

**SMS ALERT SYSTEM**

[www.smsalert.cn](http://www.smsalert.cn)

**短信猫二次开发指南**

**GSM MODEM SOFTWARE DEVELOPMENT KIT**

**SDK Ver2.2**

**2007 年 3 月**

## 目 录

1	产品介绍.....	4
1.1	功能特点.....	4
1.2	产品配件.....	4
1.3	广泛应用.....	6
2	联机指南.....	7
3	产品测试.....	10
3.1	测试工具.....	10
3.2	AT 测试指令.....	11
4	AT 指令集与开发参考.....	14
5	JAVA 二次开发指南.....	18
5.1	总体概述.....	18
5.2	GSMMultiPort 类 —— 基础短信开发接口.....	19
5.2.1	常量说明.....	19
5.2.2	GSMModemAutoTest —— 自动检测获取通讯信息.....	19
5.2.3	GSMModemGetSnInfo —— 获取注册信息码.....	21
5.2.4	GSMModemInit —— 初始化短信设备.....	21
5.2.5	GSMModemSMSsend —— 发送短信息.....	22
5.2.6	GSMModemSMSReadAll —— 读取短信.....	22
5.2.7	GSMModemGetErrorMsg —— 获取过程错误信息.....	26
5.2.8	GSMModemIsConn —— 获取当前连接状态.....	26
5.2.9	GSMModemRelease —— 断开连接，并释放资源.....	26
5.3	GSMModem 类 —— 接口应用类.....	28
5.3.1	属性.....	28
5.3.2	GSMModemPortInfos —— 自动获取所有通讯信息.....	28
5.3.3	GSMModemGetSnInfo —— 获取注册信息码.....	29
5.3.4	GSMModemInit —— 初始化短信设备.....	29
5.3.5	GSMModemSMSsend —— 发送短信息.....	29
5.3.6	GSMModemSMSReadAll —— 读取短信.....	29
5.3.7	GSMModemGetErrorMsg —— 获取过程错误信息.....	30
5.3.8	GSMModemIsConn —— 获取当前连接状态.....	30
5.3.9	GSMModemRelease —— 断开连接，并释放资源.....	30
5.4	GSMMessage 类 —— 短信存储类（略）.....	31
5.5	COMInfo 类 —— 通讯端口信息类（略）.....	32
5.6	GSMCommon 类 —— 公共方法类（略）.....	32
5.7	演示代码.....	33
5.8	开发包说明.....	33
6	VC/C 二次开发指南.....	33
6.1	函数说明.....	33
6.1.1	GSMModemAutoTest —— 自动检测获取通讯信息.....	33
6.1.2	GSMModemGetSnInfo —— 获取注册信息码.....	33
6.1.3	GSMModemInit —— 初始化短信设备.....	34
6.1.4	GSMModemSMSsend —— 发送短信息.....	34

6.1.5	GSModemSMSReadAll —— 读取短信.....	34
6.1.6	GSModemGetErrorMsg —— 获取过程错误信息.....	38
6.1.7	GSModemIsConn —— 获取当前连接状态 .....	38
6.1.8	GSModemRelease —— 断开连接, 并释放资源.....	38
6.2	演示代码.....	38
6.3	开发包说明.....	39
7	C#二次开发指南 .....	40
7.1	函数声明.....	40
7.2	演示界面.....	42
7.3	演示代码.....	42
8	VB.NET 二次开发指南 .....	42
8.1	函数声明.....	42
8.2	演示界面.....	43
8.3	演示代码.....	43
9	Delphi 二次开发指南.....	44
9.1	函数声明.....	44
9.2	演示界面.....	45
9.3	演示代码.....	45
10	VB6 二次开发指南 .....	46
10.1	函数声明.....	46
10.2	演示界面.....	47
10.3	演示代码.....	47
11	PB 二次开发指南.....	47
11.1	函数声明.....	48
11.2	演示界面.....	49
11.3	演示代码.....	49

## 短信猫介绍及接口开发指南

短信猫，又名 GSM MODEM，专门针对短信应用设计，内含工业级短信发送模块，简化了通信接口，性能稳定可靠，符合各种商业和工业级短信应用要求，支持向移动、联通以及小灵通用户收发短信，适用于各行各业各个领域作无线数据通信，短信息通告，短信查询等应用。

### 1 产品介绍



图 (1)

#### 1.1 功能特点

- ◆ 模块：WAVECOM Q2403A
- ◆ 频段：双频 EGSM900/GSM1800
- ◆ 标准：兼容 GSM Phase2/2+ 标准
- ◆ 服务：支持语音、数据、短消息和传真服务
- ◆ 低功耗：Class4(2W@900MHz) / Class1(1W@1800MHz)
- ◆ 外形：小巧耐震铝外壳设计，外形美观
- ◆ 天线：外置工业级天线，设备放置在任意地方均可
- ◆ 接口：RS-232
- ◆ 控制：AT 命令

#### 1.2 产品配件

- ◆ 充电器：专用电源充电器，如图 (2)。



图 (2)

- ◆ 通讯线：串口数据通讯线



图 (3)

- ◆ 天线：外置天线（下面二图中三种不同形式的天线）



图 (4.1)



图 (4.2)

### 1.3 广泛应用

工业级 **DG-C1A** 适用于各行各业，各个领域作无线数据通信、短信息通告之用。

- ◆ **>>与金仓短信王软件结合使用:**DG-C1A 短信猫设备与金仓短信王软件结合可以让客户建立自主的企业短信平台，实现以下功能：
  - 短信自动发送，支持号段、群组群发，实时或定时发送
  - 短信自动接收存储，自动辨别发信人，可指定用户收阅
  - 短信分信箱管理，分为待发送信箱、已发送信箱、发送失败信箱、收信箱
  - 通讯录分群组管理，支持文件导入联系人，支持联系人信息备份恢复
  - 支持常用短语维护，可在短信编辑时插入短语
  - 支持多路短信收发设备并发操作，也可指定模块收发
  - 支持自动路由，接收信人手机号自动分移动、联通模块发送
  - 支持短信自动分割功能，短信内容超过 70 字自动分割分条发送
  - 支持自动回复功能，可定制短信自动回复代码及内容并支持动态回复扩展功能
  - 支持短信发送审批：在短信发送过程中嵌入 workflow 管理
  - 支持用户权限管理，可设置用户角色，密码登录安全可靠
  - 支持短信群发批量导入功能
  
- ◆ **>>与金仓家校通软件结合使用:** DG-C1A 短信猫设备与金仓家校通软件结合构建学生平安短信平台，可实现以下应用：
  - 发送平安短信
  - 年级维护
  - 班级管理
  - 教师管理
  - 学生管理
  - 卡片管理
  - 配置管理
  - 考勤管理
  - 数据统计

◆ >>行业应用:

- 政府办公: 办公短信通知、短信日程提醒、应急信息短信发布、短信审批等
- 银行: 短信 CRM、短信帐务变动通知
- 保险: 保单查询、续费提醒、客户生日提醒、保费计算等
- 电力: 监控信息通知、客户缴费通知等
- 考试培训中心: 培训通知、考分查询等
- 民办大学: 内部短信办公、家校通
- 移动运营商: 短信促销、VIP 客户管理、大客户营销工具
- 制造业企业: 短信 CRM、短信商品防伪
- 物流企业: 短信物流跟踪
- 证券行业: 股价短信查询、短信交易提醒
- 餐饮行业: 短信促销、VIP 客户管理
- 商品流通业: 商场促销、VIP 客户管理
- 会员制俱乐部: 短信会员管理、客户关怀等
- 影戏院: 短信影院信息查询、短信订票等

## 2 联机指南

DG-C1A 与计算机串口的连接步骤和注意事项说明如下:

- 1) **使用普通手机测试 SIM 卡是否能正常收发短信:** 即先通过手机菜单进入短信设置项查看短信中心号码是否设置正确, 然后向其他手机号码发送短信, 对方是否能收到短信。再用其他手机向该 SIM 卡发送短信, 是否能收到短信。若都成功, 则该 SIM 卡能够正常工作。
- 2) **安装 SIM 卡:** 用圆珠笔或其他工具顶 SIM 卡座按钮 (图 5 中弹出卡座的右边小黑点), 设备会弹出 SIM 卡座, 将 SIM 卡放入卡座, 再将卡座插回设备的 SIM 卡插孔即可。如图 (5), (6), (7)。



图 (5)



图 (6)



图 (7)

- 3) **安装天线**: 在设备后面有一铜螺柱, 将天线拧上。注意: **天线的方向最好为垂直向上**。如图 (8)。



图 (8)

- 4) **连接串口线**: 将圆形插针与 DG-C1A 相连, 另一头为 9 针 DB 插座与计算机相连即可。**注意: 要插紧, 否则会影响数据通讯。**如图 (9), (10)。



图 (9)



图 (10)

- 5) **连接电源**: 将电源线的圆形插孔插入设备的电源插孔。**检查以上步骤是否都完成了。**最后将电源插座查到电源插线板上即可。如图 (11)。



图 (11)

- 6) 通过指示灯检查连接状态：正确连接指示灯为：红灯间歇闪烁（见图 12 中右侧红灯）。



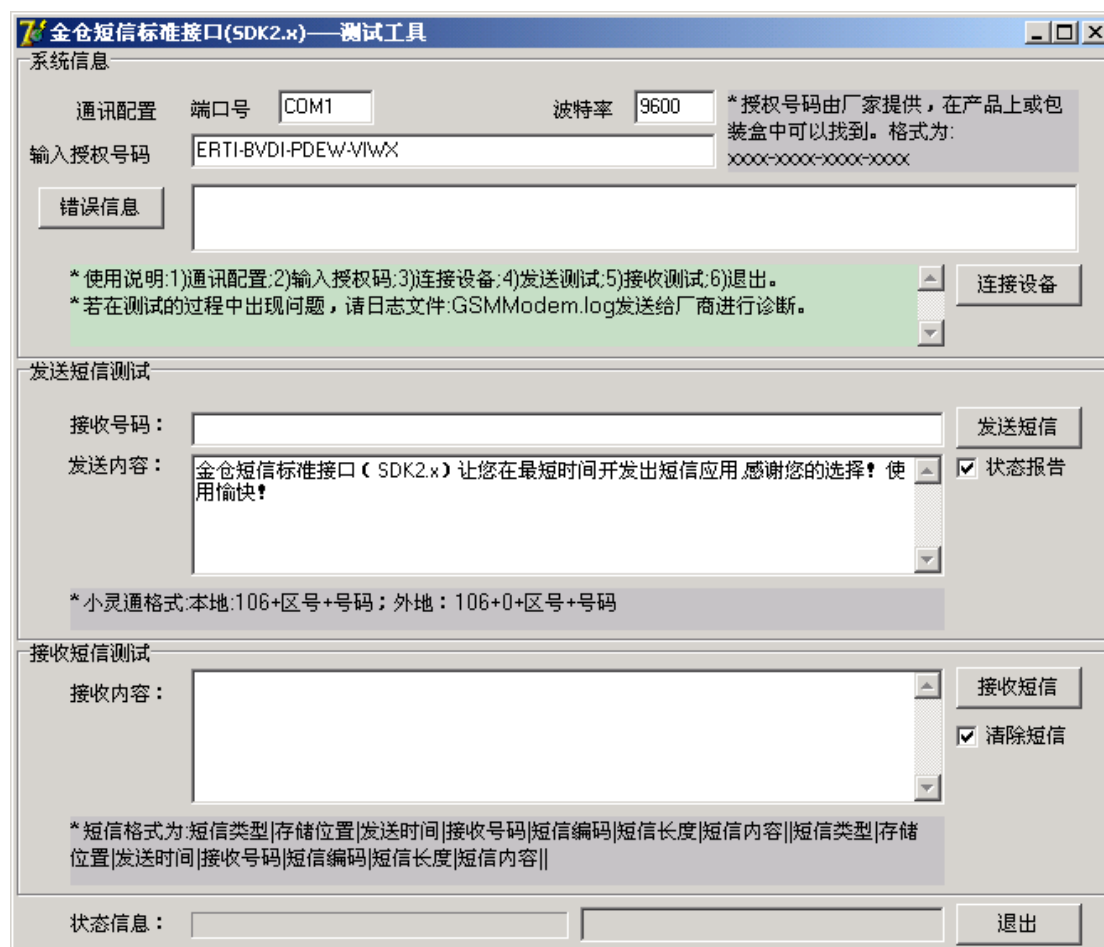
图 (12)

- 7) 系统可以正常使用了，**DG-C1A** 的二次开发接口会自动检查设备与计算机连接的通讯端口。

### 3 产品测试

#### 3.1 测试工具

在 SDK 包中提供了友好界面的测试工具：Tools\ ModemTools.exe，双击执行的界面如下：



通过使用此工具：可以进行短信收发测试。若在测试的过程中出现问题，请将日志文件:GSMModem.log 发送给厂商进行诊断。

### 3.2 AT 测试指令

当您的 DG-C1A 与计算机连接好后，您可以通过 WINDOWS 自带的超级终端，进行简单的检测设备是否正常，当然您也可以通过我们的测试程序进行测试。

通过超级终端的连接方式如下：

- 1) 进入超级终端：

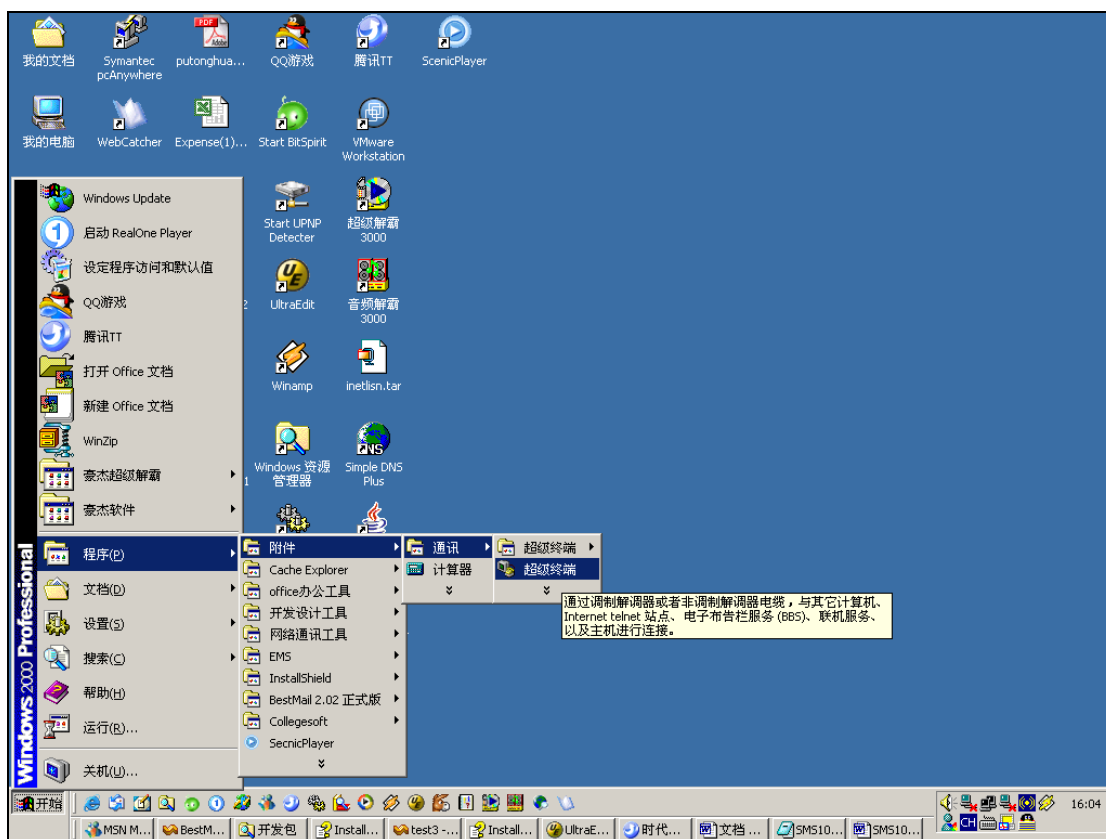


图 (4)

2) 设置连接参数:



图 (5)

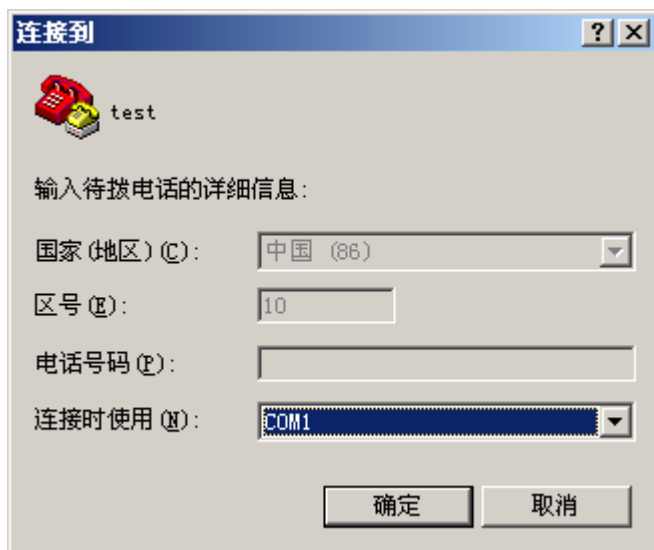


图 (6)



图 (7)

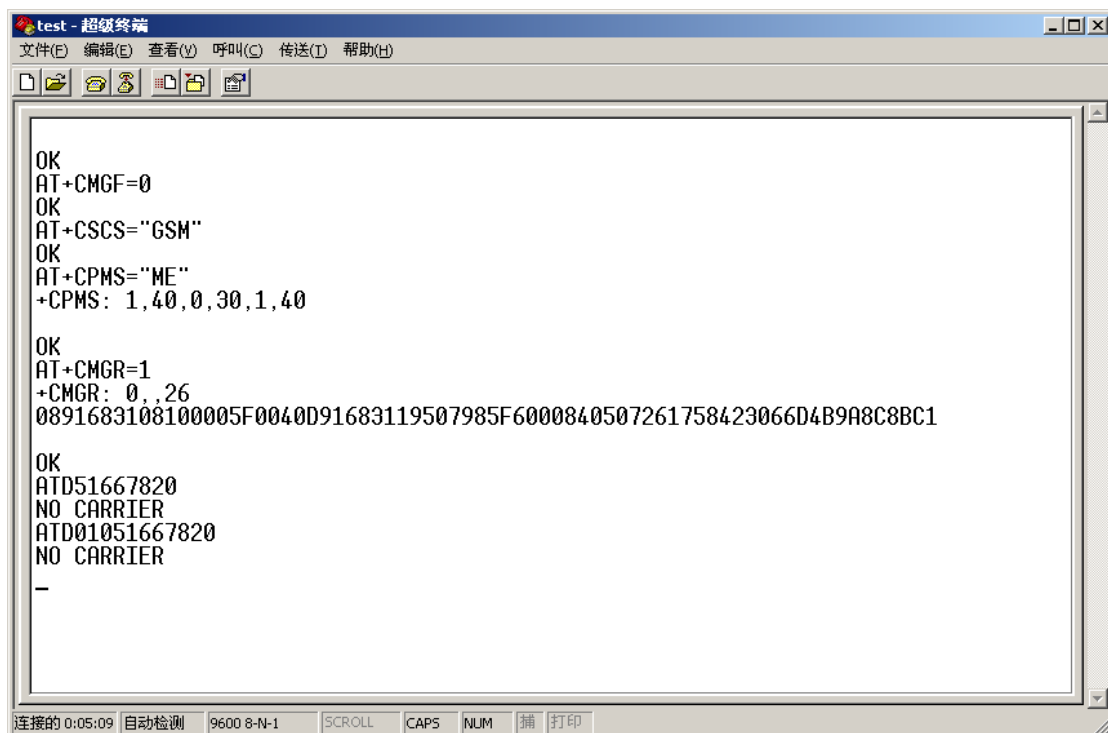


图 (8)

3) 输入指令进行测试:

ATZ //初始化

//1) 设置通讯编码

AT+CMGF=0 //pdu

AT+CSCS="GSM" //设置编码

//2) 需要先发送一条短信

AT+CPMS="ME" //指定读取短信

AT+CMGR=1 //读取一条

## 4 AT 指令集与开发参考

指令名称	用法	说明
CSMS	选择信息服务 0 兼容 GSM07.05 Phase 2 version 4.7.0 1 兼容 GSM07.05 Phase 2+ version	
	T+CSMS=0 +CSMS: 1, 1, 1 AT+CSMS=1 +CSMS: 1, 1, 1	SMS-MO SMS-MT SMS-CB (小区广播) 全部支持
CNMA	新 SMS 收到确认	

CPMS	<p>首选的 SMS 存储区 SM 为 SIM 卡区, BM 为内存区, 缺省为 SM</p> <p>AT+CPMS=? +CPMS: (("SM", "BM"), ("SM")) AT+CPMS="SM" +CPMS: 3, 25, 3, 25 AT+CPMS="BM" +CPMS: 0, 20, 3, 25</p>	<p>(1) 为读和删的信息 (2) 为写和发的信息 已用 1, 总共 1, 已用 2, 总共 2</p>
CMGF	<p>设置 SMS 编码方式 (1 文本格式, 0 为 PDU 格式)</p>	
CSAS	<p>存储 CSCA 与 CSMP 的相关信息于 E<sup>2</sup>PROM</p>	
CRES	<p>从 E<sup>2</sup>PROM 中恢复相关信息</p>	
CSDH	<p>显示文本格式参数 (1 为显示, 0 为不显示)</p> <p>AT+CDSH=1</p> <p>+CMT: "+8613501154105", , "01/09/12, 18:04:09+32", 145, 4, 0, 0, "+8613800100500" , 145, 3 AAA</p> <p>AT+CSDH=0</p> <p>+CMT: "+8613501154105", , "01/09/12, 18:04:48+32" AAA</p>	<p>相关的信息有 +CMTI, +CMT, +CDS, +CMGR, +CMGL</p> <p>左边两个例子同样是 发送和接收 AAA 为内容 的 SMS</p>
CNMI	<p>SMS 的输出方式</p> <p>AT+CNMI=2, 2, 0, 0, 0</p> <p>+CMT: "+8613501154105", , "01/09/13, 11:04:09+32" AAA</p> <p>AT+CNMI=2, 1, 0, 0, 0</p> <p>+CMTI: "SM", 4</p>	<p>其中第 2 位决定 SMS 直 接输出还是保存于 "SM"中</p> <p>左边两个例子同样是 发送和接收 AAA 为内容 的 SMS, 为 0 则不接收</p>
CMGR	<p>读取存于"SM"中的信息</p>	

	<p>AT+CMGR=1 +CMGR: "REC READ", "+8613501154102",,"01/08/14, 10:46:47+32" X000E8</p> <p>AT+CMGR=4 +CMGR: "REC UNREAD", "+8613501154105",,"01/09/13, 11:02:06+32" AAA</p> <p>AT+CMGR=2 +CMGR: "REC READ", "+8613501154105",,"01/08/23, 16:32:34+32" DCS format error at+cmgf=0;+cmgr=2 +CMGR: 1,,26 0891683108100005F0040D91683105114501F500081080326123432306004400460 044</p>	<p>读取第 1 条 来源 13501154102</p> <p>读取第 4 条 来源 13501154105</p> <p>读取第 2 条时遇到 DCS 格式错误, 转成 PDU 格 式后, 读出数据 来源 13501154105</p>
CMGL	<p>信息列表, (+CMGL: 索引, 类型,, 长度&lt;内容&gt;=</p> <p>AT+CMGL=4 +CMGL: 1, 1,, 26 0891683108100005F0040D91683105114501F20000108041016474230658180C56C 401 +CMGL: 2, 1,, 26 0891683108100005F0040D91683105114501F500081080326123432306004400460 044 +CMGL: 3, 1,, 32 0891683108100005F0040D91683105114501F50008108032617492230C003400310 032003100320033 +CMGL: 4, 1,, 23 0891683108100005F0040D91683105114501F500001090311120602303C16010</p>	<p>类型</p> <p>"REC UNREAD" 0 "REC READ" 1 "STO UNSENT" 2 "STO SENT" 3 "ALL" 4</p>
CMGS	<p>发送 SMS, 可按文本方式或 PDU 方式</p>	
	<p>AT+CMGF=1 OK AT+CMGS=13501154105 &gt; AAA&lt;^Z&gt; +CMGS: 204 OK</p> <p>AT+CMGF=0 OK AT+CMGS=18 &gt;0011000D91683105114501F500040103414141&lt;^Z&gt; +CMGS: 205 OK</p>	<p>按文本方式发送 按 PDU 方式发送</p>
CMGW	<p>写信息到存储器</p>	
	<p>AT+CMGW="+8613501154105"</p>	<p>按文本方式写入</p>

	<p>&gt; <u>AAA</u>&lt;^Z&gt;                  +CMGW: 5                  OK                  AT+CMGW=&lt;lengh&gt;&lt;CR&gt;&lt;pdu&gt;&lt;^Z&gt;</p>	按 PDU 方式写入
CMSS	<p>从存储器中发送信息</p> <p>AT+CMSS=5                  +CMSS: 207                  OK</p> <p>+CMT: ,23                  0891683108100005F0040D91683105114501F500001090316163932303C16010</p> <p>AT+CMSS=5,+8613501154102                  +CMSS: 210                  OK</p>	<p>发送存储器中的第 5 条信息</p> <p>收到信息</p> <p>也可将第 5 条信息发送到指定用户</p>
CSMP	<p>设置文本格式参数, +CSMP: &lt;fo&gt;&lt;vp&gt;&lt;pid&gt;&lt;dc&gt;</p>	
	<p><u>AT+CSMP?</u></p> <p>+CSMP: 1,167,0,0</p>	<p>其中为信息有效期, 167 默认</p> <p>0-143: (VP+1) * 5 分钟</p> <p>143-167: 12 小时 + (VP-143) * 30 分钟</p> <p>168-196: (VP-166) * 1 天</p> <p>197-255: (VP-192) * 1 周</p>
CMGD	<p>删除信息, AT+CMGD=&lt;索引&gt;</p>	
CSCA	<p>短信息中心号码</p>	
	<p><u>at+csc?</u></p> <p>+CSCA: "+8613800100500",145                  OK</p>	北京短信息中心号码
CSCB	<p>选择小区广播信息类型</p>	
WCBM	<p>小区广播信息标识符</p>	
WMSC	<p>信息状态更正</p>	
WMGO	<p>信息重写</p>	

## 5 JAVA 二次开发指南

### 5.1 总体概述

为了便于广大开发人员使用 SDK2.x 进行程序开发，JAVA 开发包提供了以下的接口类：

类名	类图	说明
GSMMultiPort	图（9）	短信模块开发接口原始类，可以直接调用各个接口，进行开发使用。
GSMModem	图（10）	继承 GSMMultiPort 类，将开发接口进行二次封装，并实现信息分解、转换、合成。减少了用户的开发量，并降低了出错率。建议用户使用此类进行短信应用的开发。
GSMMessage	图（11）	保存接收到的短信息。
COMInfo	图（12）	保存接口信息。主要是通讯端口和波特率。
GSMCommon	图（13）	信息处理过程中使用的公共方法。
TestGSMModem		开发示例。

## 5.2 GSMMultiPort 类 —— 基础短信开发接口

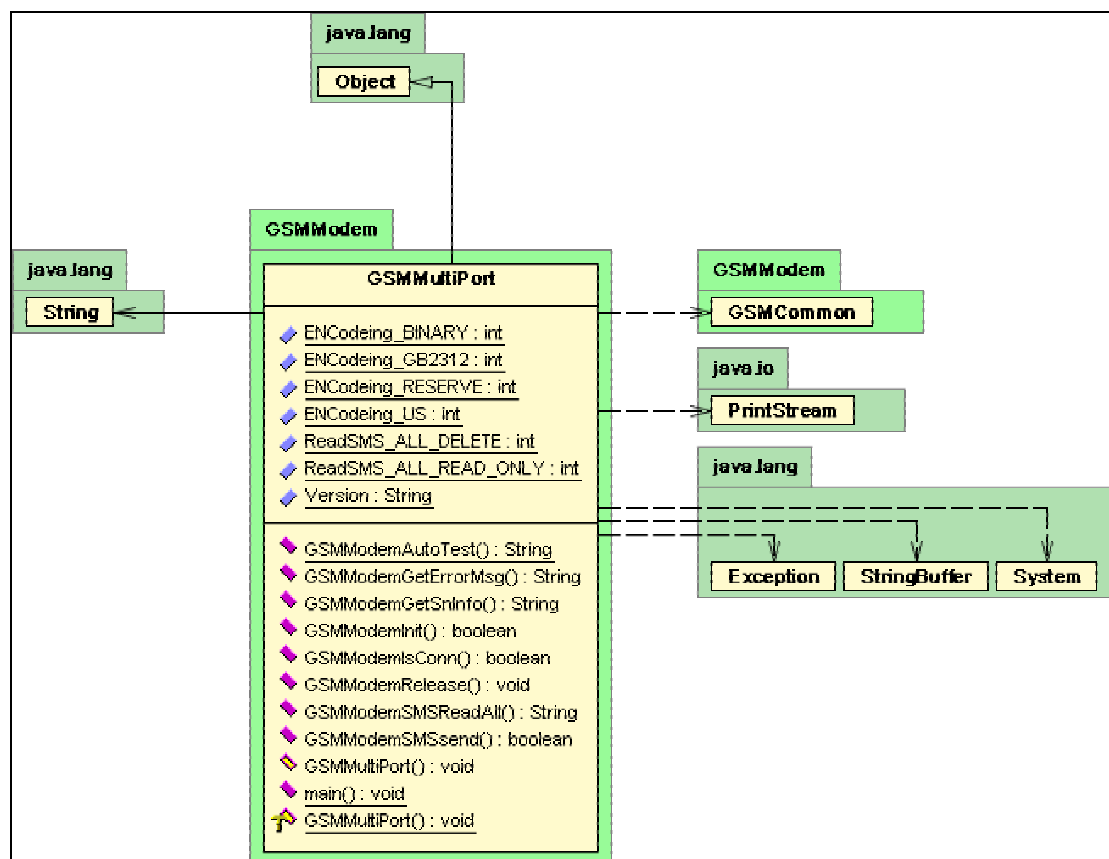


图 (9)

### 5.2.1 常量说明

参数	类型	说明
ENCoding_US	Int	等于 00，为 7bit 编码（内容为英文、ASCII）不需要处理其他编码处理；
ENCoding_BINARY	Int	等于 04，为 8bit 编码（内容为数据）不需要处理其他编码处理；
ENCoding_GB2312	Int	等于 08，为 16bit 编码（内容为 GB2312），要生成 GB2312 字符串才能正常显示；
ENCoding_RESERVE	Int	等于 0B，为保留。
ReadSMS_ALL_DELETE	Int	读完短信后，将所有短信删除。
ReadSMS_ALL_READ_ONLY	Int	读完短信后，不做任何处理。

### 5.2.2 GSMModemAutoTest —— 自动检测获取通讯信息

```
public native static String GSMModemAutoTest();
```

参数	类型	说明
返回值	String	获取所有通讯端口情况，格式如下： 端口 1 波特率 1  端口 2 波特率 2   例如： COM1 9600  COM2  19200

端口信息分解算法推荐：

```

Vector AllPortInfos = new Vector();
String ports = this.GSMModemAutoTest();
if (ports == null)
    return AllPortInfos;

char device1[] = new char[10];
char baud1[] = new char[10];
int j = 0, k = 0, sepOff = 0;
char portsStr[] = ports.toCharArray();
int portsLen = portsStr.length;

for (int i = 0; i < portsLen; i++) {
    if (portsStr[i] == '|') {
        if (sepOff < 1) {
            sepOff++;
        }
        if (sepOff == 1) {
            if ( (i > 0 && portsStr[i - 1] != '|') ||
                (i < (portsLen - 1) && portsStr[i + 1] == '|'))
                continue;

            COMInfo com1 = new COMInfo();
            //端口
            if (j > 0) {
                com1.setDevice(new String(device1, 0, j));
                j = 0;
            }
            //波特率
            if (k > 0) {
                com1.setBaudrate(new String(baud1, 0, k));
                k = 0;
            }
            AllPortInfos.addElement(com1);
            //读取下一个端口
            sepOff = 0;
        }
    }
}
continue;

```

```

    }
    if (sepOff == 0 && j<10) {
        device1[j++] = portsStr[i];
    }
    else if (sepOff == 1 && k<10) {
        baud1[k++] = portsStr[i];
    }
}
return AllPortInfos;

```

### 5.2.3 GSMModemGetSnInfo —— 获取注册信息码

```

public native String GSMModemGetSnInfo(
    String device, //端口号
    String baudrate); //波特率

```

参数	类型	说明
device	String	通讯端口，可用自动检测获得或直接指定。
baudrate	String	通讯波特率，可用自动检测获得或直接指定。
返回值	String	短信标识码，将此号码发送给厂商即可获得正式的授权码。

### 5.2.4 GSMModemInit —— 初始化短信设备

```

public native boolean GSMModemInit(
    String device, //端口号
    String baudrate, //波特率
    String initstring, //at 初始化命令
    String charset, //与 GSM Modem 通讯的字符集 GSM
    boolean swHandshake, //软件握手
    String sn); //通讯许可证书

```

参数	类型	说明
Device	String	通讯端口，可用自动检测获得或直接指定。
baudrate	String	通讯波特率，可用自动检测获得。
initstring	String	at 初始化命令，设为 null，系统默认即可。
charset	String	通讯字符集，设为 null，系统默认即可。
swHandshake	boolean	是否进行软件握手，设为 false 即可
Sn	String	通讯许可证书，区分大小写。例如： “REEE-IVKD-VKTZ-VDZB”
返回值	boolean	True 为成功，false 连接失败

## 5.2.5 GSMModemSMSsend —— 发送短信息

```
public native boolean GSMModemSMSsend(
    String device,           //端口号
    String serviceCenterAddress, //短信中心号码
    int codeval,           //文本编码格式,0-7bit;4-8bit,8-16bit
    String content,        //发送文本
    String phonenumber,    //电话号码
    boolean requestStatusReport); //状态报告
```

参数	类型	说明
device	String	通讯端口，可用自动检测获得或直接指定。
serviceCenterAddress	String	短信中心号码
codeval	int	文本编码格式,0-7bit;4-8bit,8-16bit
content	String	短信内容的 UNICODE 字节数组，
phonenumber	String	接收电话号码。
requestStatusReport	boolean	状态报告，一般不进行状态报告。
返回值	boolean	True 发送成功，false 发送失败

## 5.2.6 GSMModemSMSReadAll —— 读取短信

```
public native String GSMModemSMSReadAll(
    String device,           //端口号
    int selectOper); //对短信息的处理，0-删除，1-不做处理
```

参数	类型	说明
device	String	通讯端口，可用自动检测获得或直接指定。
selectOper	int	对读取后短信息处理，0-删除，1-不做处理
返回值	String	取得所有短信息包括，SIM 卡和手机中的。格式如下： 短信类型 存储位置 发送时间 接收号码 短信编码 短信长度 短信内容 短信类型 存储位置 发送时间 接收号码 短信编码 短信长度 短信内容  多条短信以” ”进行分隔，每条短信中各项以“ ”进行分隔。

短信内容介绍：

名称	描述与值
短信类型	取得的短信类型包括： 0 ——接收到的短信（位于收件箱中） 1 ——发送短信（位于发件箱或草稿箱中） 2 ——短信息发送状态报告（在发送短信时，可以要求回执短信息发送状态报告，即短信到达接收方手机的时间）
存储位置	短信息来自的地方，可能为： SM: sim 卡的短信存储区；

	<b>BM:</b> 手机内存存储卡区 <b>ME:</b> 手机的短信存储区 <b>SR:</b> 短信发送状态报告存储区
发送时间	发送时间根据短信类型的不同，有以下不同意义： 短信类型 = 0 时，表示发信方发送的短信时间； 短信类型 = 1 时，表示编辑此短信的时间； 短信类型 = 2 时，表示接收方接收到短信时间。
手机号码	接收号码根据短信类型的不同，有以下不同意义： 短信类型 = 0 时，表示发信方发送的手机号码； 短信类型 = 1 时，表示接收方的手机号码； 短信类型 = 2 时，表示接收方的手机号码
短信编码	短信息内容的编码，有以下几种情况： 00 —— 为 7bit 编码（内容为英文、ASCII）不需要处理其他编码处理； 04 —— 为 8bit 编码（内容为数据）不需要处理其他编码处理； 08 —— 为 16bit 编码（内容为 GB2312），要生成 GB2312 字串才能正常显示； 0B —— 为保留
短信长度	短信息的长度，读取短信时以此长度进行读取短信内容。
短信内容	短信息的内容，当短信长度为 0 时，短信内容为空。

短信内容分解算法推荐：

```
String smscontent = this.GSMModemSMSReadAll(this.device, selectOper);
Vector AllMsg = new Vector();
if (smscontent == null)
    return AllMsg;

char smstype[] = new char[4];
char storename[] = new char[10];
char sendtime[] = new char[50];
char number1[] = new char[30];
char smscode[] = new char[10];
char smslenStr[] = new char[10];
char smscontent1[] = new char[512];

int j = 0, k = 0, l = 0, m = 0, n = 0, o = 0, p = 0, sepOff = 0;
char sms_msg[] = smscontent.toCharArray();
int AllSMSLen = sms_msg.length;
int smslen = -1;

for (int i = 0; i < AllSMSLen; i++) {
    if (sms_msg[i] == '|') {
        if (sepOff < 6) {
            sepOff++;
        }
    }
}
```

```
if (sepOff == 6 && smslen == -1) {
    try {
        smslen = Integer.parseInt(new String(smslenStr, 0, o));
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
else if (sepOff == 6 && (p == smslen)) {
    // 排出此种情况: 2|SM|08/26/05 08:35:10 (+0800)|8613910597586|00|0||
    if ( (i > 0 && sms_msg[i - 1] != '|') ||
        (i < (AllSMSLen - 1) && sms_msg[i + 1] == '|'))
        continue;

    GSMMMessage gsmmsg1 = new GSMMMessage();
    //保存短信类型
    if (j > 0) {
        try {
            int val1 = Integer.parseInt(new String(smstype, 0, j));
            gsmmsg1.setType(val1);
        }
        catch (Exception e1) {}
        j = 0;
    }
    //保存存储位置
    if (k > 0) {
        gsmmsg1.setStoreName(new String(storename, 0, k));
        k = 0;
    }
    //保存发送时间
    if (l > 0) {
        gsmmsg1.setSendTime(new String(sendtime, 0, l));
        l = 0;
    }
    //保存接收号码
    if (m > 0) {
        gsmmsg1.setNumber(new String(number1, 0, m));
        m = 0;
    }
    //保存短信编码
    if (n > 0) {
        try {
            int val1 = Integer.parseInt(new String(smscode, 0, n));
            gsmmsg1.setEncoding(val1);
```

```
    }
    catch (Exception e1) {}
    n = 0;
}
//保存短信长度
if(o > 0) {
    try {
        int val = Integer.parseInt(new String(smslenStr, 0, o));
        gsmmsg1.setLength(val);
    }
    catch (Exception e1) {}
    o = 0;
}
//保存短信内容
if(p > 0) {
    String msg1 = new String(smscontent1, 0, p);
    gsmmsg1.setContent(GSMCommon.HexToBuf(msg1));
    p = 0;
}
AllMsg.addElement(gsmmsg1);

//初始化准备接收下一条短信
sepOff = 0;
smslen = -1;
}
continue;
}

if (sepOff == 0 && j<4) { //保存短信类型
    smstype[j++] = sms_msg[i];
}

else if (sepOff == 1 && k<10) { //保存存储位置
    storename[k++] = sms_msg[i];
}

else if (sepOff == 2 && l<50) { //保存发送时间
    sendtime[l++] = sms_msg[i];
}

else if (sepOff == 3 && m<30) { //保存手机号码
    number1[m++] = sms_msg[i];
}
}
```

```

else if (sepOff == 4 && n<10) { //保存短信编码
    smscode[n++] = sms_msg[i];
}
else if (sepOff == 5 && o<10) { //保存短信长度
    smslenStr[o++] = sms_msg[i];
}
else if (sepOff == 6 && p<512) { //保存短信内容
    smscontent1[p++] = sms_msg[i];
}
}
}

```

### 5.2.7 GSMModemGetErrorMsg —— 获取过程错误信息

```
public native String GSMModemGetErrorMsg(String device);
```

参数	类型	说明
device	String	通讯端口，可用自动检测获得或直接指定。
返回值	String	错误说明文字

### 5.2.8 GSMModemIsConn —— 获取当前连接状态

```
public native boolean GSMModemIsConn(String device);
```

参数	类型	说明
device	String	通讯端口，可用自动检测获得或直接指定。
返回值	boolean	系统是否连接，true 正在连接，false 未连接

### 5.2.9 GSMModemRelease —— 断开连接，并释放资源

```
public native void GSMModemRelease(String device);
```

参数	类型	说明
device	String	通讯端口，可用自动检测获得或直接指定。
返回值	void	

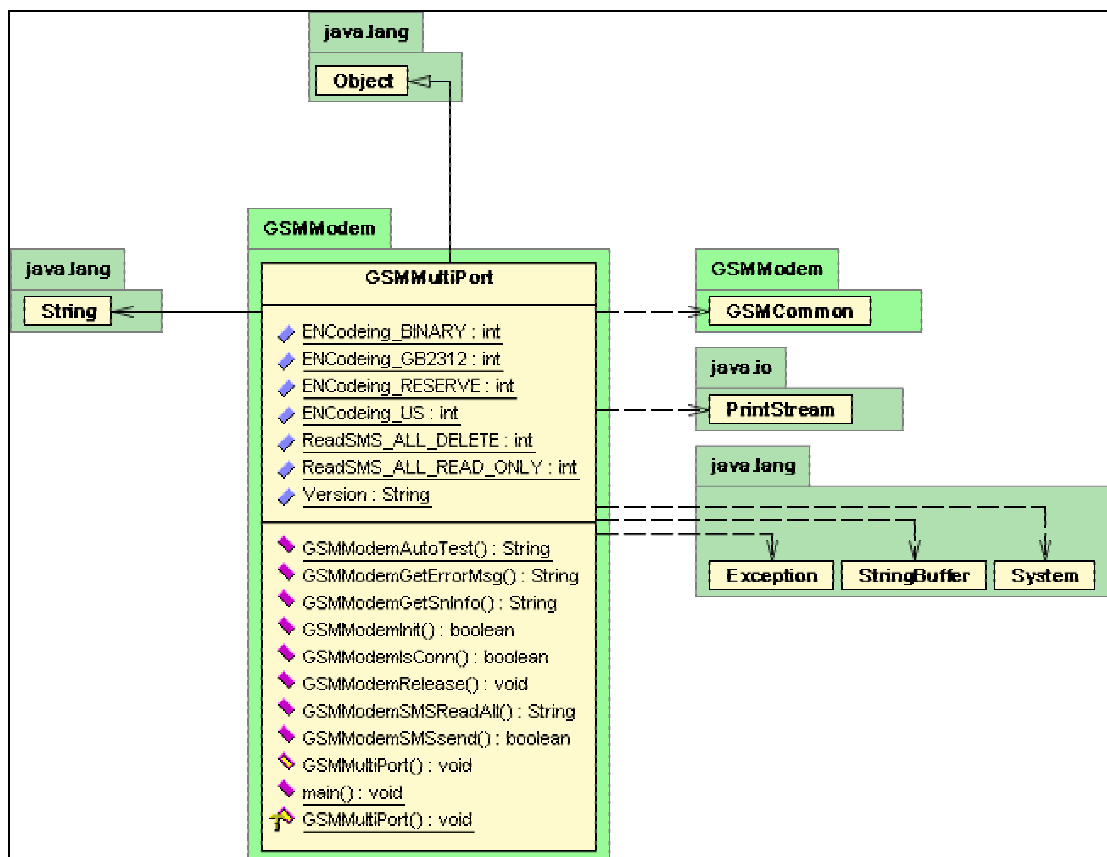
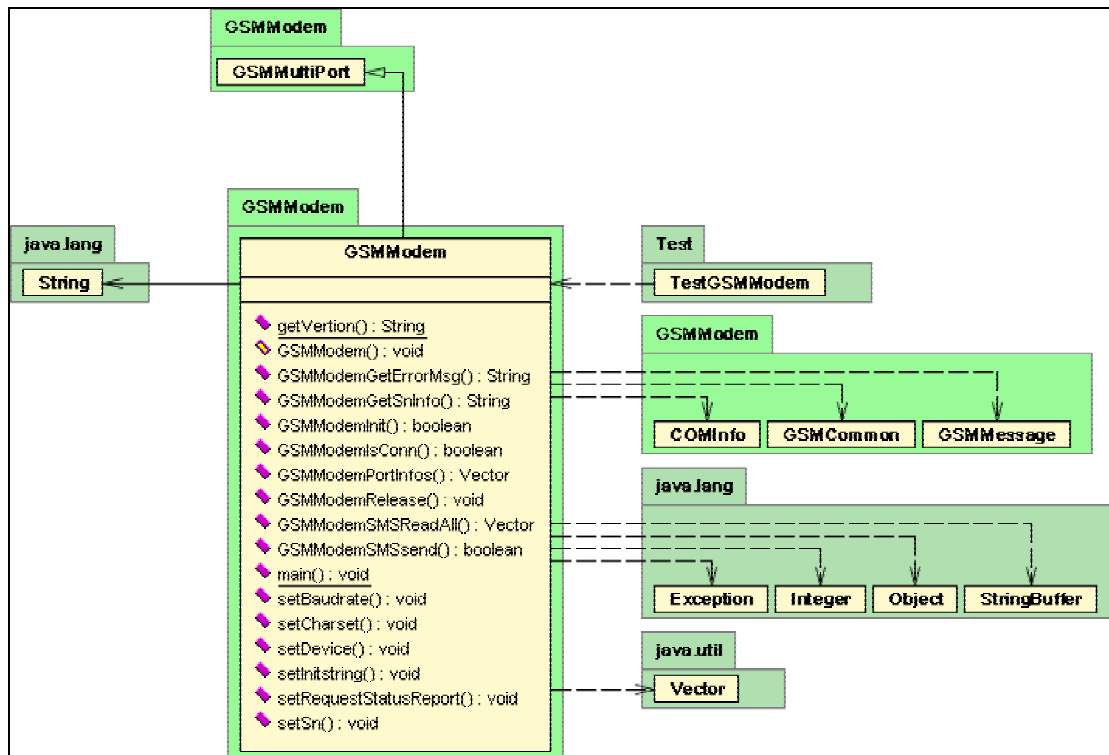


图 (10)

### 5.3 GSMModem 类 —— 接口应用类



#### 5.3.1 属性

属性	类型	说明
device	String	设备通讯端口（串口），通讯前必须指定。
baudrate	String	设备通讯波特率，通讯前必须指定。
requestStatusReport	Boolean	发送短信是否需要短信发送状态报告，默认为 false。
charset	String	与设备通讯的字符集，默认为“GSM”。
sn	String	指定端口上的通讯设备，对应的授权注册码。

#### 5.3.2 GSMModemPortInfos —— 自动获取所有通讯信息

```
public native static String GSMModemPortInfos ();
```

参数	类型	说明
返回值	Vector	获取所有通讯端口情况。值为 COMInfo 对象的列表。

### 5.3.3 GSMModemGetSnInfo —— 获取注册信息码

```
public native String GSMModemGetSnInfo();
```

参数	类型	说明
返回值	String	短信标识码，将此号码发送给厂商即可获得正式的授权码。

### 5.3.4 GSMModemInit —— 初始化短信设备

```
public native boolean GSMModemInit();
```

参数	类型	说明
返回值	boolean	True 为成功，false 连接失败

### 5.3.5 GSMModemSMSsend —— 发送短信息

```
public native boolean GSMModemSMSsend(
    String serviceCenterAddress, //短信中心号码
    int codeval, //文本编码格式,0-7bit;4-8bit,8-16bit
    String content, //发送文本
    String phonenumber, //电话号码
    boolean requestStatusReport); //状态报告
```

参数	类型	说明
serviceCenterAddress	String	短信中心号码
codeval	int	文本编码格式,为下列值: <ul style="list-style-type: none"> <li>➤ ENCodeing_US</li> <li>➤ ENCodeing_BINARY</li> <li>➤ ENCodeing_GB2312</li> </ul>
content	String	短信内容的 UNICODE 字节数组，
phonenumber	String	接收电话号码。
requestStatusReport	boolean	状态报告，一般不进行状态报告。
返回值	boolean	True 发送成功，false 发送失败

### 5.3.6 GSMModemSMSReadAll —— 读取短信

```
public native String GSMModemSMSReadAll(
    int selectOper);
```

参数	类型	说明
selectOper	int	对读取后短信息处理，为下列值: <ul style="list-style-type: none"> <li>➤ ReadSMS_ALL_DELETE</li> <li>➤ ReadSMS_ALL_READ_ONLY</li> </ul>

返回值	Vector	取得所有短信息包括，SIM 卡和手机中的。 值为 GSMMessage 对象的列表，保存着所有读取后的短信。
-----	--------	---

### 5.3.7 GSMModemGetErrorMsg —— 获取过程错误信息

```
public native String GSMModemGetErrorMsg();
```

参数	类型	说明
返回值	String	错误说明文字

### 5.3.8 GSMModemIsConn —— 获取当前连接状态

```
public native boolean GSMModemIsConn();
```

参数	类型	说明
返回值	boolean	系统是否连接，true 正在连接，false 未连接

### 5.3.9 GSMModemRelease —— 断开连接，并释放资源

```
public native void GSMModemRelease();
```

参数	类型	说明
返回值	void	

## 5.4 GSMMessage 类 —— 短信存储类（略）

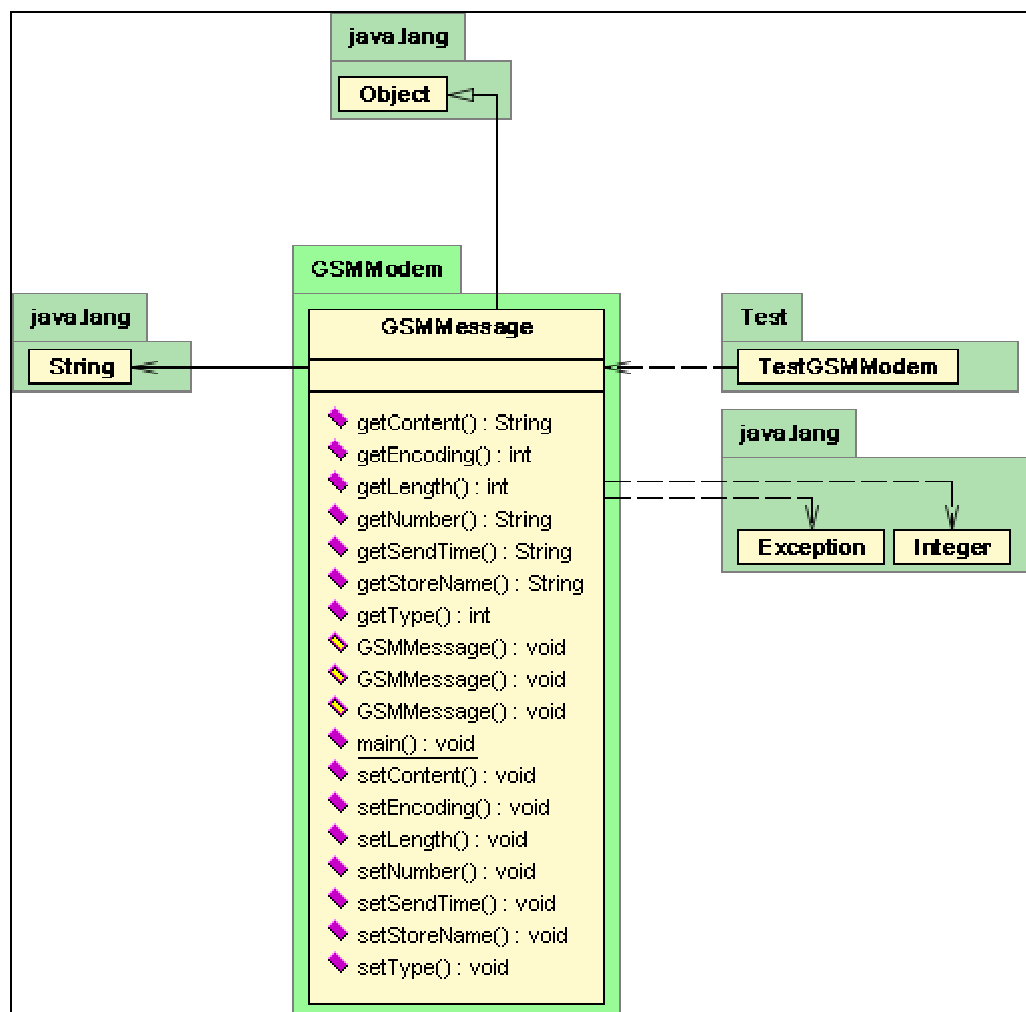


图 (11)

### 5.5 COMInfo 类 —— 通讯端口信息类（略）

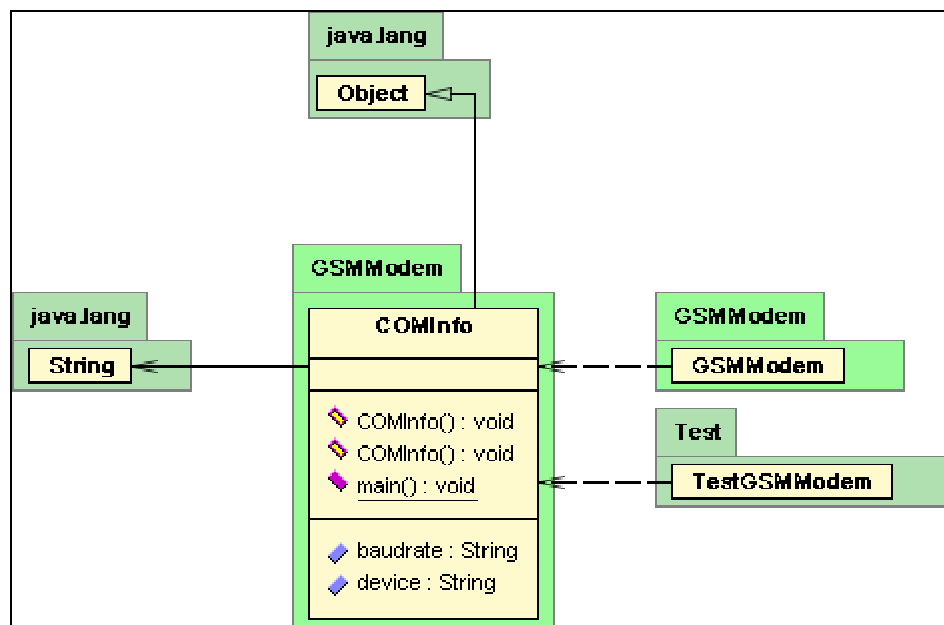


图 (12)

### 5.6 GSMCommon 类 —— 公共方法类（略）

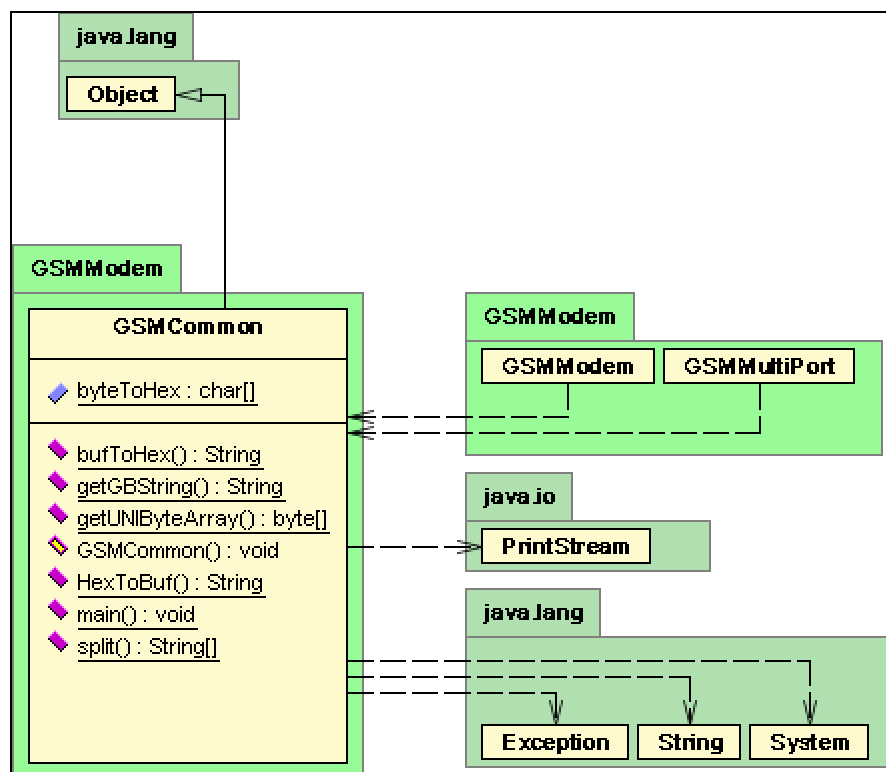


图 (13)

## 5.7 演示代码

见《DemoSRC\TestForJava》

## 5.8 开发包说明

TestSDK2\_For\_J.zip 为在 jbuilder9 下的测试工程。

其中：

- .\TestGSModem.java ——为测试类
- .\GSMMultiPort.dll ——为 GSModem.jar 使用的通讯动态库。
- .\GSMMultiPortForJ.dll ——为 GSModem.jar 使用的通讯动态库。
- .\lib\GSModem.jar ——为封装的 java 类库。用户直接 import 即可。

注意：

- ◆ 在使用时需要给 JAVA 虚拟机指定 DLL 的所在的路径。  
-Djava.library.path=".;K:\umail\_src\GSModem"
- ◆ 在开发时，用户只需连接设备一次，可以进行若干发送和接收操作；

## 6 VC/C 二次开发指南

### 6.1 函数说明

#### 6.1.1 GSModemAutoTest —— 自动检测获取通讯信息

GSMMULTIPOINT\_API char\* GSModemAutoTest();

参数	类型	说明
返回值	char*	获取所有通讯端口情况，格式如下： 端口 1 波特率 1  端口 2 波特率 2   例如：COM1 9600  COM2 19200

#### 6.1.2 GSModemGetSnInfo —— 获取注册信息码

GSMMULTIPOINT\_API char \* GSModemGetSnInfo(char \*device, char \*baudrate);

参数	类型	说明
device	char *	通讯端口，可用自动检测获得或直接指定。
baudrate	char *	通讯波特率，可用自动检测获得或直接指定。
返回值	char *	短信标识码，将此号码发送给厂商即可获得正式的授权码。

### 6.1.3 GSMModemInit —— 初始化短信设备

```

GSMMULTIPOINT_API bool GSMModemInit(
    char *device,
    char *baudrate,
    char *initstring,
    char *charset,
    bool swHandshake,
    char *sn);
    
```

参数	类型	说明
Device	char *	通讯端口，可用自动检测获得或直接指定。
baudrate	char *	通讯波特率，可用自动检测获得。
initstring	char *	at 初始化命令，设为 null，系统默认即可。
charset	char *	通讯字符集，设为 null，系统默认即可。
swHandshake	bool	是否进行软件握手，设为 false 即可
Sn	char *	通讯许可证书，区分大小写。例如：“REEE-IVKD-VKTZ-VDZB”
返回值	bool	True 为成功，false 连接失败

### 6.1.4 GSMModemSMSsend —— 发送短信息

```

GSMMULTIPOINT_API bool GSMModemSMSsend(
    char *device,
    char *serviceCenterAddress,
    int encodeval,
    char *text,
    int textlen,
    char *phonenumber,
    bool requestStatusReport);
    
```

参数	类型	说明
device	Char *	通讯端口，可用自动检测获得或直接指定。
serviceCenterAddress	Char *	短信中心号码
codeval	int	文本编码格式,0-7bit;4-8bit,8-16bit
content	char *	短信内容的 UNICODE 字节数组，
phonenumber	char *	接收电话号码。
requestStatusReport	bool	状态报告，一般不进行状态报告。
返回值	bool	True 发送成功，false 发送失败

### 6.1.5 GSMModemSMSReadAll —— 读取短信

```

GSMMULTIPOINT_API char* GSMModemSMSReadAll(
    
```

```
char *device,
int RD_opt);
```

参数	类型	说明
device	char *	通讯端口，可用自动检测获得或直接指定。
RD_opt	int	对读取后短信息处理，0-删除，1-不做处理
返回值	char *	取得所有短信息包括，SIM 卡和手机中的。格式如下：短信类型 存储位置 发送时间 接收号码 短信编码 短信长度 短信内容 短信类型 存储位置 发送时间 接收号码 短信编码 短信长度 短信内容 多条短信以” ”进行分隔，每条短信中各项以“ ”进行分隔。

短信内容介绍：

名称	描述与值
短信类型	取得的短信类型包括： 0 ——接收到的短信（位于收件箱中） 1 ——发送短信（位于发件箱或草稿箱中） 2 ——短信息发送状态报告（在发送短信时，可以要求回执短信息发送状态报告，即短信到达接收方手机的时间）
存储位置	短信息来自的地方，可能为： SM: sim 卡的短信存储区； BM: 手机内存存储卡区 ME: 手机的短信存储区 SR: 短信发送状态报告存储区
发送时间	发送时间根据短信类型的不同，有以下不同意义： 短信类型 = 0 时，表示发信方发送的短信时间； 短信类型 = 1 时，表示编辑此短信的时间； 短信类型 = 2 时，表示接收方接收到短信时间。
手机号码	接收号码根据短信类型的不同，有以下不同意义： 短信类型 = 0 时，表示发信方发送的手机号码； 短信类型 = 1 时，表示接收方的手机号码； 短信类型 = 2 时，表示接收方的手机号码
短信编码	短信息内容的编码，有以下几种情况： 00 —— 为 7bit 编码（内容为英文、ASCII）不需要处理其他编码处理； 04 —— 为 8bit 编码（内容为数据）不需要处理其他编码处理； 08 —— 为 16bit 编码（内容为 GB2312），要生成 GB2312 字串才能正常显示； 0B —— 为保留
短信长度	短信息的长度，读取短信时以此长度进行读取短信内容。
短信内容	短信息的内容，当短信长度为 0 时，短信内容为空。

短信内容分解算法推荐：

```
/**
*
```

```

* 3.4) 分解短信, 短信息格式如下
*
* 短信类型|存储位置|发送时间|接收号码|短信编码|短信长度|短信内容||短信类型|存储位置|发送时
间|接收号码|短信编码|短信长度|短信内容||
*
*/
char smscode[10], smslenStr[10], smscontent1[512]; //保存手机号码和短信内容
char smstype[2], storename[10], sendtime[50], number1[30];
int j=0, k=0, l=0, m=0, n=0, o=0, p=0, sepOff = 0;
int count=0;
//接受到的短信数量

memset(smstype, 0, 2);
memset(storename, 0, 10);
memset(sendtime, 0, 50);
memset(number1, 0, 30);
memset(smscode, 0, 10);
memset(smslenStr, 0, 10);
memset(smscontent1, 0, 512);
int smslen = -1;

for(int i = 0; i<AllSMSLen; i++){
    if(sms_msg[i]=='|'){
        if(sepOff<6){
            sepOff++;
        }

        if(sepOff==6 && smslen== -1){
            smslen = atoi(smslenStr);

        }else if( sepOff==6 && (p==smslen)){

            /**
             * 排出此种情况: 2|SM|08/26/05 08:35:10 (+0800)|8613910597586|00|0|||
             *
             */
            if( (i>0 && sms_msg[i-1]!='|') || (i<(AllSMSLen-1) &&
sms_msg[i+1]=='|') )

                continue;

            //输出每条短信息具体内容
            //保存完成
            printf("\n短信#%d:-----\n", count);
            printf("\t短信类型=%s\n", smstype);

```

```
printf("\t存储位置=%s\n", storename);
printf("\t发送时间=%s\n", sendtime);
printf("\t电话号码=%s\n", number1);
printf("\t短信编码=%s\n", smscode);
printf("\t短信长度=%s\n", smslenStr);
printf("\t短信内容=%s\n", smscontent1);
printf("短信#%d:-----\n", count);
count++;

//初始化准备接收下一条短信
j=0;
k=0;
l=0;
m=0;
n=0;
o=0;
p=0;
sep0ff=0;
smslen=-1;

memset(smstype, 0, 2);
memset(storename, 0, 10);
memset(sendtime, 0, 50);
memset(number1, 0, 30);
memset(smscode, 0, 10);
memset(smslenStr, 0, 10);
memset(smscontent1, 0, 512);
}
continue;
}

if(sep0ff==0 && j<2) { //保存短信类型
    smstype[j++] = sms_msg[i];

} else if(sep0ff==1 && k<10) { //保存存储位置
    storename[k++] = sms_msg[i];

} else if(sep0ff==2 && l<50) { //保存发送时间
    sendtime[l++] = sms_msg[i];

} else if(sep0ff==3 && m<30) { //保存手机号码
    number1[m++] = sms_msg[i];

} else if(sep0ff==4 && n<10) { //保存短信编码
```

```

        smscode[n++] = sms_msg[i];

    }else if(sepOff==5 && o<10){           //保存短信长度
        smslenStr[o++] = sms_msg[i];

    }else if(sepOff==6 && p<smslen){       //保存短信内容
        smscontent1[p++] = sms_msg[i];
    }
}

```

### 6.1.6 GSMModemGetErrorMsg —— 获取过程错误信息

GSMMULTIPOINT\_API char \*GSMModemGetErrorMsg(char \*device);

参数	类型	说明
device	char *	通讯端口，可用自动检测获得或直接指定。
返回值	char *	错误说明文字

### 6.1.7 GSMModemIsConn —— 获取当前连接状态

GSMMULTIPOINT\_API bool GSMModemIsConn(char \*device);

参数	类型	说明
device	char *	通讯端口，可用自动检测获得或直接指定。
返回值	bool	系统是否连接，true 正在连接，false 未连接

### 6.1.8 GSMModemRelease —— 断开连接，并释放资源

GSMMULTIPOINT\_API void GSMModemRelease(char \*device);

参数	类型	说明
device	char *	通讯端口，可用自动检测获得或直接指定。
返回值	void	

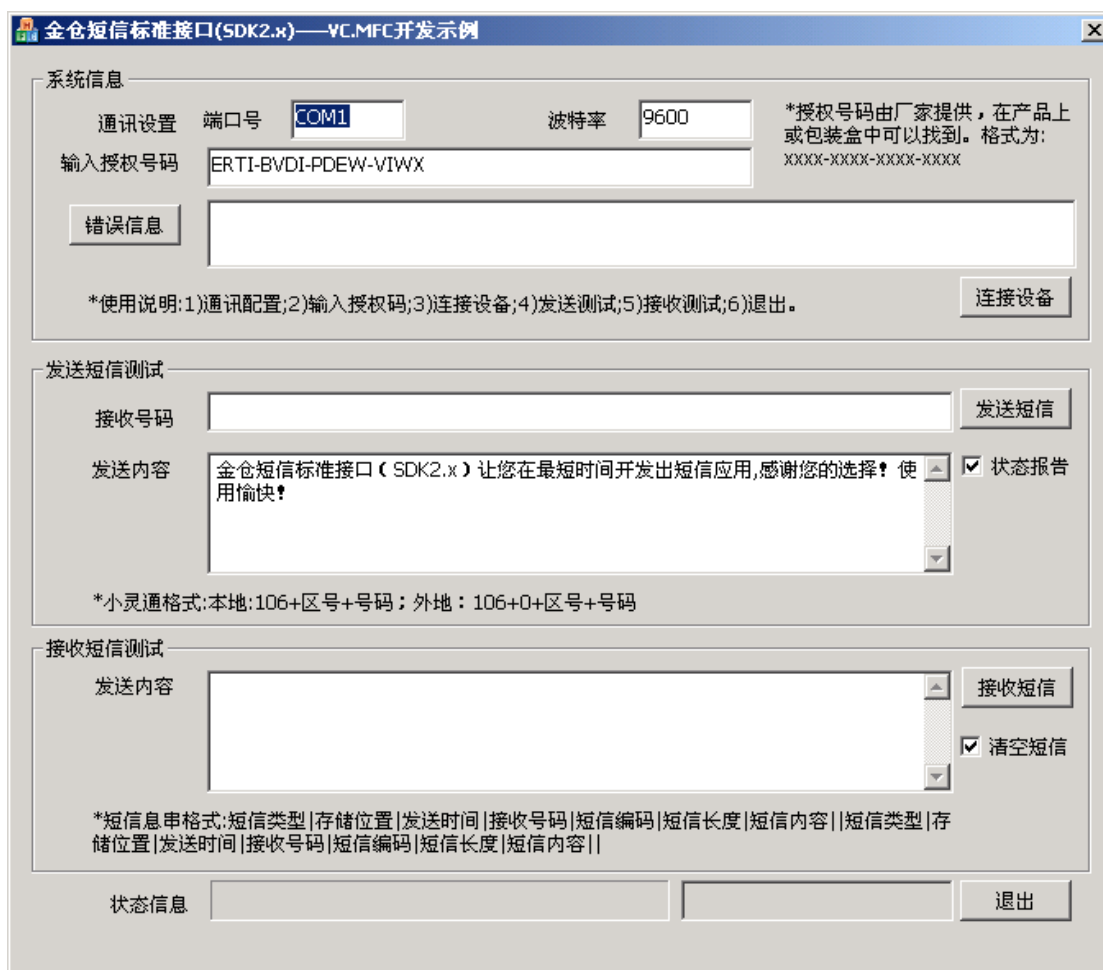
## 6.2 演示代码

函数的具体使用见演示代码。

《DemoSRC\ TestForVC.mfc》

《DemoSRC\ TestForVC.WIN32》

示例是 VC7.0 下工程打开可以直接使用，界面如下：



## 6.3 开发包说明

SDK 库在《Libs\VC\_VB\_DELPHI》中，文件说明：

- ◆ \GSMMultiPort.dll ——GSMModem 的动态开发包。
- ◆ \GSMMultiPort.lib ——为静态加载包，直接添加到工程的附加依赖项中
- ◆ \GSMMultiPort.h ——为头文件。

**注意：**

- ◆ 在用 C 动态连接库时，同样也要指定正确的路径。有几种方式，
  - 1) 动态库与应用程序放在同一个目录中。
  - 2) 或将路径加入系统环境变量 PATH 中
  - 3) 或将其拷贝到系统的 system32 下
- ◆ 在开发时，用户只需连接设备一次，可以进行若干发送和接收操作；

## 7 C#二次开发指南

### 7.1 函数声明

函数说明参照 VC 部分说明部分，执行程序时将 GSMMultiPort.dll 拷贝到执行目录下即可。

```
//初始化gsm modem, 并连接gsm modem
[DllImport("GSMMultiPort.dll",
    EntryPoint="GSMModemInit",
    CharSet=CharSet.Ansi,
    CallingConvention=CallingConvention.StdCall)]
public static extern bool GSMModemInit(
    string device,
    string baudrate,
    string initstring,
    string charset,
    bool swHandshake,
    string sn);

//获取短信猫标识号码
[DllImport("GSMMultiPort.dll",
    EntryPoint="GSMModemGetSnInfo",
    CharSet=CharSet.Ansi,
    CallingConvention=CallingConvention.StdCall)]
public static extern string GSMModemGetSnInfo(string device, string baudrate);

//获取设备的连接状态
[DllImport("GSMMultiPort.dll",
    EntryPoint="GSMModemIsConn",
    CharSet=CharSet.Ansi,
    CallingConvention=CallingConvention.StdCall)]
public static extern bool GSMModemIsConn(string device);

//断开连接并释放内存空间
[DllImport("GSMMultiPort.dll",
    EntryPoint="GSMModemRelease",
    CharSet=CharSet.Ansi,
    CallingConvention=CallingConvention.StdCall)]
public static extern void GSMModemRelease(string device);

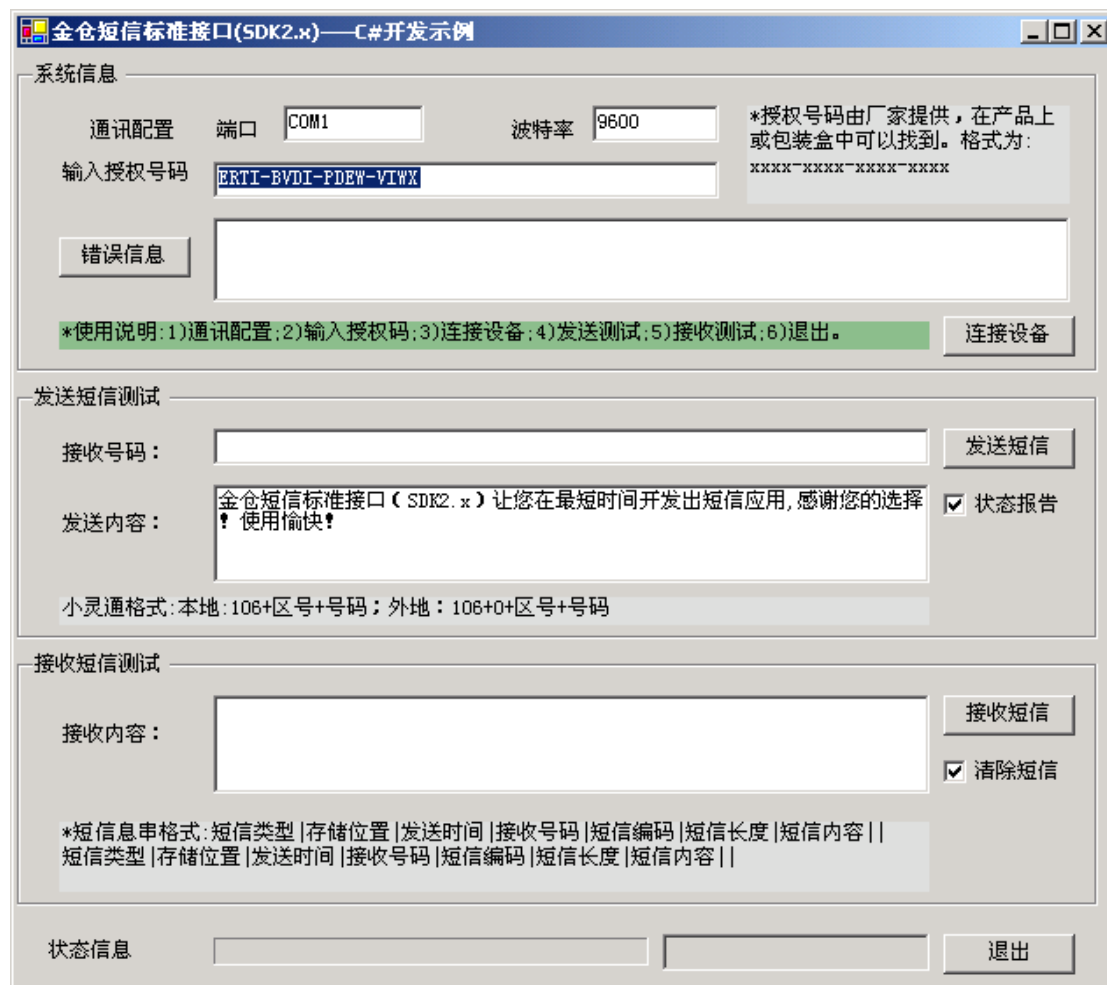
//取得错误信息
[DllImport("GSMMultiPort.dll",
    EntryPoint="GSMModemGetErrorMsg",
```

```
        CharSet=CharSet.Ansi,
        CallingConvention=CallingConvention.StdCall)]
public static extern string GSMModemGetErrorMsg(string device);

//发送短信息
[DllImport("GSMMultiPort.dll",
    EntryPoint="GSMModemSMSsend",
    CharSet=CharSet.Ansi,
    CallingConvention=CallingConvention.StdCall)]
public static extern bool GSMModemSMSsend(
    string device,
    string serviceCenterAddress,
    int encodeval,
    string text,
    int textlen,
    string phonenumber,
    bool requestStatusReport);

//接收短信息
[DllImport("GSMMultiPort.dll",
    EntryPoint="GSMModemSMSReadAll",
    CharSet=CharSet.Ansi,
    CallingConvention=CallingConvention.StdCall)]
public static extern string GSMModemSMSReadAll(string device, int RD_opt);
```

## 7.2 演示界面



## 7.3 演示代码

《DemoSRC\ TestForCSharp》

## 8 VB.NET 二次开发指南

### 8.1 函数声明

函数说明同 VC 说明部分，执行程序时将 GSMMultiPort.dll 拷贝到执行目录下即可。

```
Private Declare Sub GSMModemRelease Lib "GSMMultiPort.dll" (ByVal device As String)
Private Declare Function GSMModemIsConn Lib "GSMMultiPort.dll" (ByVal device As String) As Boolean
Private Declare Function GSMModemAutoTest Lib "GSMMultiPort.dll" () As String
Private Declare Function GSMModemGetErrorMsg Lib "GSMMultiPort.dll" (ByVal device As String) As String
```

```

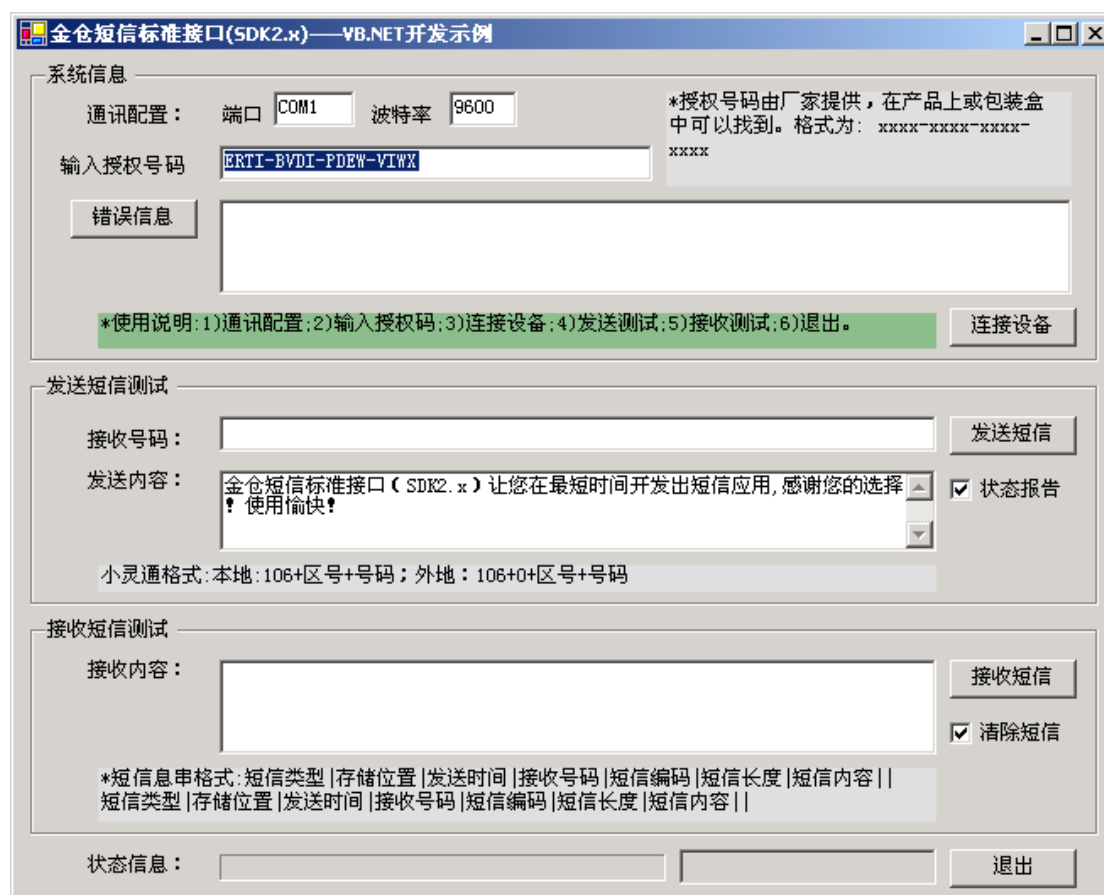
Private Declare Function GSMModemGetSnInfo Lib "GSMMultiPort.dll" (ByVal device As String, ByVal baudrate As String) As String

Private Declare Function GSMModemInit Lib "GSMMultiPort.dll" (ByVal device As String, ByVal baudrate As String, ByVal initstring As String, ByVal charset As String, ByVal swHandshake As Boolean, ByVal sn As String) As Boolean

Private Declare Function GSMModemSMSsend Lib "GSMMultiPort.dll" (ByVal device As String, ByVal serviceCenterAddress As String, ByVal encodeval As Integer, ByVal text As String, ByVal textlen As Integer, ByVal phonenummer As String, ByVal requestStatusReport As Boolean) As Boolean

Private Declare Function GSMModemSMSReadAll Lib "GSMMultiPort.dll" (ByVal device As String, ByVal RD_opt As Integer) As String
    
```

## 8.2 演示界面



## 8.3 演示代码

《DemoSRC\ TestForVB.NET》

## 9 Delphi 二次开发指南

### 9.1 函数声明

函数说明同 VC 说明部分，执行程序时将'GSMMultiPort.dll' 拷贝到执行目录下即可。  
可以直接在公用变量申请调用 DLL 声明具体定义格式如下：

```
{ 获取短信猫授权机器码}
function GSMModemAutoTest():PChar;
        stdcall; external 'GSMMultiPort.dll' name 'GSMModemAutoTest';
{ 初始化 gsm modem,并连接 gsm modem}
function GSMModemInit(device:PChar;
        baudrate:PChar;
        initstring:PChar;
        charset:PChar;
        swHandshake:Boolean;
        sn:PChar):Boolean;
        stdcall; external 'GSMMultiPort.dll' name 'GSMModemInit';

{ 获取短信猫授权机器码}
function GSMModemGetSnInfo(device:PChar;
        baudrate:PChar):PChar;
        stdcall; external 'GSMMultiPort.dll' name 'GSMModemGetSnInfo';

{ 获得错误信息}
function GSMModemGetErrorMsg(device:PChar):PChar;
        stdcall; external 'GSMMultiPort.dll' name
'GSMModemGetErrorMsg';

{ 断开连接并释放内存空间}
procedure GSMModemRelease(device:PChar)
        stdcall; external 'GSMMultiPort.dll' name 'GSMModemRelease';

{ 连接状态}
function GSMModemIsConn(device:PChar):Boolean;
        stdcall; external 'GSMMultiPort.dll' name 'GSMModemIsConn';

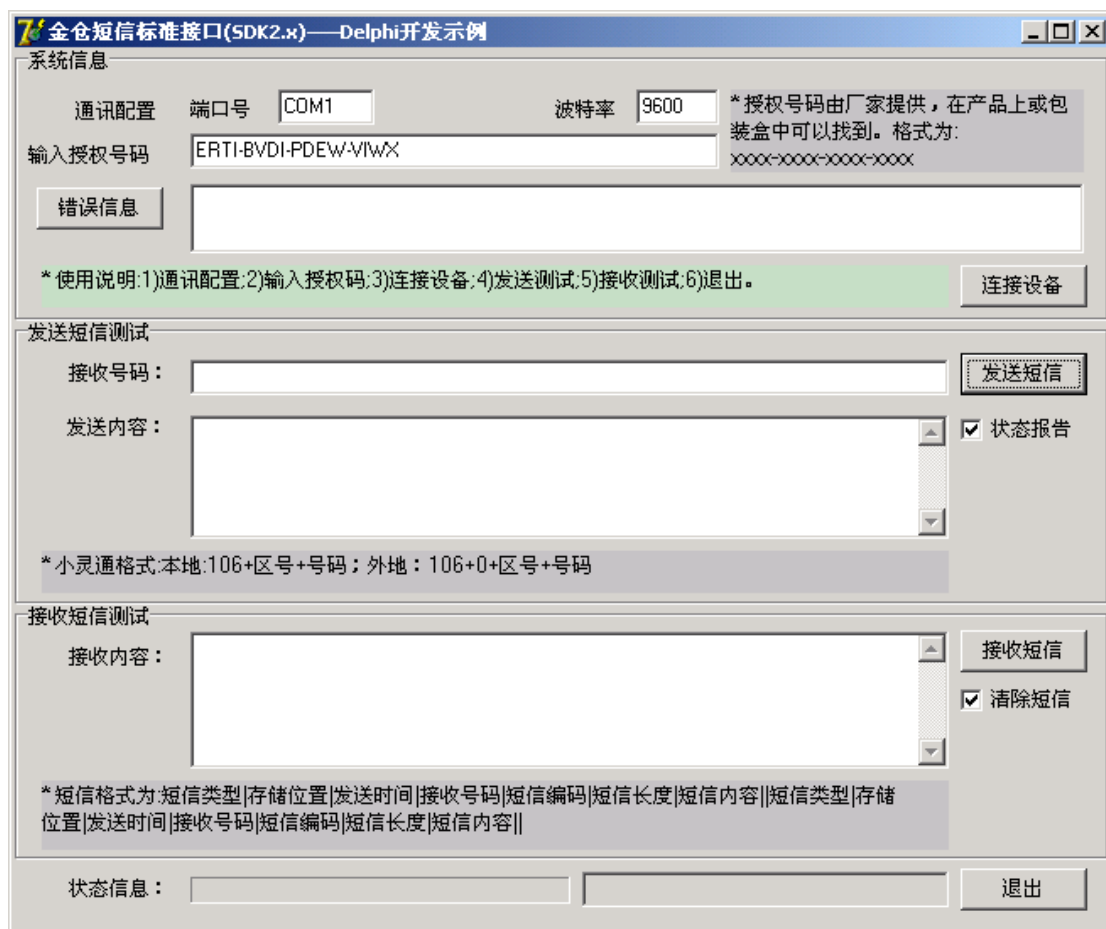
{ 发送短信息}
function GSMModemSMSsend(device:PChar;
        serviceCenterAddress:PChar;
        encodeval:Integer;
        text:PChar;
        textlen:Integer;
```

```

phonenum: PChar;
requestStatusReport: Boolean): Boolean;
stdcall; external 'GSMMultiPort.dll' name 'GSMModemSMSsend';

{ 接收短信息 }
function GSMModemSMSReadAll(device: PChar; RD_opt: Integer): PChar;
stdcall; external 'GSMMultiPort.dll' name
'GSMModemSMSReadAll';
    
```

## 9.2 演示界面



## 9.3 演示代码

《DemoSRC\ TestForDelphi》

## 10 VB6 二次开发指南

注意：接口专门为 VB6/PB 提供以下专用函数：

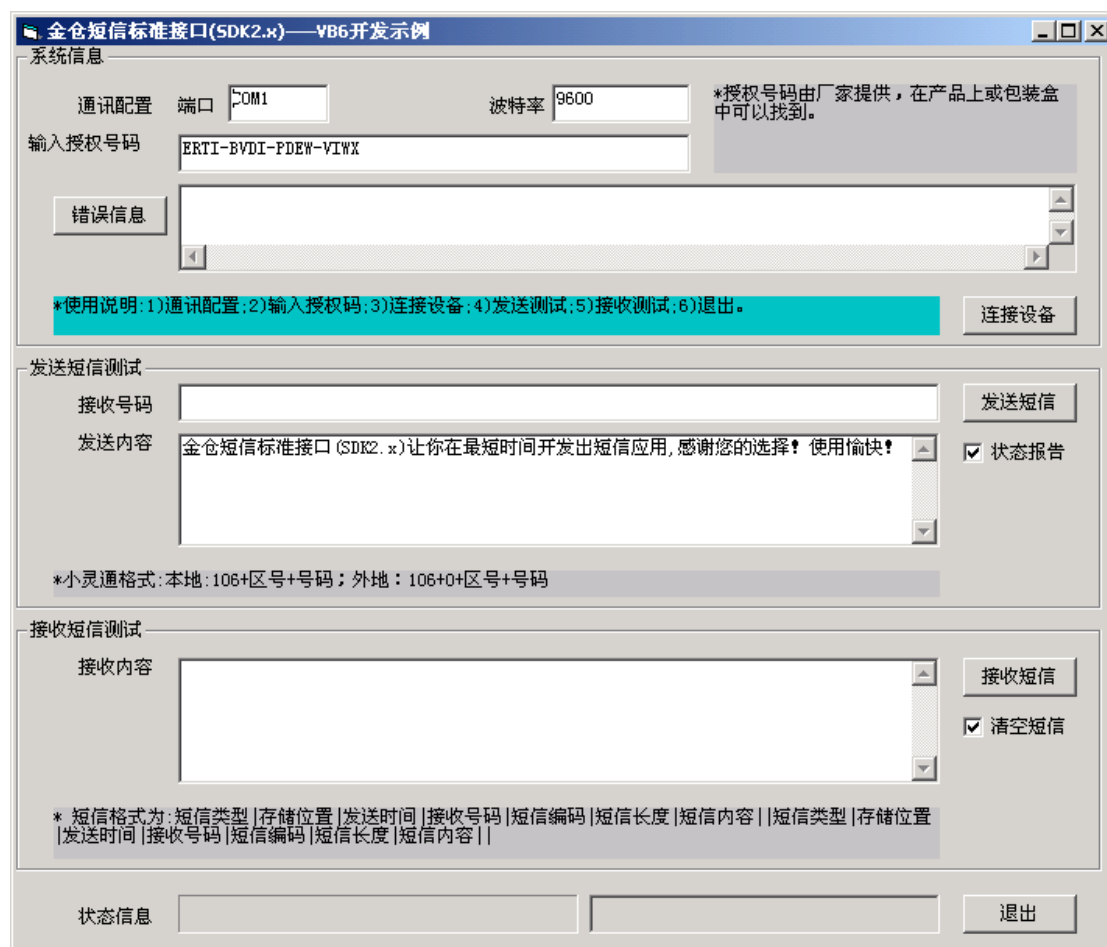
```
GSMModemAutoTestVB6  
GSMModemGetSnInfoVB6  
GSMModemGetErrorMsgVB6  
GSMModemSMSReadAllVB6
```

### 10.1 函数声明

函数说明参照 VC 说明部分，执行程序时将 GSMMultiPort.dll 拷贝到执行目录下即可。

```
Private Declare Sub GSMModemRelease Lib "GSMMultiPort.dll" (ByVal Device As String)  
Private Declare Function GSMModemIsConn Lib "GSMMultiPort.dll" (ByVal Device As String) As Boolean  
Private Declare Function GSMModemAutoTest Lib "GSMMultiPort.dll" Alias "GSMModemAutoTestVB6" () As String  
Private Declare Function GSMModemGetErrorMsg Lib "GSMMultiPort.dll" Alias "GSMModemGetErrorMsgVB6" (ByVal Device As String) As  
String  
Private Declare Function GSMModemGetSnInfo Lib "GSMMultiPort.dll" Alias "GSMModemGetSnInfoVB6" (ByVal Device As String, ByVal  
baudrate As String) As String  
Private Declare Function GSMModemInit Lib "GSMMultiPort.dll" (ByVal Device As String, ByVal baudrate As String, ByVal initstring As  
String, ByVal charset As String, ByVal swHandshake As Boolean, ByVal sn As String) As Boolean  
Private Declare Function GSMModemSMSsend Lib "GSMMultiPort.dll" (ByVal Device As String, ByVal serviceCenterAddress As String, ByVal  
encodeval As Integer, ByVal text As String, ByVal textlen As Integer, ByVal phonenummer As String, ByVal requestStatusReport As Boolean) As  
Boolean  
Private Declare Function GSMModemSMSReadAll Lib "GSMMultiPort.dll" Alias "GSMModemSMSReadAllVB6" (ByVal Device As String,  
ByVal RD_opt As Integer) As String
```

## 10.2 演示界面



## 10.3 演示代码

《DemoSRC\ TestForVB6》

## 11 PB 二次开发指南

注意：接口专门为 VB6/PB 提供以下专用函数：

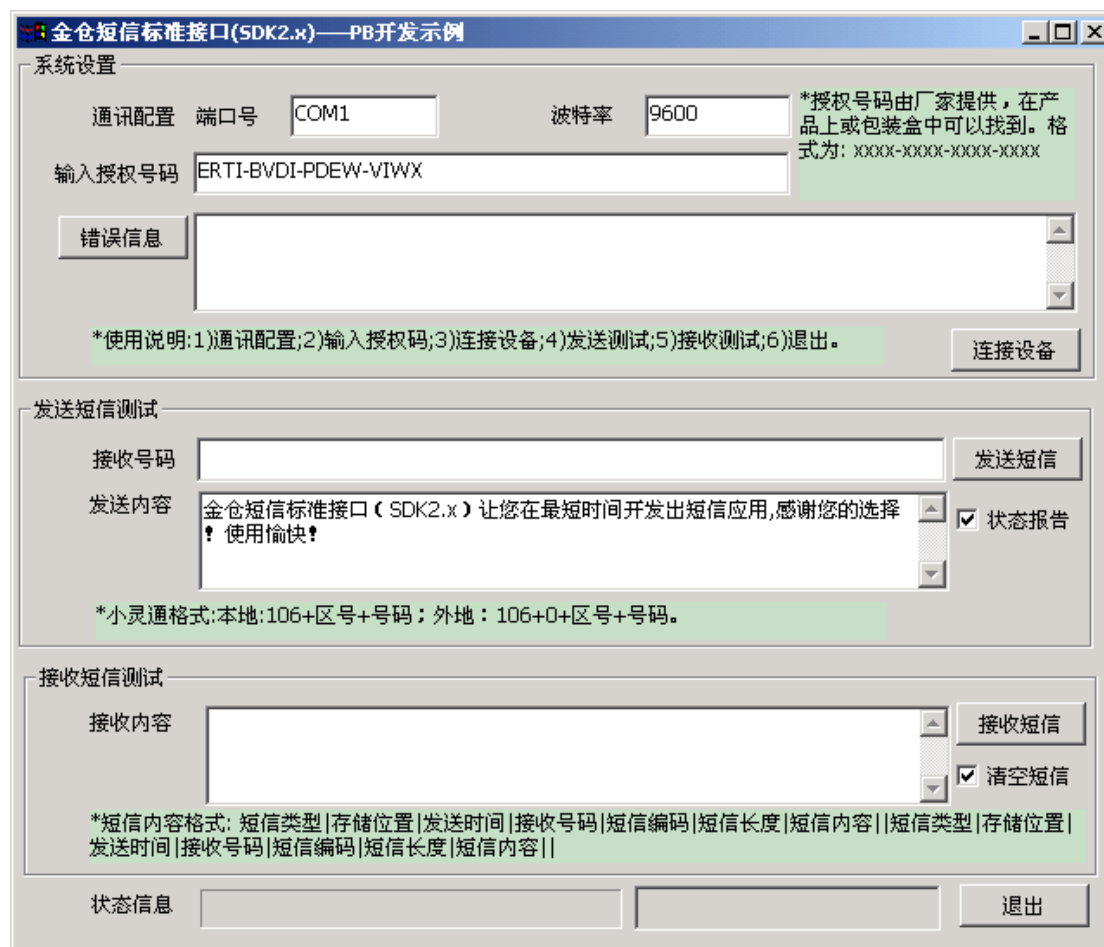
GSMModemAutoTestVB6  
GSMModemGetSnInfoVB6  
GSMModemGetErrorMsgVB6  
GSMModemSMSReadAllVB6

## 11.1 函数声明

函数说明参照 VC 说明部分，执行程序时将 GSMMultiPort.dll 拷贝到执行目录下即可。  
在 local external 中进行声明。

SUBROUTINE	GSMModemRelease(String device)	LIBRARY	"GSMMultiPort.dll"	ALIAS	FOR
"GSMModemRelease;ansi"					
Function Boolean	GSMModemIsConn(String device)	LIBRARY	"GSMMultiPort.dll"	ALIAS	FOR
"GSMModemIsConn;ansi"					
Function String	GSMModemAutoTest()	LIBRARY	"GSMMultiPort.dll"	ALIAS FOR	"GSMModemAutoTestVB6;ansi"
Function String	GSMModemGetErrorMsg(String device)	LIBRARY	"GSMMultiPort.dll"	ALIAS	FOR
"GSMModemGetErrorMsgVB6;ansi"					
Function String	GSMModemGetSnInfo(String device,String baudrate)	LIBRARY	"GSMMultiPort.dll"	ALIAS	FOR
"GSMModemGetSnInfoVB6;ansi"					
Function Boolean	GSMModemInit(String device,String baudrate,String initstring, String charset, Boolean swHandshake, String sn)	LIBRARY	"GSMMultiPort.dll"	ALIAS FOR	"GSMModemInit;ansi"
Function Boolean	GSMModemSMSsend(String device, String serviceCenterAddress, Integer encodeval, String text, Integer textlen, String phonenumber, Boolean requestStatusReport)	LIBRARY	"GSMMultiPort.dll"	ALIAS	FOR
"GSMModemSMSsend;ansi"					
Function String	GSMModemSMSReadAll(String device, Integer RD_opt)	LIBRARY	"GSMMultiPort.dll"	ALIAS	FOR
"GSMModemSMSReadAllVB6;ansi"					

## 11.2 演示界面



## 11.3 演示代码

《DemoSRC\ TestForPB》