RFID 读写设备
开发指南 JAVA

编写人：万通

V0.8.0.0

目录

1. 前言

1.1. 概述

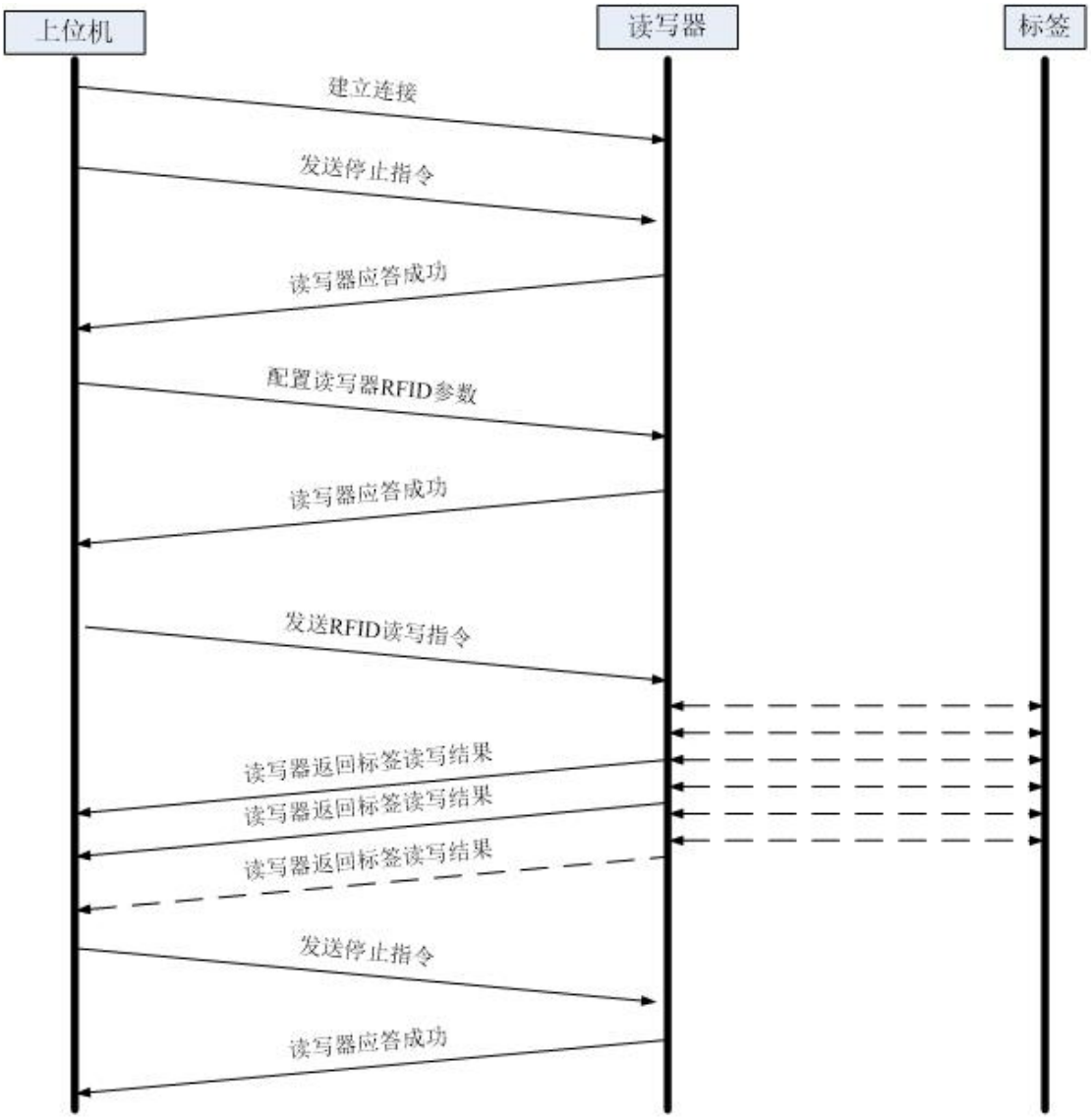
为了方便进行二次开发，我们提供了可以在 Java 平台进行运行的函数库。该库采用 Java 语言编写并封装成标准的 jar 包。

1.2. 适用设备

功能模块	适用设备类型
读写器配置与管理	
RFID 配置与操作	所有设备

1.3. 版权说明

1.4. 读写基本流程



2. Android 开发说明

Android 内部开发只能使用 TCP 127.0.0.1:8160 进行连接，AndroidRs232 与 AndroidRs485 连接是对外提供服务，硬件接口都是提供给外部服务,UHF 服务同时只能服务一个连接。

3. 快速上手

```
import com.gg.reader.api.dal.GClient;
import com.gg.reader.api.dal.HandlerTagEpcLog;
import com.gg.reader.api.dal.HandlerTagEpcOver;
import com.gg.reader.api.protocol.gx.*;


public class App {
    public static void main(String[] args) {
        GClient client = new GClient();
        Scanner sc = new Scanner(System.in);
       // tcp 连接，默认端口 8160
 //    if (client.openTcp("192.1.1.168:8160", 2000)) {

        if (client.openSerial("COM11:115200", 2000)) {
            // 订阅标签上报事件
            client.onTagEpcLog = new HandlerTagEpcLog() {
                @Override
                public void log(String readName, LogBaseEpcInfo logBaseEpcInfo) {
                    // 回调内部如有阻塞，会影响 API 正常使用
                    // 标签回调数量较多，请将标签数据先缓存起来再作业务处理
                    if (null != logBaseEpcInfo && 0 == logBaseEpcInfo.getResult() )
{
                        System.out.println(logBaseEpcInfo);
                    }
                }
            };
            client.onTagEpcOver = new HandlerTagEpcOver() {
                @Override
                public void log(String s, LogBaseEpcOver logBaseEpcOver) {
                        System.out.println("Epc log over.");
                }
            };

            // 功率配置，将 4 个天线功率都设置为 30dBm.
            MsgBaseSetPower msgBaseSetPower = new MsgBaseSetPower();
            Hashtable<Integer, Integer> hashtable = new Hashtable<>();
            hashtable.put(1,30);
            hashtable.put(2,30);
            hashtable.put(3,30);
            hashtable.put(4,30);
            msgBaseSetPower.setDicPower(hashtable);
```

```
        client.sendSynMsg(msgBaseSetPower);
        if (0 == msgBaseSetPower.getRtCode()) {
            System.out.println("Power configuration succ essful.");
        } else {
            System.out.println("Power configuration error.");
        }
        //按任意键开始读卡
        System.out.println("Enter any character to start reading the tag.");
        sc.nextLine();
        // 4 个天线读卡，读取 EPC 数据区以及 TID 数据区
        MsgBaseInventoryEpc msgBaseInventoryEpc = new MsgBaseInventoryEpc();

        msgBaseInventoryEpc.setAntennaEnable(EnumG.AntennaNo_1|EnumG.Antenn
        aNo_2|EnumG.AntennaNo_3|EnumG.AntennaNo_4);


        msgBaseInventoryEpc.setInventoryMode(EnumG.InventoryMode_Inventory)
        ;

        ParamEpcReadTid tid = new ParamEpcReadTid();
        tid.setMode(EnumG.ParamTidMode_Auto);
        tid.setLen(6);
        msgBaseInventoryEpc.setReadTid(tid);

         client.sendSynMsg(msgBaseInventoryEpc);
        if (0 == msgBaseInventoryEpc.getRtCode()) {
            System.out.println("Inventory epc successful.");
        } else {
            System.out.println("Inventory epc error.");
        }

       //按任意键停止读卡
        sc.nextLine();
        // 停止读卡，空闲态
        client.sendSynMsg(msgBaseStop);
        if (0 == msgBaseStop.getRtCode()) {
            System.out.println("Stop successful.");
        } else {
            System.out.println("Stop error.");
        }
    } else {
        System.out.println("Connect failure.");
    }
```

```
}
```

4. 连接说明

4.1. RS232 连接

包	com.gg.reader.api.dal
类	GClient
方法	public boolean OpenSerial(String readerName, int timeout)
说明	readerName: 连接字符串，如”COM1:115200” timeout: 连接确认超时时间（毫秒），如 “1000”

4.2. RS485 连接

包	com.gg.reader.api.dal
类	GClient
方法	public boolean openSerial485(String readerName, int timeout)
说明	readerName: 连接字符串，如”COM1:115200:1” timeout: 连接确认超时时间（毫秒），如 “1000”

4.3. TCP 客户端连接

包	com.gg.reader.api.dal
类	GClient
方法	public boolean openTcp(String readerName, int timeout)

说明	readerName: 连接字符串，如”192.1.1.168:8160”, 默认端口 8160 timeout: 连接确认超时时间（毫秒），如 “1000”
Android 说明	**按附录 2 Android 环境搭建提示添加权限，同时此方法请在子线程中使用** Android4.0 以上 tcp 通信请在子线程中调用 SendSynMsg()
断连心跳检测	public void setSendHeartBeat(boolean _isSendHeartbeat) 请在 **连接成功之后** 使用此方法设置（true）即可开启断连心跳检测，同时订阅了 tcp 断开连接事件就可收到断连信息。

4.4. TCP 服务器监听

包	com.gg.reader.api.dal
类	GServer
方法	public boolean open(int param)
说明	param: 上位机监听的本地接口。 使用该方法监听，需要将 UHF 读写设备配置成“客户端模式”。 “客户端模式”配置方法详见《RFID 演示软件操作手册》。

4.5. USB-HID 连接

包	com.gg.reader.api.dal
类	GClient
方法	public boolean openUsbHid(UsbDevice device)
说明	device: 可通过 UsbHidUtils 工具类 getAttachedUsbDevices()方法获取 USB 列表, 可用设备 VendorId = 1003 与 ProductId = 9249

4.6. Android RS232 连接

包	com.gg.reader.api.dal
类	GClient
方法	public boolean openAndroidSerial(String readerName, int timeout)
说明	readerName: 连接字符串，如”/dev/ttyS1:115200” timeout: 连接确认超时时间（毫秒），如 “1000”

4.7. Android RS485 连接

包	com.gg.reader.api.dal
类	GClient
方法	public boolean openAndroidRs485(String readerName, int timeout)
说明	readerName: 连接字符串，如”/dev/ttyS3:115200:1” timeout: 连接确认超时时间（毫秒），如 “1000”

4.8. Android 蓝牙连接

包	com.gg.reader.api.dal
类	GClient
方法	public boolean openBluetooth(String address, int timeout, int way, BluetoothClient bluetoothClient)

<table>
<tr><td>说明</td><td>address: 连接蓝牙地址，如”38:83:9A:0D:13:C4”
timeout: 连接确认超时时间（毫秒），如 “1000”
way: 连接方式，0-不配对,1-配对(根据系统版本高低有差异)
bluetoothClient: 蓝牙工具类，实例化、扫描，打开蓝牙等功能，详细如下。</td></tr>
<tr><td>BluetoothClient</td><td>void registerBluetoothScanReceiver(Context context)-注册蓝牙广播，初始化时调用
void unRegisterBluetoothScanReceiver(Context context)-卸载蓝牙广播，退出应用调用
void scanBluetooth(Context context)-扫描蓝牙设备且蓝牙未打开时进入系统设置打开蓝牙
void startScanner()-扫描蓝牙设备
void stopScanner()-停止扫描蓝牙
Set<BluetoothDevice> getBondDevice()-获取已配对的蓝牙设备
void openBluetooth()-打开蓝牙(异步过程)
void closeBluetooth()-关闭蓝牙(异步过程)
BluetoothAdapter getAdapter()-适配器，能够对蓝牙进行更多操作
BluetoothHandler bluetoothHandler-蓝牙开始扫描、停止扫描、以及接收扫描设备事件</td></tr>
<tr><td>权限申请</td><td><uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-feature android:name="android.hardware.location.gps" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
按需申请权限或者根据 android 版本动态申请权限</td></tr>
<tr><td>回调事件
BluetoothHandler</td><td>bluetoothClient.bluetoothHandler = new BluetoothHandler() {
@Override
public void dispense(BluetoothDevice info) {
//接收扫描的蓝牙设备(包括已配对的设备)
}
@Override
public void startDiscover() {
//开始扫描
}
@Override
public void finishDiscover() {
//扫描结束
}
};</td></tr>
<tr><td>断连心跳检测</td><td>public void setSendHeartBeat(boolean _isSendHeartbeat)
请在 <u>连接成功之后</u> 使用此方法设置（true）即可开启断连心跳检测，同时订阅了 tcp 断开连接事件就可收到断连信息。</td></tr>
</table>

4.9. Android USB To RS232 连接

包	com.gg.reader.api.dal
类	GClient
方法	public boolean openAndroidUsbToSerial(String readerName, AndroidUsbToSerialClient usbClient)
说明	readerName: 连接字符串，如"/dev/bus/usb/001/021:115200"-以实际检测情况为准，每次拔出再插入此地址都会改变("/dev/bus/usb/001/021") usbClient: usb 操作工具类，详细如下。
AndroidUsbToSerialClient	void registerUsb(Context context)-注册 usb 广播-初始化时调用 void unregisterReceiver(Context context)-卸载 usb 广播-退出应用调用 boolean applyFor(Context context, String deviceName)-usb 使用权限申请，每次拔出再插入都需要申请权限 void setReadTimeout(int readTimeout)-设置 read 超时，默认 100ms void setWriteTimeout(int writeTimeout)-设置发送超时，默认 100ms Map<String, UsbDevice> getUsbDeviceMap()-获取当前插入的 usb 设备,key 地址(打开串口方法 readerName=key)，value 设备详细信息,可通过 ProductID 与 VendorId 筛选可用设备。 HandlerAndroidUsb handlerAndroidUsb-usb 授权、拔插事件
权限申请	<uses-feature android:name="android.hardware.usb.host" />

回调接口
HandlerAndroidUsb
	usbClient.handlerAndroidUsb = new HandlerAndroidUsb() {
	@Override
	public void connectAction() {
	//usb 插上
	}
	@Override
	public void disconnectAction() {
	//usb 断开
	}
	@Override
	public void cancelApplyFor() {
	//取消 usb 授权
	}
	@Override
	public void affirmApplyFor() {
	//确认 usb 授权，在此方法里调用 openAndroidUsbToSerial()方法
	}
	};

使用示例	先授权再连接
	boolean b = usbClient.applyFor(this,"/dev/bus/usb/001/021");
	if(!b){//授权失败 确认授权或者已经授权会触发 affirmApplyFor()回调
	//授权失败处理逻辑
	}

4.10. 关闭连接

包	com.gg.reader.api.dal
类	GClient
方法	public void Close()
说明	关闭并释放链接资源。 注：对于已经失效的链接对象，需要主动调用此方法释放资源。

5. 事件说明

5.1. ISO18000-6C 标签上报事件

包	com.gg.reader.api.dal
类	GClient
事件	public HandlerTagEpcLog OnTagEpcLog;
说明	client.onTagEpcLog = new HandlerTagEpcLog() { @Override public void log(String s, LogBaseEpcInfo logBaseEpcInfo) {} 6C 标签主去上报事件：当读写器为读卡状态时，标签信息会通过此事件上报。 示例详见"**快速上手**"。 LogBaseEpcInfo：详见"**上报对象**"

上报对象

包	com.gg.reader.api.protocol.gx
类	LogBaseEpcInfo
属性	Epc：16 进制 EPC 字符串 BEpc：EPC 字节数组 Pc：PC 值 AntId：天线编号 Rssi：信号强度 Result：标签读取结果，0 为读取成功，非 0 为失败 Tid：16 进制 TID 字符串 BTid：TID 字节数组 Userdata：16 进制 Userdata 字符串 BUser：Userdata 字节数组 Reserved：16 进制保留区字符串 BRes：保留区字节数组
说明	6C 标签主动上报参数。

5.2. ISO18000-6C 标签上报结束事件

包	com.gg.reader.api.dal
类	GClient
属性	public HandlerTagEpcOver OnTagEpcOver;
说明	client.onTagEpcOver = new HandlerTagEpcOver() { @Override public void log(String s, LogBaseEpcOver logBaseEpcOver) {} 6C 标签主动上报结束参数，以确保异步消息得到同步状态。

5.3. ISO18000-6B 标签上报事件

包	com.gg.reader.api.dal
类	GClient
事件	public HandlerTag6bLog OnTag6bLog;
说明	client.onTagEpcLog = new HandlerTag6bLog() { @Override public void log(String s, LogBase6bInfo logBase6bInfo) {} 6B 标签主去上报事件：当读写器为读卡状态时，标签信息会通过此事件上报。 示例详见"**快速上手**"。 LogBase6bInfo：详见"**上报对象**"

上报对象

包	com.gg.reader.api.protocol.gx
类	LogBase6bInfo

属性	AntId：天线编号 Rssi：信号强度 Result：标签读取结果，0 为读取成功，非 0 为失败 Tid：16 进制 TID 字符串 BTid：TID 字节数组 Userdata：16 进制 Userdata 字符串 BUser：Userdata 字节数组
说明	6b 标签主动上报参数。

5.4. ISO18000-6B 标签上报结束事件

包	com.gg.reader.api.dal
类	GClient
属性	public HandlerTag6bOver OnTag6bOver;
说明	client.onTagEpcOver = new HandlerTag6bOver() { @Override public void log(String s, LogBase6bOver logBase6bOver) {} 6B 标签主动上报结束参数，以确保异步消息得到同步状态。

5.5. 国标标签上报事件

包	com.gg.reader.api.dal
类	GClient
事件	public HandlerTagGbLog onTagGbLog;

说明	client.onTagGbLog = new HandlerTagGbLog() {
	@Override
	public void log(String readerName, LogBaseGbInfo info) {
	}}
	GB 标签主动上报事件：当读写器为读卡状态时，标签信息会通过此事件上报。
	示例详见"**快速上手**"。
	LogBaseGbInfo：详见"**上报对象**"

上报对象

包	com.gg.reader.api.protocol.gx
类	LogBaseGbInfo
属性	Epc：16 进制 EPC 字符串(标签编码区) BEpc：EPC 字节数组(标签编码区) Pc：PC 值 AntId：天线编号 Rssi：信号强度 Result：标签读取结果，0 为读取成功，非 0 为失败 Tid：16 进制 TID 字符串(标签信息区) BTid：TID 字节数组(标签信息区) Userdata：16 进制 Userdata 字符串 BUser：Userdata 字节数组
说明	Gb 标签主动上报参数。

5.6. 国标标签上报结束事件

包	com.gg.reader.api.dal
类	GClient
属性	public HandlerTagGbOver OnTagGbOver;

说明	client.onTagGbOver = new HandlerTagGbOver() {
	@Override
	public void log(String readerName, LogBaseGbOver info) {}
	};
	GB 标签主动上报结束参数，以确保异步消息得到同步状态。

5.7. GPI 触发上报事件

包	com.gg.reader.api.protocol.gx
类	LogAppGpiStart
属性	public HandlerGpiStart onGpiStart;
说明	client.onGpiStart=new HandlerGpiStart() { @Override public void log(String readerName, LogAppGpiStart info) {}} 当触发起始条件满足时，读写器会主动上传一条通知消息，通知上位机触发操作已开始。

上报对象

包	com.gg.reader.api.protocol.gx
类	LogAppGpiStart
属性	gpiPort：触发 GPI 端口号 gpiPortLevel：GPI 端口电平 systemTime：当前系统时间
说明	GPI 触发上报参数。

5.8. GPI 触发上报结束事件

包	com.gg.reader.api.protocol.gx
类	LogAppGpiOver
属性	public HandlerGpiOver onGpiOver;
说明	client.onGpiOver=new HandlerGpiOver() { @Override public void log(String readerName, LogAppGpiOver info) {}} 当触发停止条件满足时，读写器会主动上传一条通知消息，通知上位机触发操作已开始。

上报对象

包	com.gg.reader.api.protocol.gx
类	LogAppGpiOver
属性	gpiPort：触发 GPI 端口号 gpiPortLevel：GPI 端口电平 systemTime：当前系统时间
说明	GPI 触发停止上报参数。

5.9. TCP 连接断开事件

包	com.gg.reader.api.dal
类	GClient
事件	public HandlerTcpDisconnected onDisconnected;

说明	Client.onDisconnected = new HandlerTcpDisconnected() {
	@Override
	public void log(String readName) {
	}}
	说明：
	➢ 连接处于 TCP，当远程连接主动断开或者物理层异常时，此事件上报。
	➢ 此事件上报后，需由上位机（调用者）释放连接对象，否则事件会循环上报，直至连接对象被释放。
	➢ 为满足不同需求，是否重连该远程设备，由上位机（调用者）自主控制。
	readerName：连接对象名称。

5.10. TCP 连接事件

包	com.gg.reader.api.dal
类	GServer
事件	public HandlerGClientConnected onGClientConnected;

说明	gServer.onGClientConnected = new HandlerGClientConnected() {
	@Override
	public void log(GClient gClient) {
	}}
	TCP 处于监听状态，远程读写设备主动连接上位机时，会触发此事件上报。
	GClient：详见”**上报对象**”。

上报对象

包	com.gg.reader.api.dal
类	GClient
属性	无。
说明	说明：此连接对象与其他主动连接的对象相同，用法完全一致。

断连心跳检测	`public void setSendHeartBeat(boolean _isSendHeartbeat)` 此事件监听上报成功之后 使用此方法设置（true）即可开启断连心跳检测，同时订阅了 tcp 断开连接事件就可收到断连信息。

5.11. 命令日志打印事件

包	com.gg.reader.api.dal
类	GClient
事件	`public HandlerDebugLog debugLog;`

说明	`client.debugLog = new HandlerDebugLog() {`
	`@Override`
	`public void sendDebugLog(String msg) {`
	`System.out.println(msg);`
	`}`
	
	`@Override`
	`public void receiveDebugLog(String msg) {`
	`System.out.println(msg);`
	`}`
	`};`
	订阅此事件即可收到发送的命令与返回的命令。

关闭日志	`public void setPrint(boolean print)` 默认为 true,设置 false 则关闭日志解析与分发。

6. 消息配置与查询说明

6.1. 发送同步消息

包	com.gg.reader.api.dal
类	GClient
方法	public void SendSynMsg(Message msg)
方法 1	public void SendSynMsg(Message msg, int timeout)
方法 2	public void SendSynMsgRetry(Message msg, int timeout, int retry)
返回值	msg.getRtCode()：消息返回码 0 为操作成功，非 0 操作失败。 msg.getRtMsg()：操作失败原因
说明	发送同步消息,详见代码示例。 提示：“读写器配置与管理”、“RFID 配置与操作”等消息，均通过此方法发送。

代码示例 1

```
// 停止指令，空闲态
MsgBaseStop msgBaseStop = new MsgBaseStop();
clientConn.SendSynMsg(msgBaseStop);
if (0 == msgBaseStop.getRtCode())
{
    System.out.println("Stop successful.");
}
else { System.out.println("Stop error."); }
```

代码示例 2

```
// 功率配置，将 4 个天线功率都设置为 30dBm.
MsgBaseSetPower msgBaseSetPower = new MsgBaseSetPower();
Hashtable<Integer, Integer> hashtable = new Hashtable<>();
hashtable.put(1, 30);
hashtable.put(2, 30);
hashtable.put(3, 30);
hashtable.put(4, 30);
```

```
msgBaseSetPower.setDicPower(hashtable);
clientConn.SendSynMsg(msgBaseSetPower);
if (0 == msgBaseSetPower.getRtCode())
{
    System.out.println("Power configuration successful.");
}
else { System.out.println("Power configuration error."); }
```

代码示例 3

```
if (null != this.clientConn)
{
    // 查询天线功率
    MsgBaseGetPower msg = new MsgBaseGetPower();
    clientConn.SendSynMsg(msg);
    if (0 == msg.getRtCode())
    {
       System.out.println(msg);
    }
```

7. 消息说明

7.1. 读写器配置与管理

7.1.1. 重启读写器

包	com.gg.reader.api.protocol.gx
类	MsgAppReset
属性	无
说明	设备重启消息，一般在修改需要重启生效的配置后执行。

7.1.2. 配置与查询串口参数

包	com.gg.reader.api.protocol.gx
配置类	MsgAppSetSerialParam
查询类	MsgAppGetSerialParam
属性	serialBaudrate: 波特率索引（0，9600 bps； 1，19200 bps； 2，115200 bps； 3，230400 bps; 4， 460800bps）
说明	(持久化配置，断电保存)配置设备串口参数。

7.1.3. 配置 GPO 状态参数

包	com.gg.reader.api.protocol.gx
配置类	MsgAppSetGpo
属性	gpo1: 0 （低，继电器断开） 1 （高，继电器闭合） gpo2: 0 （低，继电器断开） 1 （高，继电器闭合） gpo3: 0 （低，继电器断开） 1 （高，继电器闭合） gpo4: 0 （低，继电器断开） 1 （高，继电器闭合） ……
说明	(持久化配置，断电保存)配置设备 GPO 参数。 注：对于不需要控制状态的 GPO，无须赋值。

7.1.4. 查询 GPI 状态参数

包	com.gg.reader.api.protocol.gx
查询类	MsgAppGetGpiState

属性	hpGpiState: 对应 GPI 的电平状态(HashMap<Integer, Integer> , key：GPI 索引号，value：电平状态（0 低，1 高）)
说明	查询设备 GPI 状态。 注：索引号从 1 开始。

7.1.5. 配置与查询 GPI 触发参数

命名空间	com.gg.reader.api.protocol.gx
配置类	MsgAppSetGpiTrigger
查询类	MsgAppGetGpiTrigger
属性	gpiPort: GPI 端口号，索引从 0 开始 triggerStart: 触发开始（0 触发关闭，1 低电平触发，2 高电平触发，3 上升沿触发，4 下降沿触发，5 任意边沿触发） hexTriggerCommand:触发绑定命令（Hex，可为空） triggerCommand:触发绑定命令（Byte[]，可以空） triggerOver:触发停止（0 不停止，1 低电平触发，2 高电平触发，3 上升沿触发，4 下降沿触发，5 任意边沿触发，6 延时停止） overDelayTime:延时停止时间（仅当停止条件为“延时停止”生效） levelUploadSwitch:触发不停止时 IO 电平变化上传开关（0 不上传，1 上传）
说明	(持久化配置，断电保存)配置设备 GPI 触发参数。 注：此配置需在设备空闲时修改（即循环读卡状态无法更改配置）。

7.1.6. 查询软件基带版本

包	com.gg.reader.api.protocol.gx
查询类	MsgAppGetBaseVersion

属性	baseVersions：基带软件版本
说明	用于获取基带软件的版本号。

7.1.7. 查询读写器 RFID 能力

包	com.gg.reader.api.protocol.gx
查询类	MsgBaseGetCapabilities
属性	MaxPower：最大支持功率 MinPower：最小支持功率 AntennaCount：天线数量 FrequencyArray：支持的频段列表， 0， 国标 920~925MHz 1， 国标 840~845MHz 2， 国标 840~845MHz 和 920~925MHz 3， FCC，902~928MHz 4，ETSI，866~868MHz ProtocolArray：支持的协议列表， 0， ISO18000-6C/EPC C1G2 1， ISO18000-6B 2， 国标 GB/T 29768-2013 3， 国军标 GJB 7383.1-2011
说明	无。

7.1.8. 查询读写器信息

包	com.gg.reader.api.protocol.gx
查询类	MsgAppGetReaderInfo
属性	readerSerialNumber：读写器流水号 powerOnTime：上电时间 baseCompileTime：基带编译时间 appVersions：应用软件版本（如：“0.1.0.0”） appCompileTime：应用编译时间 systemVersions：操作系统版本
说明	无。

7.2. RFID 配置与操作

7.2.1. 停止指令

包	com.gg.reader.api.protocol.gx
类	MsgBaseStop
属性	无
说明	停止读写器所有的 RFID 操作，并使读写器进入到空闲状态。 提示：当读写器处于读卡状态时，所有配置消息将无法发送，必须发送停止指令。

7.2.2. 配置与查询读写器功率

包	com.gg.reader.api.protocol.gx
配置类	MsgBaseSetPower
查询类	MsgBaseGetPower

属性	dicPower：读写器对应天线功率(Hashtable<Integer, Integer>,key：天线索引号，value：天线功率值)
说明	(持久化配置，断电保存)用于对读写器当前的功率进行配置。 按所给天线常量配置 EnumG.AntennaNo_1、EnumG.AntennaNo_2.....

7.2.3. 配置与查询读写器工作频段

包	com.gg.reader.api.protocol.gx
配置类	MsgBaseSetFreqRange
查询类	MsgBaseGetFreqRange
属性	FreqRangeIndex: 频段索引，具体对应关系，详见附录 1。
说明	(持久化配置，断电保存)用于对读写器当前的工作频段进行配置。

7.2.4. 配置与查询 EPC 基带参数

包	com.gg.reader.api.protocol.gx
配置类	MsgBaseSetBaseband
查询类	MsgBaseGetBaseband
属性	BaseSpeed: EPC 基带速率（可选）。 QValue: 默认 Q 值（可选）(0~15)。 Session: （可选）(0,Session0; 1,Session1; 2,Session2; 3,Session3)。 InventoryFlag: 盘存标志参数(可选)(0,仅用 Flag A 盘存;1,仅用 Flag B 盘存;2,轮流使用 Flag A 和 Flag B)。

说明	(持久化配置，断电保存)用于配置读写器使用的基带参数。

7.2.5. 配置与查询标签上传参数

包	com.gg.reader.api.protocol.gx
配置类	MsgBaseSetTagLog
查询类	MsgBaseGetTagLog
属性	RepeatedTime: 重复标签过滤时间（可选）（表示在一个读卡指令执行周期内，在指定的重复过滤时间内相同的标签内容只上传一次，0~65535，时间单位：10ms）。 RssiTV: RSSI 阈值（可选）（标签 RSSI 值低于阈值时标签数据将不上传并丢弃）。
说明	(持久化配置，断电保存)用于对读写器上传参数进行配置。

7.2.6. 读 EPC 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseInventoryEpc
属性	AntennaEnable: 天线端口(使用天线常量,详见快速上手) InventoryMode: 连续/单次读取 (0: 单次读取模式，读写器尽在各个使能的天线上进行一轮读卡操作便结束读卡操作并自动进入空闲状态; 1: 连续读取模式，读写器一直进行读卡操作直到读写器收到停止指令后结束读卡) Filter: 选择读取参数（可选）（详见参数说明） ReadTid: TID 读取参数（可选）（详见参数说明） ReadUserdata: 用户数据区读取参数（可选）（详见参数说明） ReadReserved: 保留区读取参数（可选）（详见参数说明） HexPassword: 访问密码（可选）

说明	用于配置读写器的标签读取参数并启动读卡操作，任何读取标签数据操作都需要先获取到标签 EPC 码，所以任何读卡操作都会得到 EPC 码。

7.2.7. 写 EPC 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseWriteEpc
属性	AntennaEnable：天线端口 Area：待写入的标签数据区(0，保留区；1，EPC 区；2，TID 区；3，用户数据区) Start：待写入标签数据区的字起始地址 HexWriteData：待写入的数据内容（可选）(16 进制) BwriteData：待写入的数据内容 Filter：选择读取参数（可选）（详见参数说明） HexPassword：访问密码（可选）
说明	➢ 读写器对 EPC 标签进行写操作，本指令定义的写操作为单次操作 ➢ ISO18000-6C 协议规定读写操作最小数据单为字。 ➢ EPC 区由 CRC-16(第 0 个字) + PC(第 1 个字) + EPC 组成： CRC16：第 0 个字，不可写 。 PC：第 1 个字，前 5 个 bit 位表示 EPC 字长度，即 PC 计算方法为 EPC 的字长度左移 11 位。
写入结果	0， 写入成功 1， 天线端口参数错误 2， 选择参数错误 3， 写入参数错误 4， CRC 校验错误 5， 功率不足 6， 数据区溢出 7， 数据区被锁定 8， 访问密码错误 9， 其他标签错误 10， 标签丢失 11，读写器发送指令错误

7.2.8. 锁 EPC 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseLockEpc
属性	AntennaEnable：天线端口 Area：待锁定的标签数据区(0，灭活密码区；1，访问密码区；2，EPC 区；3，TID 区；4，用户数据区) Mode：锁操作类型(0，解锁；1，锁定；2，永久解锁；3，永久锁定) Filter：选择读取参数（可选）（详见参数说明） HexPassword：访问密码（可选）
说明	对标签进行锁或解锁操作，本指令定义的操作为单次操作

7.2.9. 灭活 EPC 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseDestoryEpc
属性	AntennaEnable：天线端口 HexPassword：销毁密码 Filter：选择读取参数（可选）（详见参数说明）
说明	对标签进灭活操作，进行灭活后的标签将永久失效，此操作为不可逆操作。本指令定义的操作为单次操作

7.2.10. 读 6B 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseInventory6b

属性	AntennaEnable: 天线端口 InventoryMode: 连续/单次读取 (0: 单次读取模式，读写器尽在各个使能的天线上进行一轮读卡操作便结束读卡操作并自动进入空闲状态; 1: 连续读取模式，读写器一直进行读卡操作直到读写器收到停止指令后结束读卡) Area：读取内容(0，仅读取 6B TID；1，读取 6B TID+用户数据；2，仅读取用户数据) ReadUserdata: 用户数据区读取参数（可选）（详见参数说明） HexMatchTid：待匹配 6B 标签的 TID 码（可选）(16 进制) BMatchTid：待匹配 6B 标签的 TID 码（可选）
说明	用于 ISO18000-6B 标签的数据读取操作

7.2.11. 写 6B 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseWrite6b
属性	AntennaEnable: 天线端口 HexMatchTid：待匹配 6B 标签的 TID 码（可选）(16 进制) BMatchTid：待匹配 6B 标签的 TID 码 Start：待写入标签数据区的字节起始地址 HexWriteData：待写入的数据内容（可选）(16 进制) BwriteData：待写入的数据内容
说明	对 6B 标签进行写操作，本指令定义的写操作为单次操作

7.2.12. 6B 标签锁定

包	com.gg.reader.api.protocol.gx
类	MsgBaseLock6b
属性	AntennaEnable: 天线端口 HexMatchTid：待匹配 6B 标签的 TID 码（可选）(16 进制) BMatchTid：待匹配 6B 标签的 TID 码 LockIndex：待锁定数据的字节地址

说明	对 6B 标签数据进行锁定操作，该操作不可撤销和逆转，本指令定义的锁定操作为单次操作

7.2.13. 6B 标签锁定查询

包	com.gg.reader.api.protocol.gx
类	MsgBaseLock6bGet
属性	AntennaEnable：天线端口 HexMatchTid：待匹配 6B 标签的 TID 码（可选）(16 进制) BMatchTid：待匹配 6B 标签的 TID 码 LockIndex：待锁定查询数据的字节地址
说明	对 6B 标签数据锁定状态进行查询，本指令定义的锁定查询操作为单次操作

7.2.14. 读 GB 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseInventoryGb
属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） InventoryMode：连续/单次读取（0：单次读取模式，读写器尽在各个使能的天线上进行一轮读卡操作便结束读卡操作并自动进入空闲状态；1：连续读取模式，读写器一直进行读卡操作直到读写器收到停止指令后结束读卡） Filter：选择读取参数（可选）（详见参数说明） ReadTid：TID 读取参数（可选）（详见参数说明） ReadUserdata：用户数据区读取参数（可选）（详见参数说明） HexPassword：访问密码（可选）
说明	用于配置读写器的标签读取参数并启动读卡操作，任何读取标签数据操作都需要先获取到标签编码，所以任何读卡操作都会得到标签编码（EPC）。

7.2.15. 写 GB 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseWriteGb
属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） Area：待写入的标签数据区(0x10，标签编码区 0x20，标签安全区 0x30~0x3F，用户子区 0~15 区) Start：待写入标签数据区的字起始地址 HexWriteData：待写入的数据内容（可选）(16 进制) BwriteData：待写入的数据内容 Filter：选择读取参数（可选）（详见参数说明） HexPassword：访问密码（可选）
说明	➢ 读写器对 GB 标签进行写操作，本指令定义的写操作为单次操作 ➢ GB 协议规定读写操作最小数据单为字。

7.2.16. 锁 GB 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseLockGb

属性	AntennaEnable: 天线端口（使用天线枚举，详见快速上手） Area：待锁定的标签数据区(0x00 标签信息区，0x10 标签编码区，0x20 标签安全区，0x30~0x3F 用户子区 0~15) Mode：锁操作类型 0x00，可读可写。 0x01，可读不可写。 0x02，不可读可写。 0x03，不可读不可写。 0x11，安全模式设置为不需要鉴别；此操作区域必须为标签安全区。 0x12，安全模式设置为需要鉴别，不需要安全通信；此操作区域必须为标签安全区。 0x13，安全模式设置为需要鉴别，需要安全通信；此操作区域必须为标签安全区。 Filter: 选择读取参数（可选）（详见参数说明） HexPassword: 访问密码（可选）
说明	对标签进行锁或解锁操作，本指令定义的操作为单次操作

7.2.17. 灭活 GB 标签

包	com.gg.reader.api.protocol.gx
类	MsgBaseDestoryGb
属性	AntennaEnable: 天线端口（使用天线枚举，详见快速上手） HexPassword: 销毁密码 Filter: 选择读取参数（可选）（详见参数说明）
说明	对标签进灭活操作，进行灭活后的标签将永久失效，此操作为不可逆操作。本指令定义的操作为单次操作

8. 参数说明

8.1.1. 6C 标签选择参数

包	com.gg.reader.api.protocol.gx
类	ParamEpcFilter
属性	Area: 要匹配的数据区(1，EPC 区；2，TID 区；3，用户数据区) BitStart: 匹配数据起始位地址 BitLength: 需要匹配的数据位长度 HexData: 需要匹配的数据内容（可选）(16 进制) BData: 需要匹配的数据内容
说明	可选参数(EPC 前 32 位为 PC 值，故区别 EPC 时的起始位地址通常为 32)

8.1.2. 6C 标签读取 TID 参数

包	com.gg.reader.api.protocol.gx
类	ParamEpcReadTid
属性	Mode：TID 读取模式配置，(0，TID 读取长度自适应，但最大长度不超过字节 1 定义的长度；1，按照字节 1 定义的长度读取 TID) Len：读写器需要读取 TID 数据的字(word，16bits，下同)长度
说明	可选参数

8.1.3. 6C 标签读取用户数据区参数

包	com.gg.reader.api.protocol.gx
类	ParamEpcReadUserdata
属性	Start：起始字地址 Len：读写器需要读取的用户数据的字长度

说明	可选参数

8.1.4. 6B 标签读用户数据区参数

包	com.gg.reader.api.protocol.gx
类	Param6bReadUserdata
属性	Start：用户数据起始字节地址 Len：用户数据字节长度
说明	可选参数

8.1.5. GB 标签读用户数据区参数

包	com.gg.reader.api.protocol.gx
类	ParamGbReadUserdata
属性	ChildArea：用户子区 Start：用户数据起始字节地址 Len：用户数据字节长度
说明	可选参数

9. 附录 1

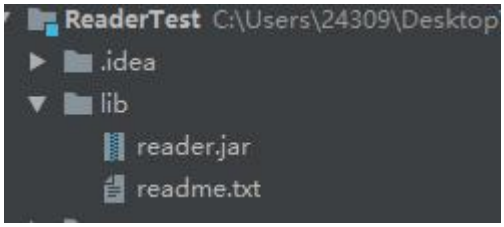
读写器所支持的频段列表

索引	说明
0	国标 920~925MHz
1	国标 840~845MHz
2	国标 840~845MHz 和 920~925MHz
3	FCC，902~928MHz
4	ETSI，866~868MHz

10. 附录 2

1. IDEA(maven)环境搭建

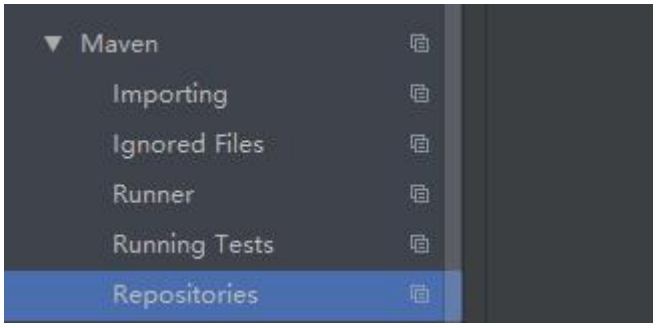
① IDEA 中创建 maven 工程，在项目目录下新建 lib 文件夹，复制 reader.jar 到该目录下。



② 打开 IDEA 中的 Terminal，依次执行安装 jar 包命令到本地仓库。

```
mvn install:install-file -Dfile=lib/reader.jar -DgroupId=com.gg.reader -DartifactId=greader-api -Dversion=1.0 -Dpackaging=jar
```

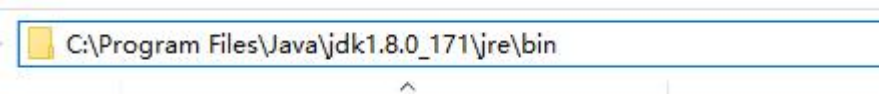
③ 安装成功后，打开 maven 配置，File-->Settings-->搜索 maven,查看当前仓库是不是所要使用的本地仓库，点击 maven 下的 Repositories,选择本地仓库，点击右边按钮更新。



④ 打开项目 pom 文件，添加以下依赖

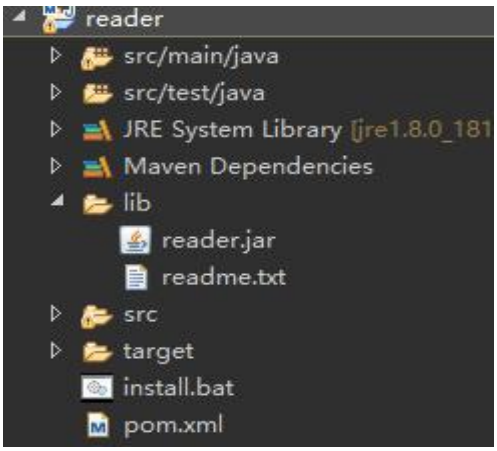
```
<dependency>
  <groupId>com.gg.reader</groupId>
  <artifactId>greader-api</artifactId>
  <version>1.0</version>
</dependency>
```

⑤ 将对应系统的静态库文件添加到 jdk 所在 jre 路径，JDK-->jre-->bin.



2. Eclipse(maven)环境搭建

① Eclipse 中创建 maven 工程，在项目目录下新建 lib 文件夹，复制 reader.jar 到该目录下。



② 点击 File-->Import-->Maven-->Install or deploy.....(安装到本地)

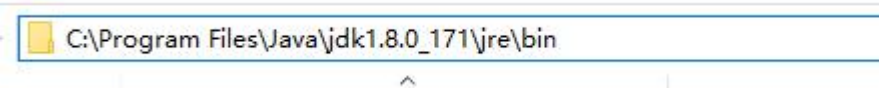
Reader:

1) Artifact file-->jar 所在路径
2) GroupId-->com.gg.reader
3) Artifact Id-->greader-api
4) Version-->1.0

③ 打开项目 pom 文件，添加以下依赖

```
<dependency>
  <groupId>com.gg.reader</groupId>
  <artifactId>greader-api</artifactId>
  <version>1.0</version>
</dependency>
```

④ 将对应系统的静态库文件添加到 jdk 所在 jre 路径，JDK-->jre-->bin.



3. 非 Maven 环境

① IDEA：

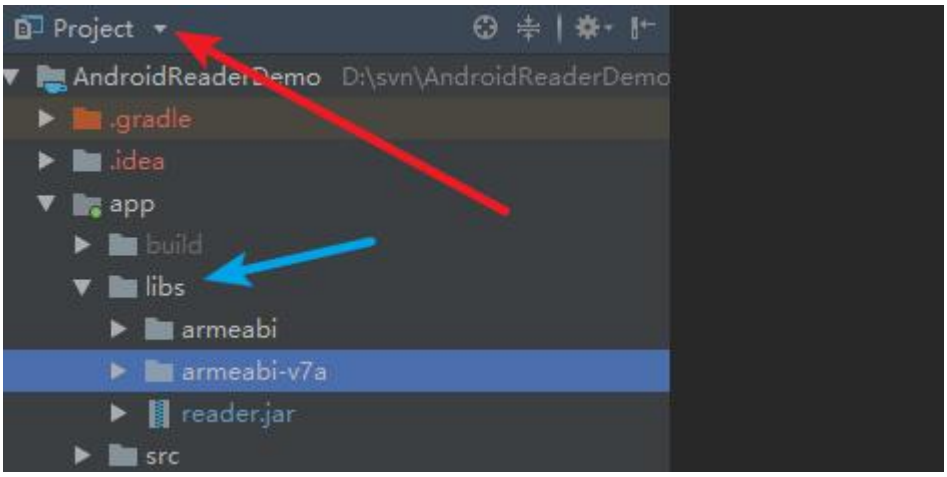
选中 jar 包，右键点击 Add as Library..确认。

② Eclipse:

选中 jar 包，右键点击 Build Path-->Add to Build Path。

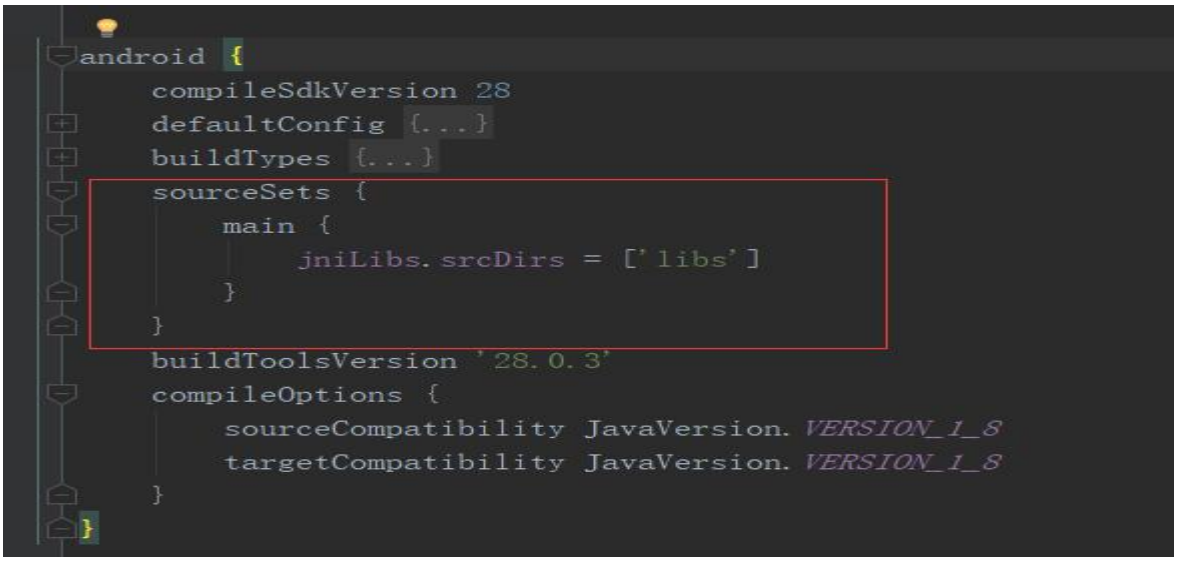
4. AndroidStudio 环境搭建

① 在项目 project 状态下，把 lib 里的三个文件复制到项目 libs 文件夹下，点击编译按钮自动加载 jar 文件，如图所示：



② 随后在 app 目录下的 build.gradle 的 android 节点下添加以下内容，随后点击编译即可。

```
sourceSets {
    main {
        jniLibs.srcDirs = ['libs']
    }
}
```



③ 最后在 AndroidManifest.xml 文件中添加以下权限(TCP 连接需要此权限)。

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.gxwl.reader">

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <application...>

</manifest>
```

5. Linux 下搭建

① 将所标识文件夹下的 librxtxSerial.so 串口支持文件放到 jdk/jre/lib/amd64/目录下。