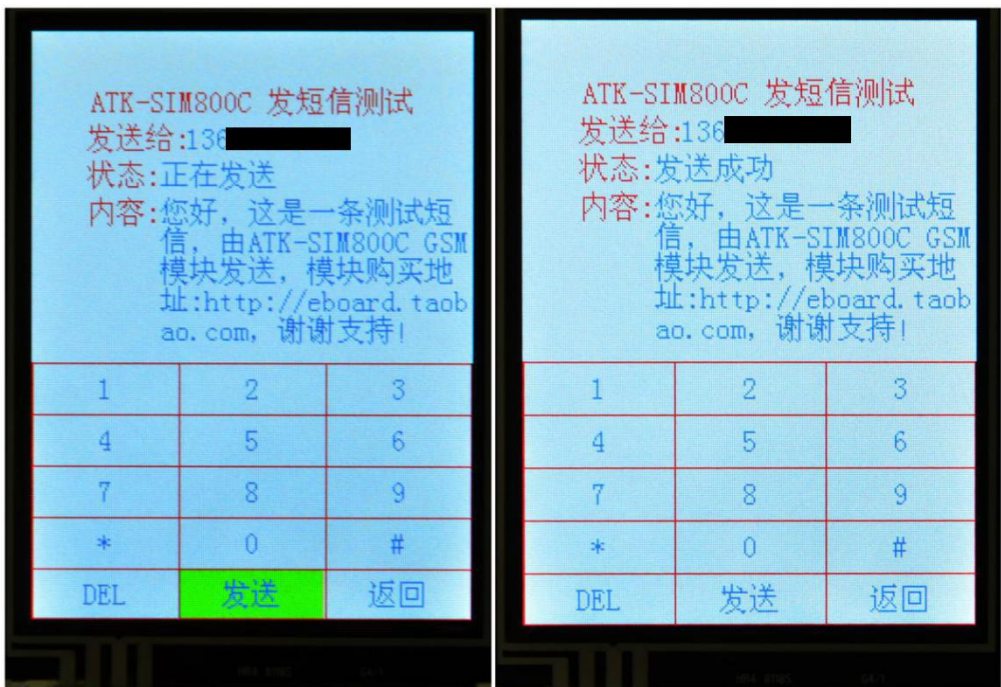


图 4.2.3 发短信测试

图 4.2.3 中，我们给自己发送了一条短信，左侧为短信发送时的界面，发送成功后如右侧图片所示。为了验证我们刚刚发送的短信是否成功，我们可以再次进入读短信测试界面，如图 4.2.4 所示：



图 4.2.4 收到刚刚发送给自己的短信

可以看到短信条数变为 43 条，由于接收到的短息不一定就是在最后一条中，我们读取第 3 条短信，和我们刚刚发送的内容是一模一样的。

证明我们刚刚发送的短信确实是发送成功了。

### 4.3 蓝牙测试

在主界面，按 KEY2，则可以进入此项测试，此项测试我们可以测试 ATK-SIM800C 模块的蓝牙通信功能。蓝牙测试主界面如图 4.3.1 所示：

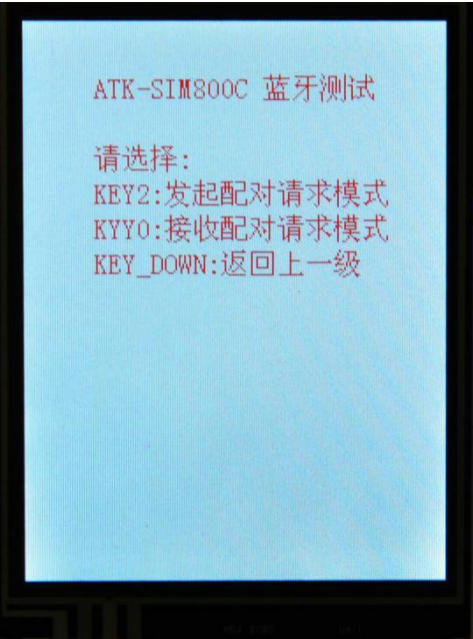


图 4.3.1 蓝牙测试主界面

在图 4.3.1 所示界面,我们可以通过按键 KEY2 和 KEY0 分别选择连接的模式,我们按 KEY2 可以进入发起配对请求模式, 如图 4.3.2 所示:

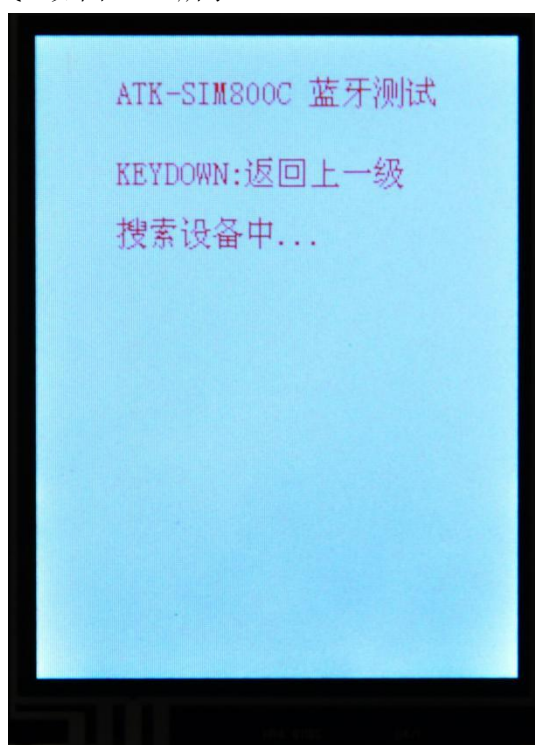


图 4.3.2 发起配对测试模式

图 4.3.2 中,模块正显示搜索设备中,这时手机端开启蓝牙设备,然后手机端接收到模块的连接请求后,配对连接成功后,这时显示等待 SPP 的连接,如图 4.3.3 所示:

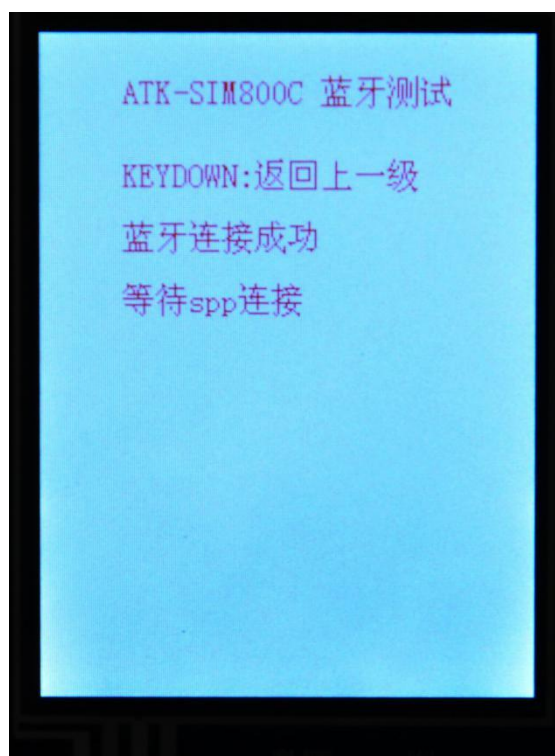


图 4.3.3 等待 SPP 连接

手机端打开已经安装好的蓝牙串口助手增强版 APP 这个软件,扫描并且连接 SIM800C 模



块，连接配对成功后，手机端点击**字节流模式**、液晶显示屏显示模块发送给手机端的数据，而且，手机端发送的数据会在液晶显示屏上显示，如图 4.3.4 所示

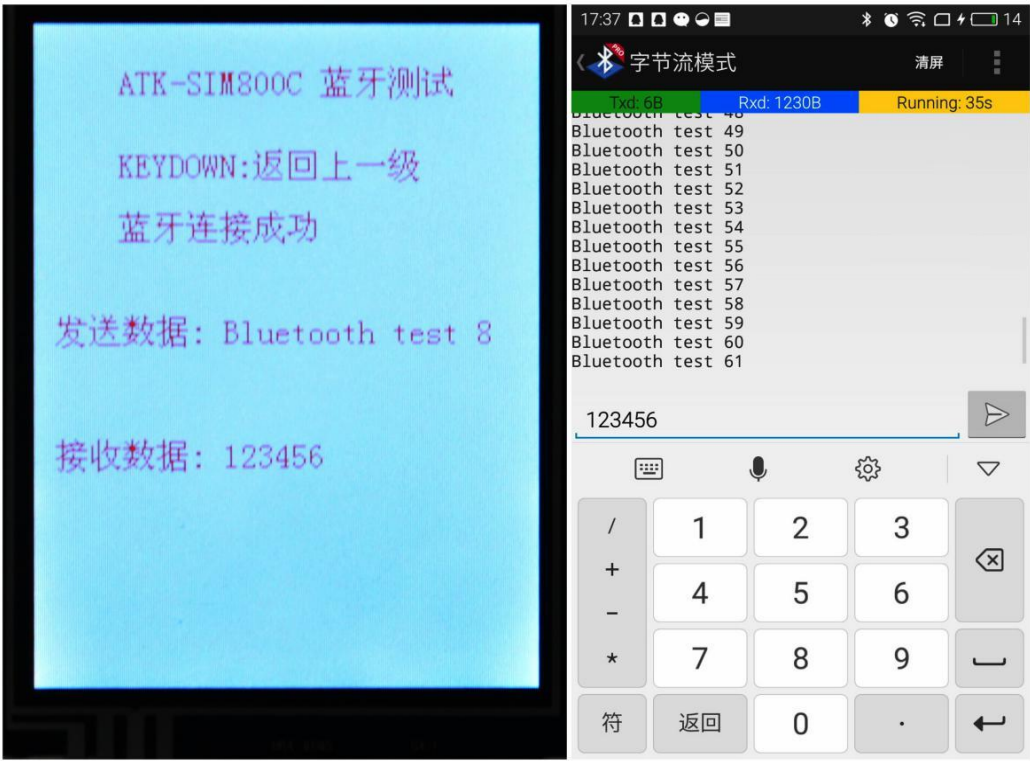


图 4.3.4 数据显示

回到蓝牙测试主界面，按 KEY0 可以进入接收配对请求模式，如图 4.3.5 所示：

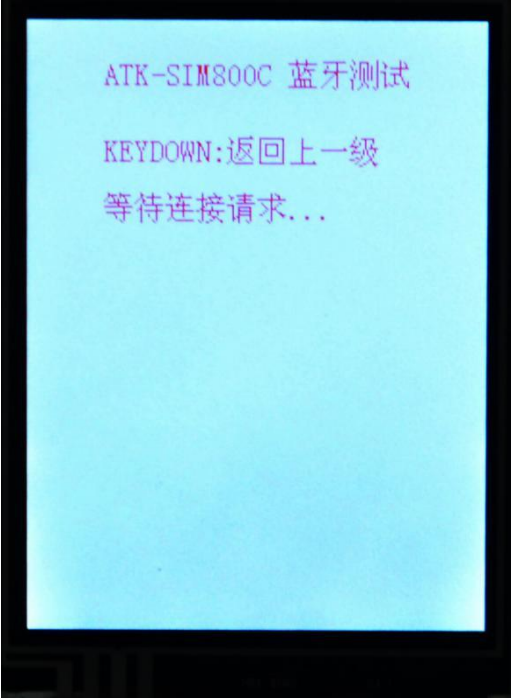


图 4.3.5 接收配对请求模式

图 4.3.5 中，模块正显示等待连接请求，手机端通过进入（设置->蓝牙），去连接可用设备 SIM800C，配对连接成功后，显示等待 spp 的连接，如图上面 4.3.3 所示，spp 的连接以及数据通信与上面的发起配对请求模式步骤和效果一样，如图 4.3.4 数据显示 所示。

## 4.4 GPRS 测试

在主界面，按 KEY\_UP，则可以进入此项测试，此项测试我们可以测试 ATK-SIM800C 模块的 GPRS 通信功能，包括 TCP 和 UDP 通信。GPRS 测试主界面如图 4.4.1 所示：

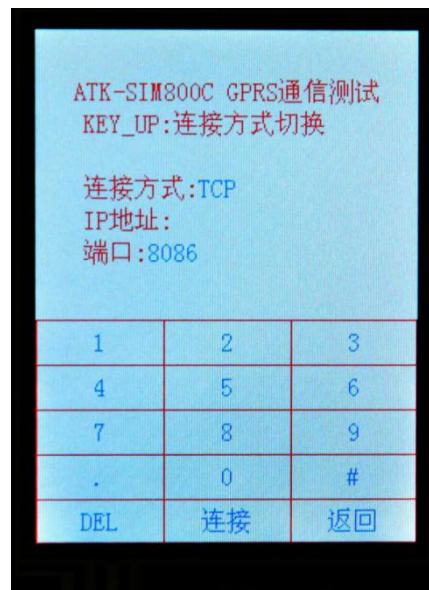


图 4.4.1 GPRS 测试主界面

在图 4.4.1 所示界面，我们可以通过键盘输入目标 IP 地址，可以通过 KEY\_UP 按键切换连接方式（TCP/UDP），连接端口号固定为：8086。本测试需要电脑配合测试，我们的电脑需要有一个公网 IP（具体方法见 1.2.4 节），这里我的电脑公网 IP 为：113.67.34.69，并在电脑端运行：网络调试助手。

首先来看 TCP 连接，我们先在电脑端运行网络调试助手，选择协议类型为：TCP Server，并设置本地端口号为：8086，然后点击连接，如图 4.4.2 所示：

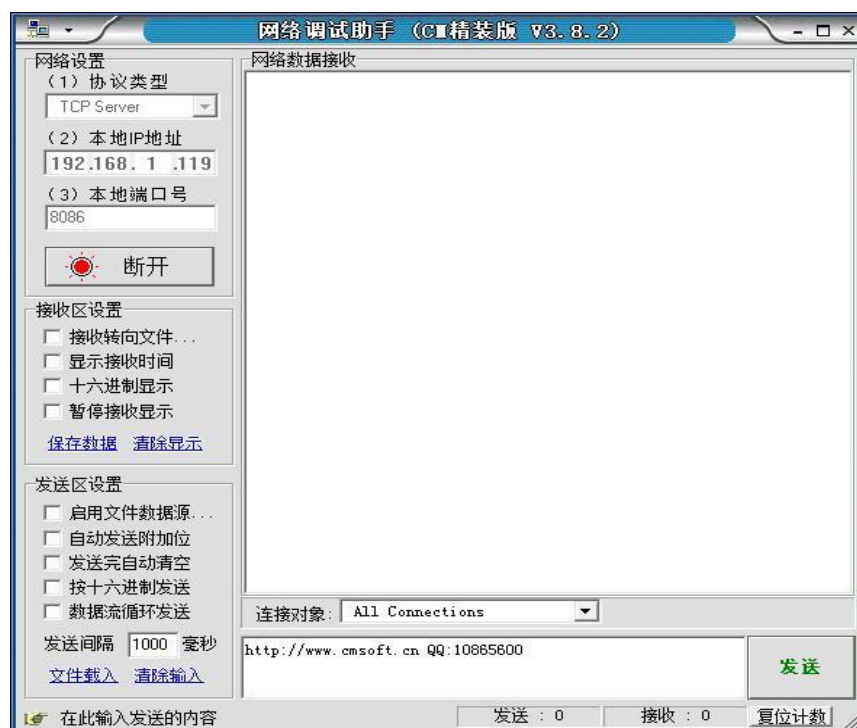


图 4.4.2 TCP 连接网络调试助手设置

然后，我们输入目标 IP 地址：113.67.34.69，如图 4.4.3 左侧图片所示：

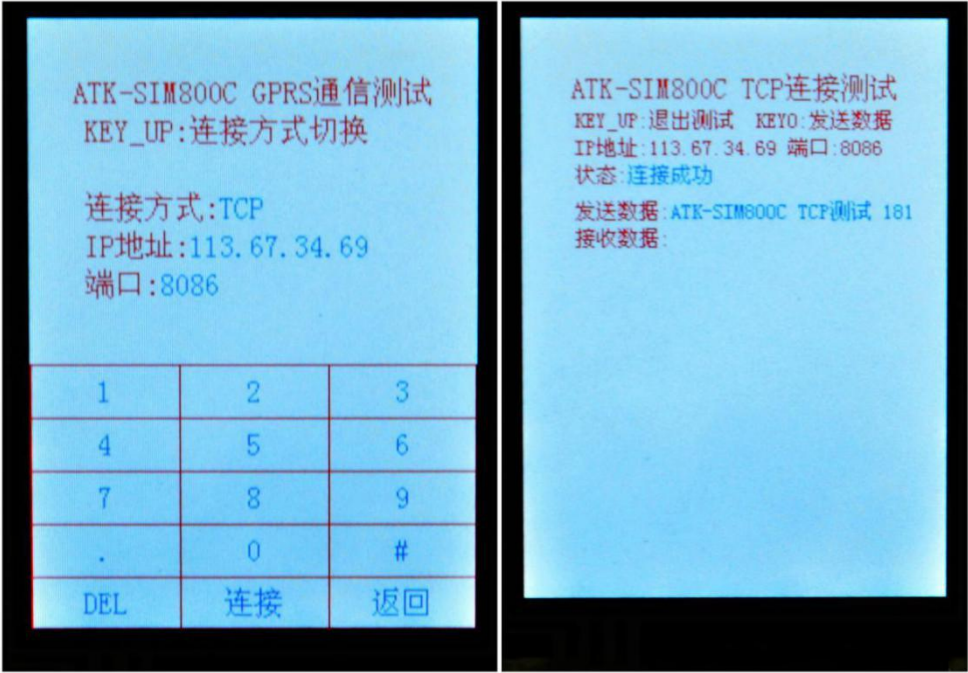


图 4.4.3 TCP 连接输入 IP 地址及建立连接

在输入 IP 地址之后，我们点击连接，即可与电脑端的网络调试助手建立一个 TCP 连接。稍等片刻，连接成功后如图 4.4.3 右侧图片所示。

连接成功后，我们便可以通过网络调试助手给 ATK-SIM800C 模块发送数据，也可以通过 ATK-SIM800C 模块给网络调试助手发送数据，如图 4.4.4 和 4.4.5 所示：

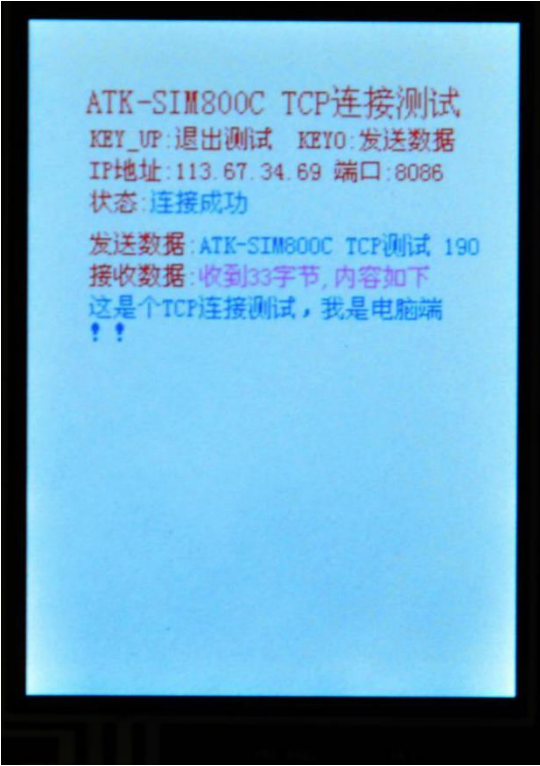


图 4.4.4 TCP 通信接收到电脑端的数据



图 4.4.5 TCP 通信收到来自 ATK-SIM800C 模块的数据

以上就是 TCP 连接测试，UDP 链接测试与这个非常相似：首先在电脑端运行网络调试助手，选择协议类型为：UDP，其他同 TCP 连接测试时一模一样。然后在 GPRS 测试主界面选择 UDP 测试，输入目标 IP 地址，点击“连接”，在连接成功后，我们就可以在电脑和模块之间实现 UDP 通信了，如图 4.4.6 和 4.4.7 所示：



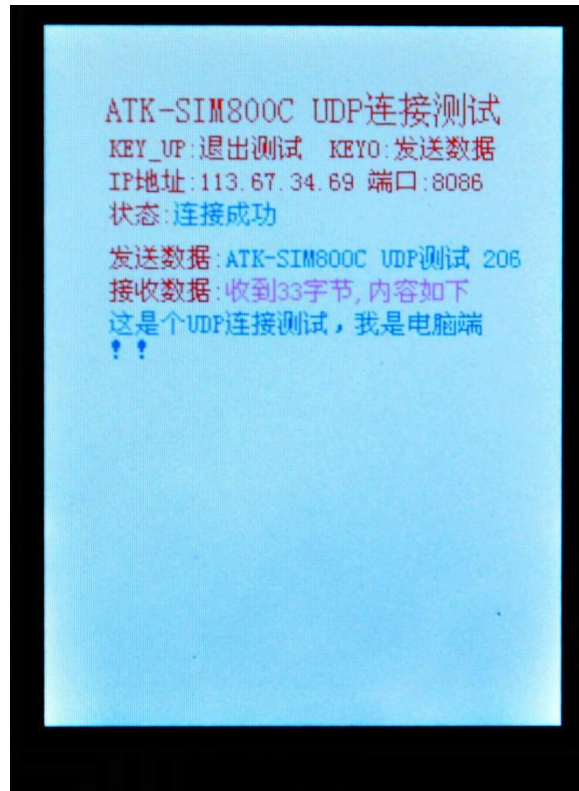


图 4.4.6 UDP 通信接收到电脑端的数据

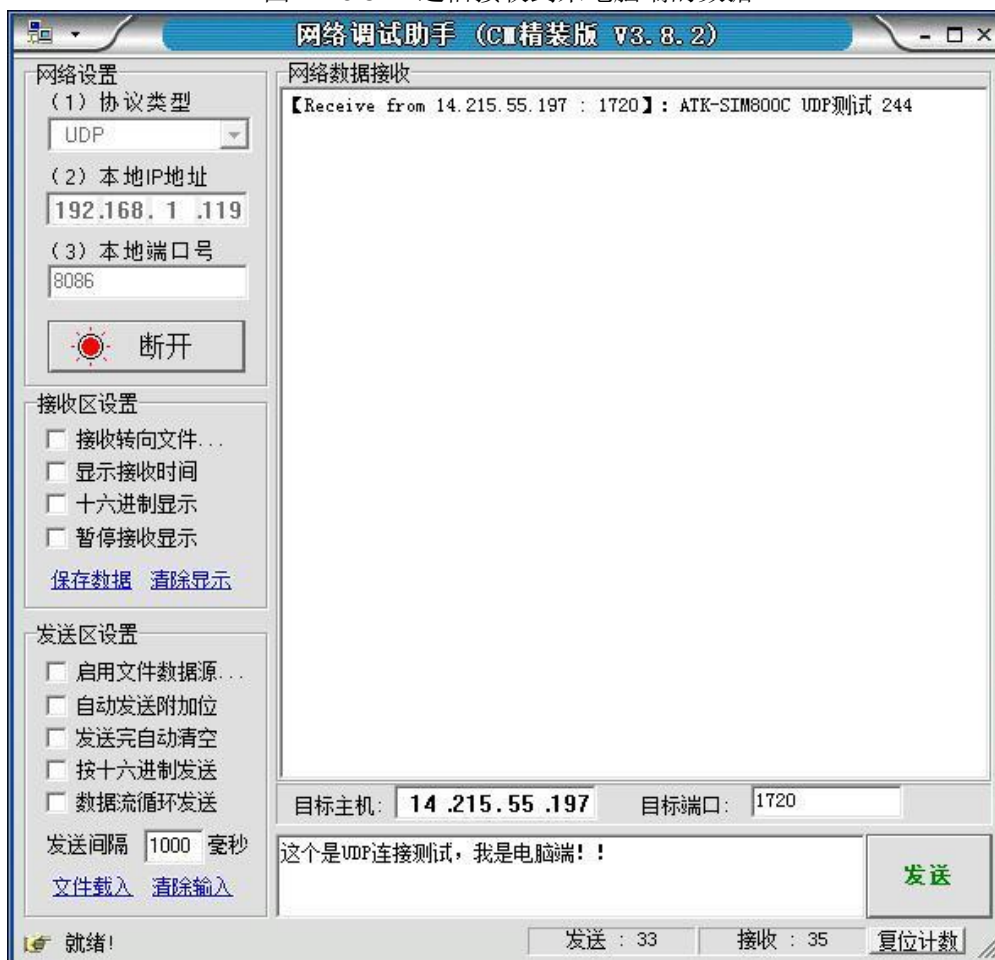


图 4.4.7 UDP 通信收到来自 ATK-SIM800C 模块的数据

注意，网络调试助手在接收到新数据时，会在前面加数据来源的提示头，如：**【Receive from 14.215.55.197: 1720】**：，用于指示当前数据来源。从这个头我们可以知道，当前数据来自：14.215.55.197，端口号为：1720，这个 IP 地址和端口，是运营商给我们的 ATK-SIM80C 模块随机分配的一个 IP 和端口，也就是 SIM 卡的 IP 地址。需要注意的是，移动为 SIM 卡分配的 IP 地址和端口号，是随机的，不是固定的，所以每次新连接的建立，这个 IP 和端口号一般都是不相同的。

至此，关于 ATK-SIM800C GSM/GPRS 模块的使用介绍，我们就讲完了，本文档详细介绍了 ATK-SIM800C 模块的使用，有助于大家快速学会 ATK-SIM800C 模块的使用。

正点原子@ALIENTEK

2016-6-18

公司网址：[www.alientek.com](http://www.alientek.com)

技术论坛：[www.openedv.com](http://www.openedv.com)

电话：020-38271790

传真：020-36773971

