CH341PAR

Android JAR 库接口说明

版本： V1.00

http://wch.cn

CH341PAR Android JAR 库接口说明

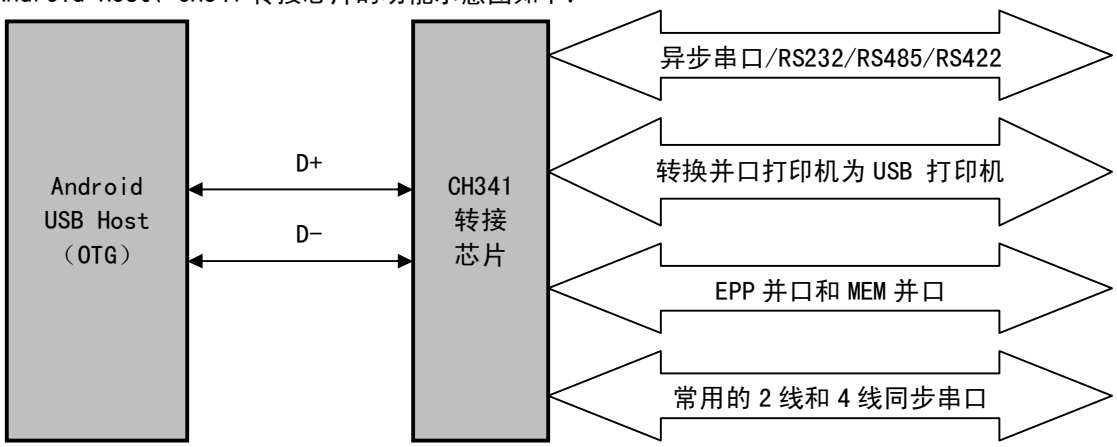
版本： 1

简介

CH341 是一个 USB 总线的转接芯片，通过 USB 总线提供异步 UART、并口、常用的 2 线和 4 线同步串行接口等方式。并口方式下，提供了 EPP/MEM 8 位并行接口，而常用的同步串行接口主要包含 2 线 IIC、4 线 SPI 等。

本文档主要介绍 CH341USB 转 EPP、MEM、IIC、SPI 模式（以下简称 CH341PAR）操作的 Java Driver 库,以及 Android 下如何使用 APK 操作 CH341PAR 实现数据通讯。该驱动基于 Android USB Host 协议完成，用户可调用所提供的相关接口 API 实现与 Android USB Host 的通讯。

Android Host、CH341 转接芯片的功能示意图如下：



CH341PAR 提供的库需要满足以下两个条件方可实现数据通讯：

1、Android 3.1 及以上系统版本；

2、Android 端具有 USB Host 或 OTG 接口。

本文档将会重点说明 Android USB Host 与 CH341 建立通讯机制的 Java 包 API 以及我司提供的演示程序的操作说明。关于 Android USB Host 协议说明，可以参考 Google 官方文档。

1、Android USB Host

本文档所描述的演示程序皆是针对 Android 3.1 及以上版本系统。Android 应用程序的启动参数是定义在 device_filter.xml 文件中的 product-id 和 vendor-id。基于 CH341PAR 开发的 Android 应用程序主要分为两个部分，如下图：

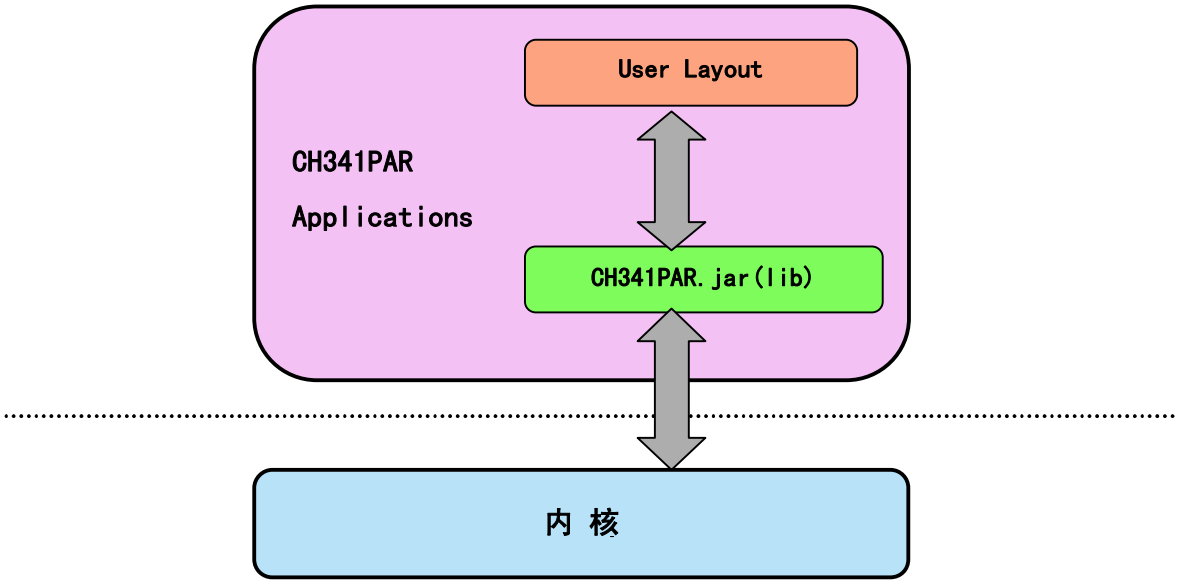


图 1.1 CH341PAR Android Application

User Layout： 主要是由用户根据自己的需求完成相关的代码，通过调用 CH341PAR 相关的接口函数实现 EPP、MEM、IIC、SPI 的通讯方式。我司所提供的例程基于 Fragment 结构设计，在 Android 手机或平板上会呈现出不同的主界面。对比效果图如下：

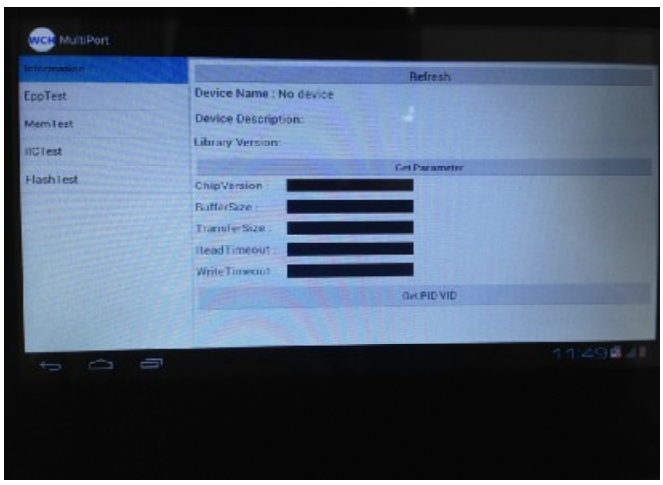


图 1.2 平板效果图

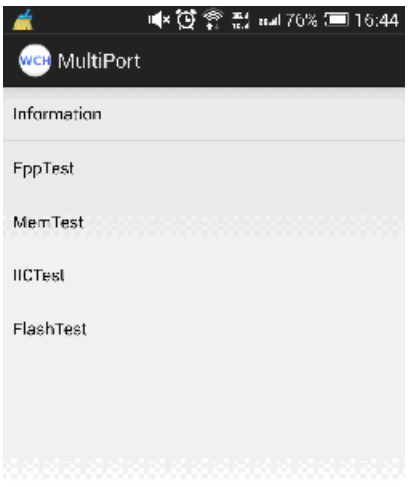


图 1.3 手机效果图

注：以下的演示内容均是针对手机每一个功能子界面来阐述，用户只需要根据相应的功能模块涉及的接口函数实现自己的功能及 UI。

CH341PAR.jar： 实现 EPP、MEM、IIC、SPI 通讯方式的 Java 包，提供 API 给 User Layout 调用，相关的接口 API 内容参考后续说明。

2、CH341PAR 设备信息

2.1 说明

针对 CH341PAR 基本信息的获取及设置操作提供了 CH341SystemInit、CH341GetBufSizeValue、CH341TransPackageSize、CH341GetPIDVID 等方法，在获取信息之前都进行了设备是否连接的判断，调用 isConnected 方法即可，库中还提供操作 CH341 IO 口的方法，如 CH341GetInput、CH341SetOutput、CH341Set_D5_D0。下图为 CH341PAR 演示界面：

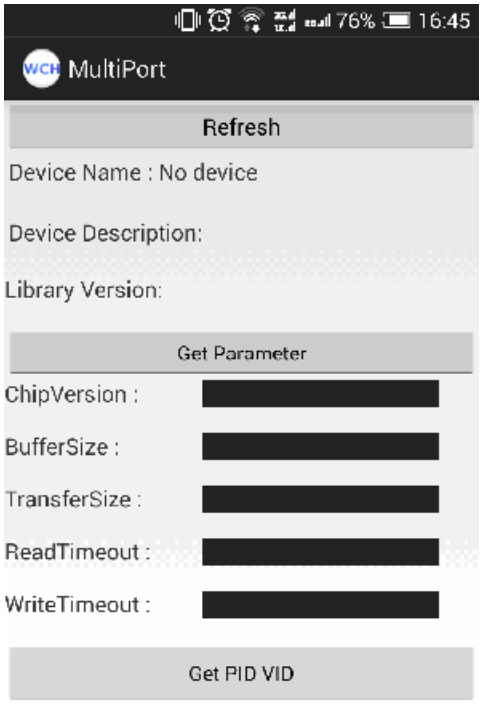


图 2.1 Information Activity

2.2 设备信息接口

CH341SystemInit ： CH341PAR 的初始化操作。
函数原型：public int CH341SystemInit()
若初始化成功，则返回 0，失败则返回负值。

CH341GetPIDVID ： 获取 CH341PAR 设备的 PIDVID。
函数原型：public int[][] CH341getPIDVID()
返回二维数组类型：
int[0][n]为 CH341PAR 设备的 VID;
int[1][n]为 CH341PAR 设备的 PID。
其中 n 为 CH341PAR 的并口及同步串行接口的 VIDPID 数目，最多可读出两组，否则设备与 CH341PAR 驱动不匹配。

CH341GetInput：通过 CH341 直接获取引脚状态
函数原型：public boolean CH341GetInput(int[] iStatus)
iStatus ：bit7-bit0 对应 CH341 的 D7-D0 引脚，bit8 对应 CH341 的 ERR#引脚，
bit9 对应 CH341 的 PEMP 引脚， bit10 对应 CH341 的 INT#引脚，
bit11 对应 CH341 的 SLCT 引脚， bit13 对应 CH341 的 BUSY/WAIT#引脚，
bit23 对应 CH341 的 SDA 引脚， bit15 对应 CH341 的 SLCTIN#/ADDRS#引脚，
bit14 对应 CH341 的 AUTOFD#/DATAS#引脚

返回为 true，则获取状态成功，User Layout 中取 iStatus[0]的值即可，返回 false，则操作失败。

CH341SetOutput ：设置 CH341 的 I/O 方向，需要谨慎使用该 API，防止修改 I/O 方向使输入引脚变为输出引脚导致与其它输出引脚之间短路而损坏芯片

函数原型 ：public boolean CH341SetOutput(long iEnable, long iSetDirOut, long iSetDataOut)

iEnable ：数据有效标志

→ 位 0 为 1 说明 iSetDataOut 的 bit15-bit8 有效,否则忽略

→ 位 1 为 1 说明 iSetDirOut 的 bit15-bit8 有效,否则忽略

→ 位 2 为 1 说明 iSetDataOut 的 bit7-bit0 有效,否则忽略

→ 位 3 为 1 说明 iSetDirOut 的 bit7-bit0 有效,否则忽略

→ 位 4 为 1 说明 iSetDataOut 的 bit23-bit16 有效,否则忽略

iSetDirOut ：设置 I/O 方向,某位清 0 则对应引脚为输入,某位置 1 则对应引脚为输出,并口方式下默认值为 0x000FC000,参考下面的**位说明**

iSetDataOut ：输出数据,如果 I/O 方向为输出,那么某位清 0 时对应引脚输出低电平,某位置 1 时对应引脚输出高电平,参考下面的**位说明**

返回为 true，则操作成功，返回 false，操作设置失败。

位说明：

bit7-bit0 对应 CH341 的 D7-D0 引脚，

bit8 对应 CH341 的 ERR#引脚， bit9 对应 CH341 的 PEMP 引脚，

bit10 对应 CH341 的 INT#引脚， bit11 对应 CH341 的 SLCT 引脚，

bit13 对应 CH341 的 WAIT#引脚， bit14 对应 CH341 的 DATAS#/READ#引脚，

bit15 对应 CH341 的 ADDRS#/ADDR/ALE 引脚

以下引脚只能输出，不考虑 I/O 方向：

bit16 对应 CH341 的 RESET#引脚， bit17 对应 CH341 的 WRITE#引脚，

bit18 对应 CH341 的 SCL 引脚， bit29 对应 CH341 的 SDA 引脚

CH341Set_D5_D0 ：设置 CH341 的 D5-D0 引脚的 I/O 方向,并通过 CH341 的 D5-D0 引脚直接输出数据,效率比 CH341SetOutput 更高，使用时需要谨慎使用该 API，防止修改 I/O 方向使输入引脚变为输出引脚导致与其它输出引脚之间短路而损坏芯片

函数原型 ：public boolean CH341Set_D5_D0(long iSetDirOut, long iSetDataOut)

iSetDirOut ：设置 D5-D0 各引脚的 I/O 方向,某位清 0 则对应引脚为输入,某位置 1 则对应引脚为输出,并口方式下默认值为 0x00 全部输入

iSetDataOut ：设置 D5-D0 各引脚的输出数据,如果 I/O 方向为输出,那么某位清 0 时对应引脚输出低电平,某位置 1 时对应引脚输出高电平

返回为 true，则操作成功，返回 false，操作设置失败。

除了上述提供的接口 API，用户还可以根据自己的设备来设置读写超时时间：

函数原型 ：public boolean CH341SetTimeOut(int WriteTimeOut, int ReadTimeOut)

WriteTimeOut：设置写超时时间，默认为 1000ms；

ReadTimeOut ：设置读超时时间，默认为 10000ms。

用户通过 CH341GetReadTimeOutValue 获取读超时时间，CH341GetWriteTimeOutValue 获取写超时时间，CH341GetChipVersion 来读取芯片版本 id 信息，根据此信息可以判断芯片的型号。

3、CH341PAR EPP 功能

3.1 说明

针对 CH341PAR EPP 操作提供了 CH341InitEpp、CH341EppReadData、CH341EppWriteData 方法。

用户可根据所提供的演示 Demo 例程合理使用上述方法。

3.2 EPP 操作接口

CH341InitEpp：设置为并口 EPP 模式，并初始化并口操作。
函数原型　：public boolean CH341InitEpp()
若设置 EPP 模式失败或初始化并口失败，返回 false，否则返回 true。

CH341EppReadData：EPP 模式读数据操作。
函数原型　：public int CH341EppReadData(byte[] oBuffer, long ioLength)
oBuffer　：接收缓冲区；
ioLength　：读取的字节数，用户也可以根据需要设置此值，长度空间为 0 到 4096 字节；
返回实际读取的字节数。

CH341EppWriteData：EPP 模式写数据操作。
函数原型　：public int CH341EppWriteData(byte[] iBuffer, long ioLength)
iBuffer　：发送缓冲区；
ioLength　：发送的字节数，（0 < ioLength <= 255 bytes）；
返回值为写成功的字节数。

除了上述提供的 API 接口对数据读写操作外，还提供针对 EPP 地址的读写操作，CH341EppWriteAddr 和 CH341EppReadAddr。

4、CH341PAR MEM 功能

4.1 说明

针对 CH341PAR MEM 操作提供了 CH341InitMem、CH341MEMWriteData、CH341MEMReadData 方法。用户可根据所提供的演示 Demo 例程合理使用上述方法。

4.2 MEM 操作接口

CH341InitMem：设置并口为 MEM 模式，并初始化并口操作。
函数原型　：public boolean CH341InitMem()
若设置 MEM 模式失败或初始化并口失败，返回 false，否则返回 true。

CH341MEMWriteData：MEM 写操作。
函数原型　：public int CH341MEMWriteData(byte[] iBuffer, long ioLength, int PipeMode)
iBuffer　：发送缓冲区；
ioLength　：发送的字节数，（0 < ioLength <= 255 bytes）；
PipeMode　：写地址通道（0、1）；
返回值为写成功的字节数。

CH341MEMReadData：MEM 读操作。
函数原型　：public int CH341MEMReadData(byte[] oBuffer, long ioLength, int PipeMode)
oBuffer　：接收缓冲区；
ioLength　：读取的字节数，用户也可以根据需要设置此值，长度空间为 0 到 4096 字节；
PipeMode　：读地址通道（0、1）；
返回实际读取的字节数。

5、CH341PAR IIC 功能

5.1 说明

针对 CH341PAR IIC 操作提供 CH341SetStream、CH341ReadEEPROM、CH341WriteEEPROM 方法，其中 CH341ReadEEPROM 与 CH341WriteEEPROM 都调用了 CH341StreamI2C 来实现两线串口的数据流读写。下图为打开 IIC Activity 演示界面：

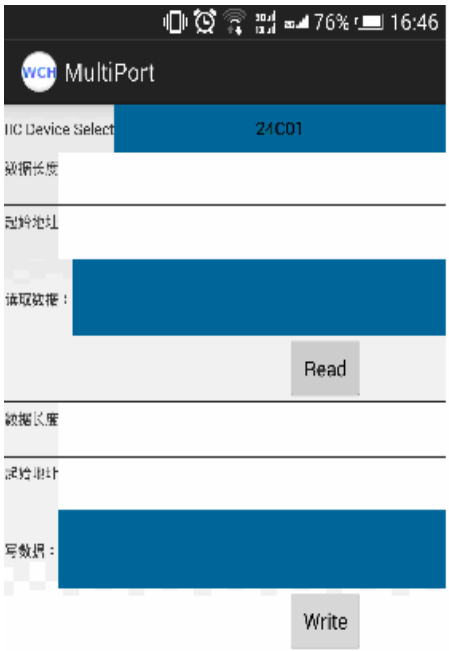


图 5.1 IIC Activity 演示界面

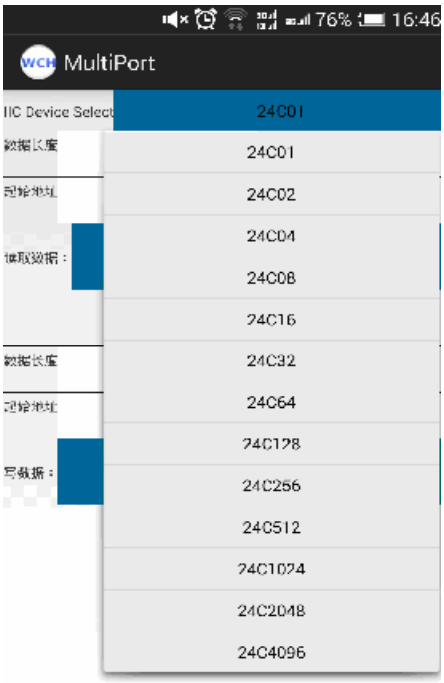


图 5.2 EEPROM 类型选择

5.2 IIC 操作接口

CH341SetStream：设置同步串口流模式。

函数原型：public boolean CH341SetStream(long Mode)

Mode ：Bit1~Bit0： IIC 速度/SCL 频率

00=低速 20KHz，01=标准 100KHz

10=快速 400KHz，11=高速 750KHz

Bit2 ： SPI 的 I/O 数/IO 引脚，0=单入单出(4 线接口)，1=双入双出(5 线接口)

Bit7 ： SPI 字节的位顺序，0=低位在前，1=高位在前

Other： 必须为 0

返回 true，则设置正确，否则设置出错，用户也可根据此方法设置 SPI。

CH341StreamI2C：处理两线串口的数据流，适用于所有两线串口的设备

函数原型：public boolean CH341StreamI2C(long WriteLen, char[]WriteBuffer, long ReadLen, char[]ReadBuffer)

WriteLen ：准备写入的数据字节数

WriteBuffer ：指向写入缓冲区,首字节是设备地址及读写位

ReadLen ：准备读取的数据字节数

ReadBuffer ：指向输出缓冲区

对两线串口设备进操作。例如，从 24C256 中 3200H 开始的地址读出 10 字节的数据：

```
char[] OutBuf = new char[5];     //待写数据缓冲区
byte[] InBuf = new byte[300]; //读出数据缓冲区
OutBuf[0]=0xA1; OutBuf[1]=0x32; OutBuf[2]=0x00; // 待写数据：设备地址及单元地址
CH341StreamI2C(3, OutBuf, 10, InBuf ); //设备处理两线串口的数据流
```

CH341ReadEEPROM： 从 EEPROM 中读取数据块
函数原型：public boolean CH341ReadEEPROM(int id, long iAddr, long iLen, byte[]oBuffer)
id ：指定的 EEPROM 类型
iAddr ：指定数据单元的地址
iLen ：准备读取的数据字节数
oBuffer ：接收缓冲区
返回 true，则读取数据成功，否则失败。

CH341WriteEEPROM： 向 EEPROM 中写入数据块
函数原型：public boolean CH341WriteEEPROM(int id, long iAddr, long iLen, char[]iBuffer)
id ：指定的 EEPROM 类型
iAddr ：指定数据单元的地址
iLen ：准备读取的数据字节数
iBuffer ：发送数据缓冲区
返回 true，则写数据成功，否则失败。

6、CH341PAR SPI 功能

6.1 说明

针对 CH341PAR SPI 操作提供了 CH341SetStream、CH341StreamSPI4 方法，以读写 FLASH 的实例，提供给用户 Ch341WriteSpi、Ch341FlashReadByte 方法。下图为打开 Flash Activity 演示界面：

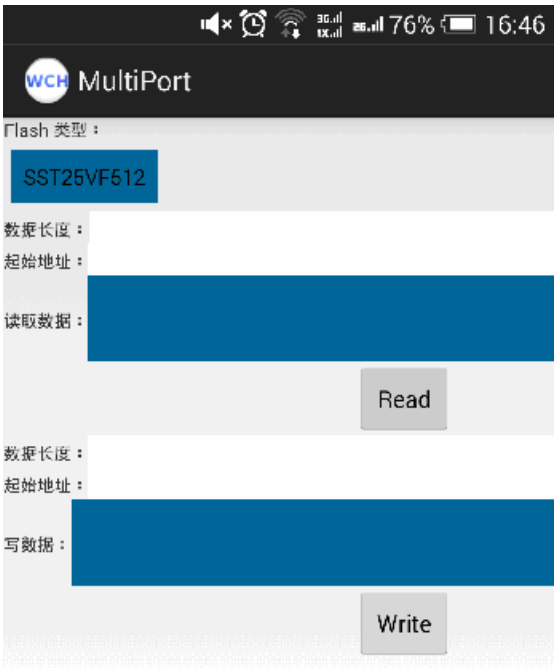


图 6.1 Flash Activity 演示界面

6.2 SPI 操作接口

CH341StreamSPI4： 处理 SPI 数据流,4 线接口
函数原型 ：public boolean CH341StreamSPI4(char chipSelect, long iLength, byte[] ioBuffer)
chipSelect ：片选控制，bit7 为 0 则忽略片选控制，bit7 为 1 则参数有效
iLength ：准备传输的数据字节数
ioBuffer ： 缓冲区,放置准备从 DOUT 写出的数据,返回后是从 DIN 读入的数据

SPI 读写成功，则返回 true，否则返回 false。

Ch341WriteSpi：写数据到 SPI。
函数原型 ：public boolean Ch341WriteSpi(long iAddr, long iLength, String writebyte)
iAddr ：指定数据单元的地址
iLength ：准备写入的数据字节数
writebyte：User Layout 层的写操作部分的 EditText 控件内容。
返回 true，则写数据成功，否则失败。

Ch341FlashReadByte：读 SPI 数据。
函数原型 ：public String Ch341FlashReadByte(long iAddr, long iLength)
iAddr ：指定数据单元的地址
iLength ：准备读取的数据字节数
返回 null，则读操作失败，否则返回直接显示在 EditText 控件上的 String 类型数据。

7、软件演示操作说明

用户在有 Host 或者 OTG 接口的 Android 设备上安装我司提供的测试软件（即 MultiPort.apk）。若是第一次使用该软件，则在插入 CH341PAR 功能模块以后，系统会自动弹出权限请求窗口，点击“Use by default for this USB device”，选择确定操作，以后再使用该模块就不会弹出这个 permissions 请求窗口；如果不选择“Use by default for this USB device”而直接确定操作，则下次使用该模块时还会弹出 permissions 请求窗口。

在软件打开过程中，在 Activity 中的 OnResume 中执行完成对 USB 设备的枚举、打开设备、获取设备资源信息等步骤，然后用户需要先操作 Information Activity，查看获取的信息是否正确，确定无误之后，进入相应功能的 Activity，执行数据传输的操作。

8、其他

用户可根据我司提供的 Demo 程序以及开放的函数接口说明，开发相关产品。若在使用我司提供的测试软件过程中出现问题，可以查看我司提供的《Android 免驱常见问题及解决手册》。