Sunmi RFID SDK 集成及开发者文档

文档更新说明

编号	更新日期	组件版本	更新内容	撰写人
1.0.0	2020/02/28	2.6.x	原始版本	曾冬阳(Darren Zeng)
1.0.1	2020/04/07	2.6.x	完善文档，集成 SDK，接口参数返回说明等	曾冬阳(Darren Zeng)

1. 简介

RFID-UHF 项目，主要用于仓库，货运等商品盘存，商米支持设备 L2k（需要单独配置 UHF 底座）。

2. SDK 集成

2.1. SDK 简介

SDK 包含服务连接，设备连接状态，标签盘存、标签操作接口封装。
SDK 结构：

com.sunmi.rfid.constant.CMD.java -- 操作类型常量
com.sunmi.rfid.constant.ParamCts.java -- 回调参数常量
com.sunmi.rfid.entity.DataParameter.java -- 数据参数
com.sunmi.rfid.ReaderCall.java -- 接口数据回调接口
com.sunmi.rfid.RFIDHelper.java -- 标签操作接口
com.sunmi.rfid.ServicesHelper.java -- 服务接口实现
com.sunmi.rfid.RFIDManager.java -- RFID 设备连接管理

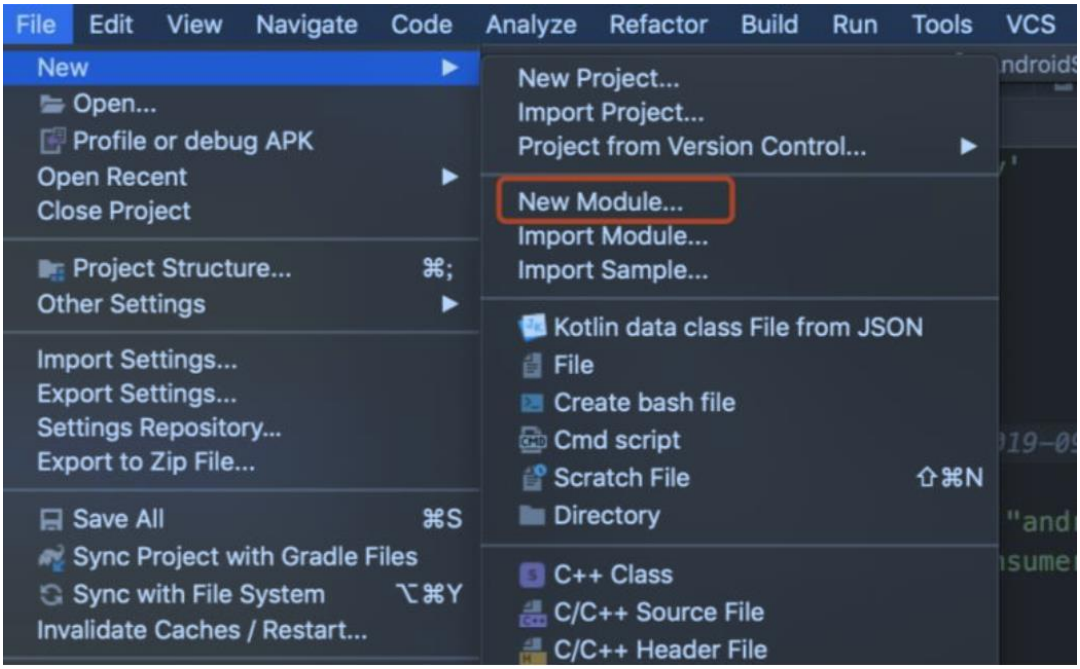
2.2. SDK 使用

2.2.1. 资源文件(AAR 文件)

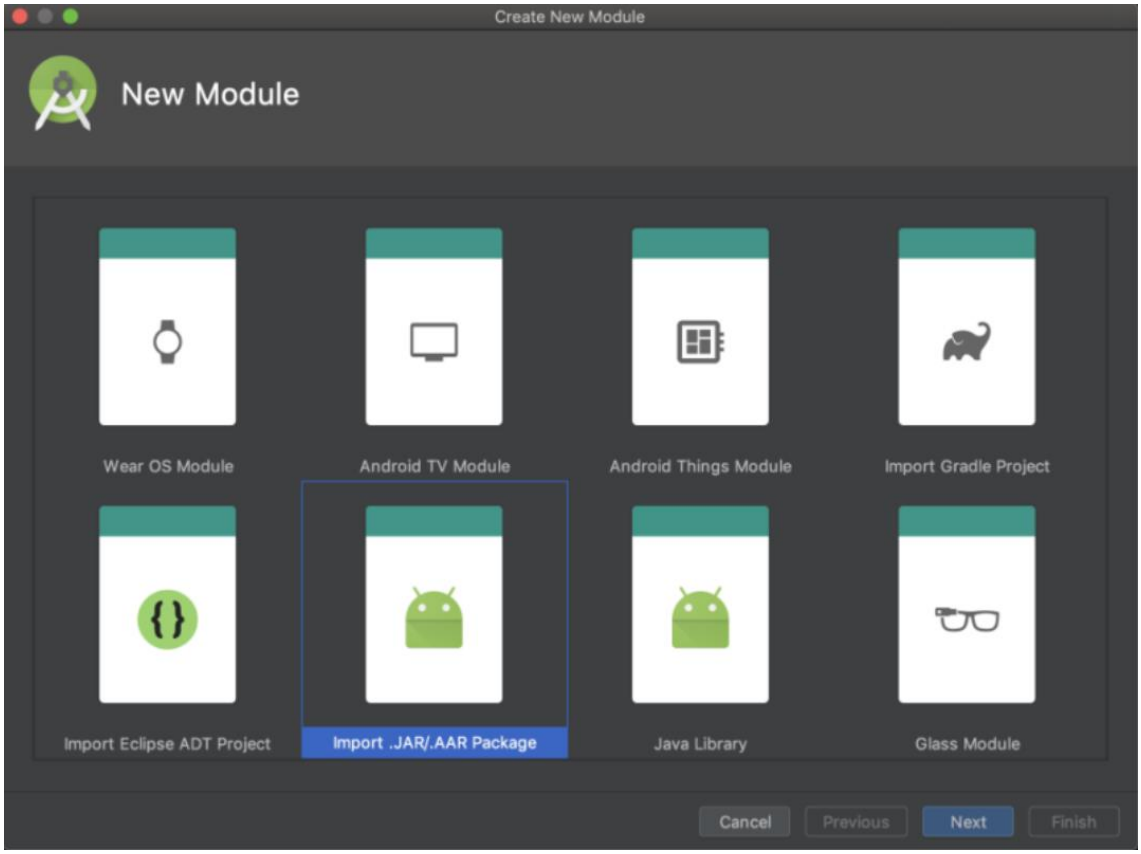
资源：SunmiRFID-SDK-release-v1.0.0.aar
参考：UHFDemo.zip

2.2.2. 导入 AAR 包

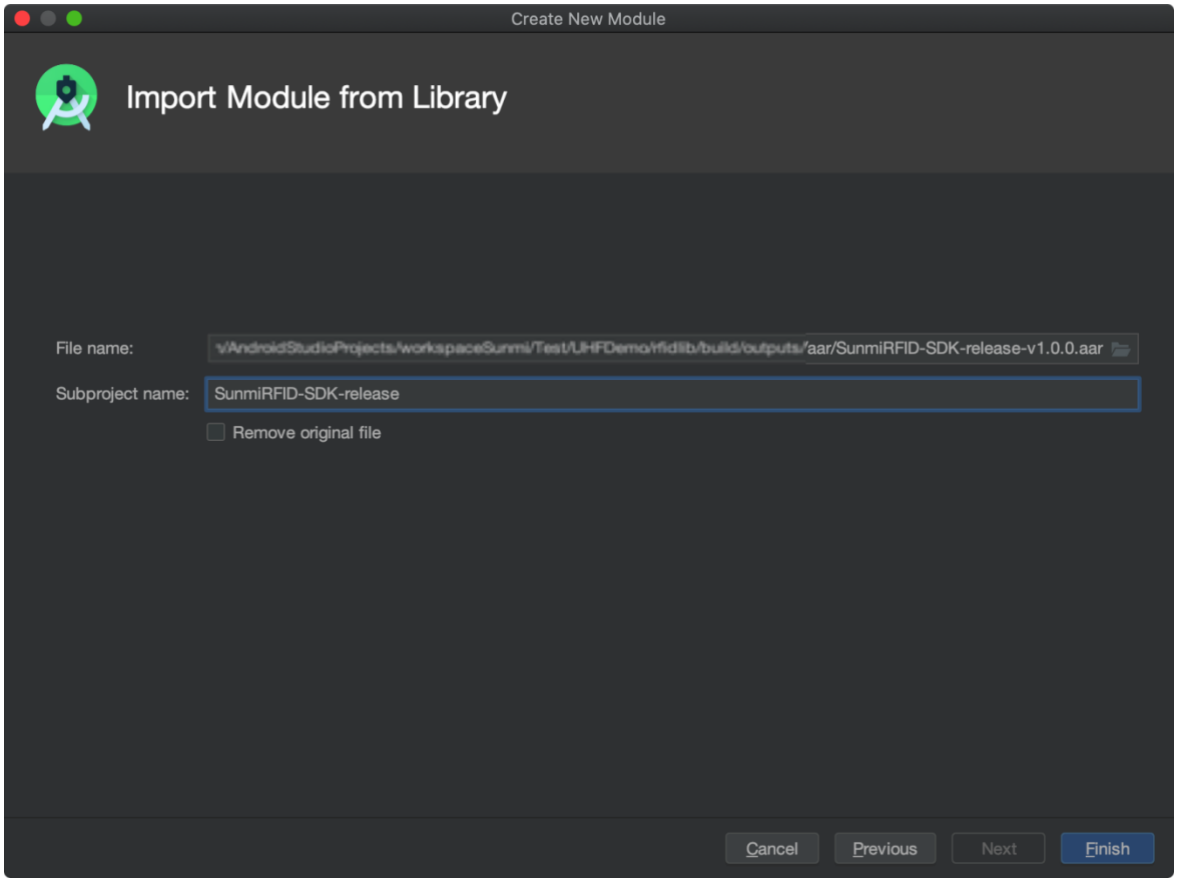
打开项目后: File -> New -> New Module…



在新的对话框:New Module -> Import .JAR/.AAR Package -> Next



下一步:Import Module from Library -> 选择 aar 文件夹下的 SunmiRFID-SDK-release-vX.X.X.aar 文件 -> Finish

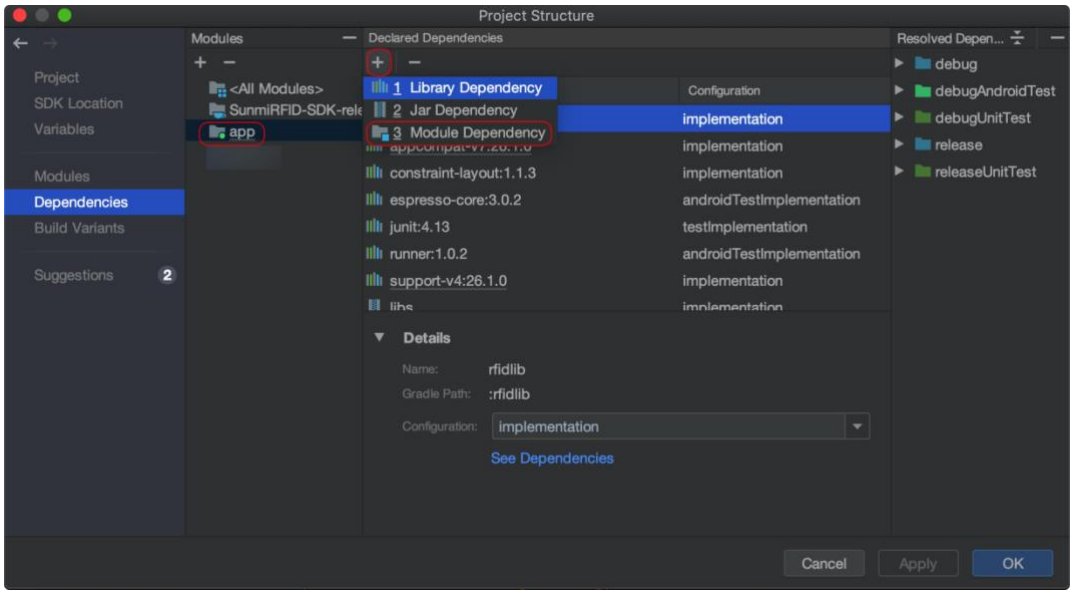


完成：

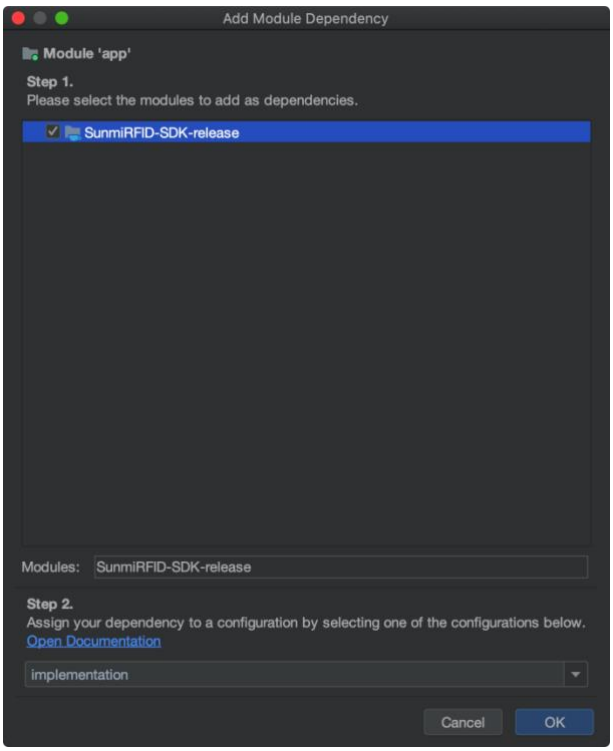
2.2.3. app 模块引用

点击-打开模块设置：

设置对话框：选择“app” -> 点击“+”号 -> 选择“Module Dependency”



显示：勾选上对应的模块（SunmiRFID-SDK_vX.X.X-release/tes）-> 点击“OK”，相关 SDK 引用成功。



2.3. RFID Manager

2.3.1. 连接服务

```
RFIDManager.getInstance().setPrintLog(true);//是否输出 log
RFIDManager.getInstance().connect(this);
```

2.3.2. 释放服务

```
RFIDManager.getInstance().disconnect();
```

2.3.3. 获取 Helper

```
RFIDManager.getInstance().getHelper();
```

2.4. Helper 接口定义说明

2.4.1. 基础信息及数据接收注册

编号	方　法
1	int **getScanModel()** 获取 RFID 类型
2	void **registerReaderCall**(**ReaderCall** call) 注册数据接收
3	void **unregisterReaderCall**()

	解注数据接收

1. 获取 RFID 类型

函数： void getScanModel()

参数： 无。

返回：

100 --> 无；

101 --> UHF R2000；

2. 注册数据接收

函数： void **registerReaderCall**(**ReaderCall** call)

参数：

call --> 接口回调实现，参考 3.1. 说明。

返回： 无。

3. 解注数据接收

函数： void **unregisterReaderCall**()

参数： 无。

返回： 无。

2.4.2. ISO18000-6C 标签盘存

编号	方　法
1	void **inventory**(byte btRepeat) 标签盘存 - 缓存模式
2	void **realTimeInventory**(byte btRepeat) 标签盘存 - 实时模式(Auto)
3	void **customizedSessionTargetInventory**(**byte** btSession, **byte** btTarget, **byte** btSL, **byte** btPhase, **byte** btPowerSave, **byte** btRepeat) 标签盘存 - 实时模式(Session)，推荐使用的盘存指令
4	void **fastSwitchAntInventory**(**byte** btA, **byte** btStayA, **byte** btB, **byte** btStayB, **byte** btC, **byte** btStayC, **byte** btD, **byte** btStayD, **byte** btInterval, **byte** btRepeat) 标签盘存 - 实时模式(Switch)，快速切换天线模式

1. 6C 标签盘存 - 缓存模式

函数： void **inventory**(**byte** btRepeat)

参数：

btRepeat --> 盘存过程重复的次数，为 0xFF 则此轮盘存时间为最短时间。 如果射频区域内只有一张标签，则此轮的盘 存约耗时为 30-50mS。一般在四通道机器上快速轮询多个天线时使用此参数值。

注意：将参数设置成 255(0xFF)时，将启动专为读少量标签设计的算法。对于少量标的应用来说，效率更高，反应更灵敏，但此参数不适合同时读取大量标签的应用。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 params 包含参数：ANT_ID、COUNT、READ_RATE、DATA_COUNT、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

缓存模式：读写器收到此命令后，进行多标签识别操作。标签数据存入读写器缓存区，使用提取缓存指令可获得标签数据，参考：2.4.5.缓存操作 说明。

2. 6C 标签盘存 - 实时模式（Auto）

函数： void realTimeInventory(byte btRepeat)

参数：

btRepeat --> 盘存过程重复的次数，为 0xFF 则此轮盘存时间为最短时间。 如果射频区域内只有一张标签，则此轮的盘 存约耗时为 30-50mS。一般在四通道机器上快速轮询多个天线时使用此参数值。

注意：将参数设置成 255(0xFF)时，将启动专为读少量标签设计的算法。对于少量标的应用来说，效率更高，反应更灵敏，但此参数不适合同时读取大量标签的应用。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 params 包含参数：READ_RATE、DATA_COUNT、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 标签回调 - 参考 3.1.-2.，其中参数 tag 包含参数：ANT_ID、TAG_PC、TAG_EPC、TAG_RSSI、TAG_READ_COUNT、TAG_FREQ、TAG_TIME，参数类型参考 附表 4.2. 说明。
3. 失败回调 - 参考 3.1.-3. 说明。

说明：

进行多标签识别操作，标签数据实时上传， 不存入读写器缓存区。此命令一轮盘存耗时较长，适用于大批量标签读取。

由于硬件为双 CPU 架构，主 CPU 负责轮询标签，副 CPU 负责数据管理。轮询标签和发送数据并行，互不占用对方的时间，因此串口的数据传输不影响读写器工作的效率。

3. 6C 标签盘存 - 实时模式（Session）

函数： void customizedSessionTargetInventory(byte btSession, byte btTarget, byte btSL, byte btPhase, byte btPowerSave, byte btRepeat)

参数：

btSession --> 指定盘存的 session。00 为 S0，01 为 S1，02 为 S2，03 为 S3。
btTarget --> 指定盘存的 Inventoried Flag，00 为 A，01 为 B。
btSL --> Select Flag；范围：00,01,02,03。
btPhase --> 相位值：00 为关闭此功能，01 为打开此功能。
btPowerSave --> 节能，省电模式：只能设置在 0 到 255 之间。
btRepeat --> 盘存过程重复的次数。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 params 包含参数：READ_RATE、DATA_COUNT、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。

2. 标签回调 - 参考 3.1.-2.，其中参数 tag 包含参数：ANT_ID、TAG_PC、TAG_EPC、TAG_RSSI、TAG_READ_COUNT、TAG_FREQ、TAG_TIME，参数类型参考 附表 4.2. 说明。
3. 失败回调 - 参考 3.1.-3. 说明。

说明：

按照指定的 session 和 inventoried flag 进行多标签识别操作。标签数据实时上传，不存入读写器缓存区。此命令一轮盘存耗时短， 普通盘存推荐使用此命令 S1 模式。

4. 6C 标签盘存 - 实时模式 (Switch)，快速切换天线模式

函数： void **fastSwitchAntInventory**(**byte** btA, **byte** btStayA, **byte** btB, **byte** btStayB, **byte** btC, **byte** btStayC, **byte** btD, **byte** btStayD, **byte** btInterval, **byte** btRepeat)

参数：

btA --> 首先轮询的天线 (00 - 03)，天线号大于三则表示不轮询。
btStayA/B/C/D --> 天线重复轮询的次数。每个天线可单独配置。
btB --> 第二个轮询的天线 (00 - 03)，天线号大于三则表示不轮询。
btC --> 第三个轮询的天线 (00 - 03)，天线号大于三则表示不轮询。
btD --> 第四个轮询的天线 (00 - 03)，天线号大于三则表示不轮询。
btInterval --> 天线间的休息时间。单位是 mS。休息时无射频输出，可降低功耗。
btRepeat --> 盘存过程重复的次数。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 params 包含参数：COMMAND_DURATION、DATA_COUNT、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 标签回调 - 参考 3.1.-2.，其中参数 tag 包含参数：ANT_ID、TAG_PC、TAG_EPC、TAG_RSSI、TAG_READ_COUNT、TAG_FREQ、TAG_TIME、TAG_ANT_1、TAG_ANT_2、TAG_ANT_、TAG_ANT_4，参数类型参考 附表 4.2. 说明。
3. 失败回调 - 参考 3.1.-3. 说明。

说明：

进行多标签识别操作，实时上传标签的数据，并且同时也会存入读写器缓存区。读写器会依次按照 A->H 的顺序自动切换天线。如 果在射频区域内没有标签，或者只有一两张标签在射频区域内，则每个天线平均耗时 30mS 左右。如果标签数量比较多，则耗时时间会相应增加。非常适合需要高速切换多个天线识别标签的应用。

2.4.3. ISO18000-6C 标签操作

编号	方 法
1	void **readTag**(**byte** btMemBank, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryPassWord) 标签操作 - 读标签
2	void **writeTag**(**byte**[] btAryPassWord, **byte** btMemBank, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryData) 标签操作 - 写标签
3	void **lockTag**(**byte**[] btAryPassWord, **byte** btMemBank, **byte** btLockType) 标签操作 - 锁标签
4	void **killTag**(**byte**[] btAryPassWord) 标签操作 - 销毁标签

5	void **setAccessEpcMatch**(**byte** btEpcLen, **byte**[] btAryEpc) 标签操作 - 设置访问 EPC 匹配（EPC 匹配有效，直到下一次刷新）
6	void **cancelAccessEpcMatch**() 标签操作 - 清除访问 EPC 匹配
7	void **getAccessEpcMatch**() 标签操作 - 获取 EPC 匹配
8	void **setImpinjFastTid**(**boolean** blnOpen, **boolean** blnSave) 标签操作 - 设置 FastTID（仅对 Impinj Monza 标签的部分型号有效）
9	void **getImpinjFastTid**() 标签操作 - 查询 FastTID

1. 6C 标签操作 - 读标签

函数： void **readTag**(**byte** btMemBank, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryPassWord)

参数：

btMemBank --> 标签存储区域：0x00:RESERVED；0x01:EPC；0x02:TID；0x03:USER。

btWordAdd --> 读取数据首地址，取值范围请参考标签规格。

btWordCnt --> 读取数据长度，字长，WORD(16 bits)长度；取值范围请参考标签规格书。

btAryPassWord --> 标签访问密码，4 字节。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 **params** 包含参数：**TAG_PC、TAG_CRC、TAG_EPC、TAG_DATA、TAG_DATA_LEN、ANT_ID、TAG_READ_COUNT、START_TIME、END_TIME**，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

需要先 2.4.3.-5.设置访问 EPC 匹配 操作，相同 EPC 的标签，若读取的数据不相同，则被视为不同的标签。

2. 6C 标签操作 - 写标签

函数： void **writeTag**(**byte**[] btAryPassWord, **byte** btMemBank, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryData)

参数：

btAryPassWord --> 标签访问密码，4 字节。

btMemBank --> 标签存储区域：0x00:RESERVED；0x01:EPC；0x02:TID；0x03:USER。

btWordAdd --> 写数据首地址，取值范围请参考标签规格；写入 EPC 存储区域一般从 02 开始，该区域前四个字节存放 PC+CRC。

btWordCnt --> 写数据长度，字长，WORD(16 bits)长度；取值范围请参考标签规格书。

btAryData --> 写入数据，长度为 btWordCnt * 2。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 **params** 包含参数：**TAG_PC、TAG_CRC、TAG_EPC、ANT_ID、TAG_READ_COUNT、START_TIME、END_TIME**，参数类型参考 附表 4.2. 说明。

2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

需要先 2.4.3.-5.设置访问 EPC 匹配 操作，相同 EPC 的标签，若读取的数据不相同，则被视为不同的标签。

3. 6C 标签操作 - 锁标签

函数： void **lockTag**(**byte**[] btAryPassWord, **byte** btMemBank, **byte** btLockType)

参数：

btAryPassWord --> 标签访问密码，4 字节。

btMemBank --> 锁操作数据区域：0x01:User Memory, 0x02:TID Memory, 0x03:EPC Memory, 0x04:Access Password, 0x05:Kill Password。

btLockType --> 锁操作类型：0x00：开放，0x01：锁定，0x02：永久开放，0x03：永久锁定。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 **params** 包含参数：**TAG_PC、TAG_CRC、TAG_EPC、ANT_ID、TAG_READ_COUNT、START_TIME、END_TIME**，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

需要先 2.4.3.-5.设置访问 EPC 匹配 操作，相同 EPC 的标签，若读取的数据不相同，则被视为不同的标签。

4. 6C 标签操作 - 销毁标签

函数： void **killTag**(**byte**[] btAryPassWord)

参数：

btAryPassWord --> 标签访问密码，4 字节。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 **params** 包含参数：**TAG_PC、TAG_CRC、TAG_EPC、ANT_ID、TAG_READ_COUNT、START_TIME、END_TIME**，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

需要先 2.4.3.-5.设置访问 EPC 匹配 操作，相同 EPC 的标签，若读取的数据不相同，则被视为不同的标签。

5. 6C 标签操作 - 设置访问 EPC 匹配（EPC 匹配有效，直到下一次刷新）

函数： void **setAccessEpcMatch**(**byte** btEpcLen, **byte**[] btAryEpc)

参数：

btEpcLen --> EPC 长度。

btAryEpc --> EPC 号，由 EpcLen 个字节组成。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 params 包含参数：START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

相同 EPC 的标签，若读取的数据不相同，则被视为不同的标签。

6. 6C 标签操作 - 清除访问 EPC 匹配

函数： void cancelAccessEpcMatch()

参数： 无。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 params 包含参数：START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

相同 EPC 的标签，若读取的数据不相同，则被视为不同的标签。

7. 6C 标签操作 - 获取 EPC 匹配

函数： void getAccessEpcMatch()

参数： 无。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 params 包含参数：START_TIME、TAG_ACCESS_EPC_MATCH、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

相同 EPC 的标签，若读取的数据不相同，则被视为不同的标签。

8. 6C 标签操作 - 设置 Fast TID

函数： void setImpinjFastTid(boolean blnOpen, boolean blnSave)

参数：

blnOpen --> FastTID 开关状态。

blnSave --> 将配置保存至内部的 Flash 中，断电不丢失。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 params 包含参数：START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

此功能仅对 Impinj Monza 标签的部分型号有效。

此功能在识别 EPC 的同时识别 TID，因此大大提高了读 TID 的效率。

打开此功能后，特定型号的标签会在盘存的过程中将 TID 打包到 EPC 中。因此，标签的 PC 会被修改，原来的 PC+EPC 变为:修改后的 PC + EPC + (EPC 的 CRC) + TID。

如果在识别 TID 的过程中出现错误，则上传原来的 PC+EPC。 ★如不需要此功能请将其关闭，避免不必要的时间消耗。

9. 6C 标签操作 - 查询 Fast TID

函数： void **getImpinjFastTid**()

参数： 无。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1. 说明，其中参数 **params** 包含参数：START_TIME、TAG_MONZA_STATUS、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1. - 3. 说明。

说明：

无。

2.4.4. ISO18000-6B 标签盘存与标签操作

编号	方 法
1	void **iso180006BInventory**() 标签盘存 - 实时模式
2	void **iso180006BReadTag**(**byte**[] btAryUID, **byte** btWordAdd, **byte** btWordCnt) 标签操作 - 读标签
3	void **iso180006BWriteTag**(**byte**[] btAryUID, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryBuffer) 标签操作 - 写标签
4	void **iso180006BLockTag**(**byte**[] btAryUID, **byte** btWordAdd) 标签操作 - 锁标签
5	void **iso180006BQueryLockTag**(**byte**[] btAryUID, **byte** btWordAdd) 标签操作 - 锁标签查询

1. 6B 标签盘存 - 实时模式

函数： void **iso180006BInventory**()

参数： 无。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 **params** 包含参数：ANT_ID、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 标签回调 - 参考 3.1.-2.，其中参数 **tag** 包含参数：ANT_ID、TAG_UID、TAG_READ_COUNT、TAG_TIME，参数类型参考 附表 4.2. 说明。
3. 失败回调 - 参考 3.1.-3. 说明。

说明：

进行多标签识别操作，标签数据实时上传， 不存入读写器缓存区。此命令一轮盘存耗时较长，适用于大批量标签读取。

由于硬件为双 CPU 架构，主 CPU 负责轮询标签，副 CPU 负责数据管理。轮询标签和发送数据并行，互不占用对方的时间，因此串口的数据传输不影响读写器工作的效率。

2. 6B 标签操作 - 读标签

函数： void **iso180006BReadTag**(**byte**[] btAryUID, **byte** btWordAdd, **byte** btWordCnt)

参数：

btAryUID --> 被操作标签的 UID，8 字节。

btWordAdd --> 读取数据开始地址。

btWordCnt --> 读取数据的数据长度。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 **params** 包含参数：**ANT_ID**、**TAG_DATA**、**TAG_DATA_LEN**、**START_TIME**、**END_TIME**，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1.-3. 说明。

说明：

无。

3. 6B 标签操作 - 写标签

函数： void **iso180006BWriteTag**(**byte**[] btAryUID, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryBuffer)

参数：

btAryUID --> 被操作标签的 UID，8 字节。

btWordAdd --> 写数据开始地址。

btWordCnt --> 写数据的数据长度。

btAryBuffer --> 数据。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 **params** 包含参数：**ANT_ID**、**TAG_DATA_LEN**、**START_TIME**、**END_TIME**，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1.-3. 说明。

说明：

可以一次性写入多个字节。一旦写入某个字节出现错误，此命令不会继续写入后面的数据。

同时命令返回已经成功写入的字节数。

4. 6B 标签操作 - 锁标签

函数： void **iso180006BLockTag**(**byte**[] btAryUID, **byte** btWordAdd)

参数：

btAryUID --> 被操作标签的 UID，8 字节。

btWordAdd --> 锁定地址。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 **params** 包含参数：**ANT_ID**、**TAG_STATUS**（**0x00**:锁操作成功，**0xFE**:已锁状态，**0xFF**:锁标签失败）、**START_TIME**、**END_TIME**，参数类型参考 附表 4.2. 说明。

2. 失败回调 - 参考 3.1.-3. 说明。

说明：

无。

5. 6B 标签操作 - 锁标签查询

函数： void iso180006BQueryLockTag(byte[] btAryUID, byte btWordAdd)

参数：

btAryUID --> 被操作标签的 UID，8 字节。

btWordAdd --> 查询地址。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 params 包含参数：ANT_ID、TAG_STATUS（0x00:未锁状态，0xFE:已锁状态）、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1.-3. 说明。

说明：

无。

2.4.5. ISO18000-6C 缓存操作

编号	方　法
1	void getInventoryBuffer() 缓存操作 - 获取缓存标签
2	void getAndResetInventoryBuffer() 标签操作 - 获取缓存标签并重置缓存
3	void getInventoryBufferTagCount() 标签操作 - 获取缓存标签数量
4	void resetInventoryBuffer() 标签操作 - 重置缓存

1. 缓存操作 - 获取缓存标签

函数： void getInventoryBuffer()

参数： 无。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 params 包含参数：TAG_PC、TAG_CRC、TAG_EPC、ANT_ID、TAG_RSSI、TAG_READ_COUNT、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1.-3. 说明。

说明：

将缓存标签数据上传，数量等于缓存中的标签数量(无重复数据)。

2. 缓存操作 - 获取缓存标签并重置缓存

函数： void getAndResetInventoryBuffer()

参数： 无。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 params 包含参数：TAG_PC、TAG_CRC、TAG_EPC、ANT_ID、TAG_RSSI、TAG_READ_COUNT、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1.-3. 说明。

说明：

将缓存标签数据上传，数量等于缓存中的标签数量(无重复数据)，并清除缓存标签。

3. 缓存操作 - 获取缓存标签数量

函数： void getInventoryBufferTagCount()

参数： 无。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 params 包含参数：COUNT、START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1.-3. 说明。

说明：

无。

4. 缓存操作 - 重置缓存

函数： void resetInventoryBuffer()

参数： 无。

回调：

需要注册数据接收，参考 2.4.1.-2. 和 2.4.1.-3. 说明。

1. 成功回调 - 参考 3.1. - 1，其中参数 params 包含参数：START_TIME、END_TIME，参数类型参考 附表 4.2. 说明。
2. 失败回调 - 参考 3.1.-3. 说明。

说明：

无。

3. 回调说明

3.1. ReaderCall - 说明

编号	方 法
1	void **onSuccess**(**byte** cmd, **DataParameter** params) throws RemoteException 成功回调 - 操作成功
2	void **onTag**(**byte** cmd, **byte** state, **DataParameter** tag) throws RemoteException 标签操作 - 标签回调
3	void **onFiled**(**byte** cmd, **byte** errorCode, **String** msg) throws RemoteException 失败回调 - 操作失败

1. 成功回调 - 操作成功

函数： void **onSuccess**(**byte** cmd, **DataParameter** params) throws RemoteException

参数：

cmd --> 操作类型，参考 2.4.方法说明 或 附表 4.1.

params --> 数据参数，参考 2.4.方法说明 或 参数类型对照表 4.2.，示例：6C 标签盘存结束获取识读标签数量，params.getInt(ParamCts.DATA_COUNT, 0)

说明：

将缓存标签数据上传，数量等于缓存中的标签数量(无重复数据)。

2. 标签操作 - 标签回调

函数： void **onTag**(**byte** cmd, **byte** state, **DataParameter** tag) throws RemoteException

参数：

cmd --> 操作类型，参考 2.4.方法说明 或 附表 4.1.

state --> 标签状态：

ParamCts.FOUND_TAG,(0x01) - 新标签；

ParamCts.UPDATE_TAG,(0x02) - 标签更新。

tag --> 标签参数，具体可参考 2.4.方法说明 与 4.2.参数类型对照表，示例：6C 标签获取 EPC 区数据，tag.getString(ParamCts.TAG_EPC)；

说明：

盘存的标签数据，由此接口返回。

3. 失败回调 - 操作失败

函数： void **onFiled**(**byte** cmd, **byte** errorCode, **String** msg) throws RemoteException

参数：

cmd --> 操作类型，参考 2.4.方法说明 或 附表 4.1.

errorCode --> 错误码，参考 附表 4.3.

msg --> 错误信息。

说明：

无。

3.2. 广播事件

Action	Description	Parameter(参考 4.2.)
com.sunmi.rfid.unFoundReader	未找到相关设备	
com.sunmi.rfid.onLostConnect	连接断开	
com.sunmi.rfid.batteryLowElec	电池电量过低	ParamCts.BATTERY_REMAINING_PERCENT

4. 附录

4.1. 操作类型对照表

Parameter (CMD)	Value	Description
CMD.INVENTORY	0x80	6C 标签盘存（缓存模式）
CMD.READ_TAG	0x81	6C 标签操作-读标签
CMD.WRITE_TAG	0x82	6C 标签操作-写标签
CMD.LOCK_TAG	0x83	6C 标签操作-锁标签
CMD.KILL_TAG	0x84	6C 标签操作-销毁标签
CMD.SET_ACCESS_EPC_MATCH	0x85	6C 标签操作-设置访问 EPC 匹配 6C 标签操作-清除访问 EPC 匹配
CMD.GET_ACCESS_EPC_MATCH	0x86	6C 标签操作-获取访问 EPC 匹配
CMD.REAL_TIME_INVENTORY	0x89	6C 标签盘存-实时模式（Auto）
CMD.FAST_SWITCH_ANT_INVENTORY	0x8A	6C 标签盘存-实时模式（Switch）
CMD.CUSTOMIZED_SESSION_TARGET_INVENTORY	0x8B	6C 标签盘存-实时模式（Session）
CMD.SET_IMPINJ_FAST_TID	0x8C	6C 标签操作-设置 Fast TID
CMD.SET_AND_SAVE_IMPINJ_FAST_TID	0x8D	6C 标签操作-设置 Fast TID 并保存
CMD.GET_IMPINJ_FAST_TID	0x8E	6C 标签操作-查询 Fast TID
CMD.ISO18000_6B_INVENTORY	0xB0	6B 标签盘存-实时
CMD.ISO18000_6B_READ_TAG	0xB1	6B 标签操作-读标签
CMD.ISO18000_6B_WRITE_TAG	0xB2	6B 标签操作-写标签
CMD.ISO18000_6B_LOCK_TAG	0xB3	6B 标签操作-锁标签
CMD.ISO18000_6B_QUERY_LOCK_TAG	0xB4	6B 标签操作-锁标签查询
CMD.GET_INVENTORY_BUFFER	0x90	6C 缓存操作-获取缓存标签
CMD.GET_AND_RESET_INVENTORY_BUFFER	0x91	6C 缓存操作-获取缓存标签并重置缓存
CMD.GET_INVENTORY_BUFFER_TAG_COUNT	0x92	6C 缓存操作-获取缓存标签数量
CMD.RESET_INVENTORY_BUFFER	0x93	6C 缓存操作-重置缓存

4.2. 参数类型对照表

Parameter (ParamCts)	Type	Description
广播参数部分		
BATTERY_REMAINING_PERCENT	int	电池电量（0-100）
标签盘存部分		
ANT_ID	byte	当前天线 ID（0x00-0x03）
COUNT	int	缓存标签数量
READ_RATE	int	标签识别率
DATA_COUNT	int	读取标签数量
START_TIME	long	操作开始时间，单位：毫秒
END_TIME	long	操作结束时间，单位：毫秒
COMMAND_DURATION	int	6C 盘存(Switch 模式)，总共消耗的时间，单位：毫秒

标签部分		
TAG_UID	String	6B 标签 UID 数据
TAG_PC	String	6C 标签 PC 数据
TAG_EPC	String	6C 标签 EPC 数据
TAG_CRC	String	6C 标签 CRC 数据
TAG_RSSI	String	6C 标签 RSSI 数据
TAG_READ_COUNT	int	标签识读次数
TAG_FREQ	String	标签识读射频频率
TAG_TIME	long	标签最后更新时间，单位：毫秒
TAG_DATA	String	标签数据（标签操作）
TAG_DATA_LEN	int	标签数据长度（标签操作）
TAG_ANT_1	int	6C 盘存(Switch 模式)-标签被 Ant1 识别次数
TAG_ANT_2	int	6C 盘存(Switch 模式)-标签被 Ant2 识别次数
TAG_ANT_3	int	6C 盘存(Switch 模式)-标签被 Ant3 识别次数
TAG_ANT_4	int	6C 盘存(Switch 模式)-标签被 Ant4 识别次数
TAG_ACCESS_EPC_MATCH	String	6C 标签操作 - 访问 EPC 匹配
TAG_MONZA_STATUS	byte	6C 标签操作 - Fast TID
TAG_STATUS	byte	6B 标签锁状态

4.3. 错误码对照表

Num	Code	Name	Description
1	0x10	success	执行成功完成
2	0x11	fail	命令执行失败
3	0x20	mcu_reset_error	CPU 复位错误
4	0x21	cw_on_error	打开 CW 错误
5	0x22	antenna_missing_error	天线未连接
6	0x23	write_flash_error	写 Flash 错误
7	0x24	read_flash_error	读 Flash 错误
8	0x25	set_output_power_error	设置发射功率错误
9	0x31	tag_inventory_error	盘存标签错误
10	0x32	tag_read_error	读标签错误
11	0x33	tag_write_error	写标签错误
12	0x34	tag_lock_error	锁定标签错误
13	0x35	tag_kill_error	灭活标签错误
14	0x36	no_tag_error	无可操作标签错误
15	0x37	inventory_ok_but_access_fail	成功盘存但访问失败
16	0x38	buffer_is_empty_error	缓存为空
17	0x3C	nxp_custom_command_fail	NXP 芯片自定义指令失败
18	0x40	access_or_password_error	访问标签错误或访问密码错误
19	0x41	parameter_invalid	无效的参数
20	0x42	parameter_invalid_wordCnt_too_long	wordCnt 参数超过规定长度
21	0x43	parameter_invalid_membank_out_of_range	MemBank 参数超出范围

22	0x44	parameter_invalid_lock_region_out_of_range	Lock 数据区参数超出范围
23	0x45	parameter_invalid_lock_action_out_of_range	LockType 参数超出范围
24	0x46	parameter_reader_address_invalid	读写器地址无效
25	0x47	parameter_invalid_antenna_id_out_of_range	Antenna_id 超出范围
26	0x48	parameter_invalid_output_power_out_of_range	输出功率参数超出范围
27	0x49	parameter_invalid_frequency_region_out_of_range	射频规范区域参数超出范围
28	0x4A	parameter_invalid_baudrate_out_of_range	波特率参数超出范围
29	0x4B	parameter_beeper_mode_out_of_range	蜂鸣器设置参数超出范围
30	0x4C	parameter_epc_match_len_too_long	EPC 匹配长度越界
31	0x4D	parameter_epc_match_len_error	EPC 匹配长度错误
32	0x4E	parameter_invalid_epc_match_mode	EPC 匹配参数超出范围
33	0x4F	parameter_invalid_frequency_range	频率范围设置参数错误
34	0x50	fail_to_get_RN16_from_tag	无法接收标签的 RN16
35	0x51	parameter_invalid_drm_mode	DRM 设置参数错误
36	0x52	pll_lock_fail	PLL 不能锁定
37	0x53	rf_chip_fail_to_response	射频芯片无响应
38	0x54	fail_to_achieve_desired_output_power	输出达不到指定的输出功率
39	0x55	copyright_authentication_fail	版权认证未通过
40	0x56	spectrum_regulation_error	频谱规范设置错误
41	0x57	output_power_too_low	输出功率过低
42	0xEE	fail_to_get_rf_port_return_loss	测量回波损耗失败
43	0x03	un_check_reader	未检测到 RFID 设备

4.4. RSSI 参数对应表

RSSI 参数	对应信号强度（对数）	RSSI 参数	对应信号强度（对数）
98(0x62)	-31dBm	64(0x40)	-65dBm
97(0x61)	-32dBm	63(0x3F)	-66dBm
96(0x60)	-33dBm	62(0x3E)	-67dBm
95(0x5F)	-34dBm	61(0x3D)	-68dBm
94(0x5E)	-35dBm	60(0x3C)	-69dBm
93(0x5D)	-36dBm	59(0x3B)	-70dBm
92(0x5C)	-37dBm	58(0x3A)	-71dBm
91(0x5B)	-38dBm	57(0x39)	-72dBm
90(0x5A)	-39dBm	56(0x38)	-73dBm
89(0x59)	-40dBm	55(0x37)	-74dBm
88(0x58)	-41dBm	54(0x36)	-75dBm
87(0x57)	-42dBm	53(0x35)	-76dBm
86(0x56)	-43dBm	52(0x34)	-77dBm
85(0x55)	-44dBm	51(0x33)	-78dBm
84(0x54)	-45dBm	50(0x32)	-79dBm
83(0x53)	-46dBm	49(0x31)	-80dBm

82(0x52)	-47dBm	48(0x30)	-81dBm
81(0x51)	-48dBm	47(0x2F)	-82dBm
80(0x50)	-49dBm	46(0x2E)	-83dBm
79(0x4F)	-50dBm	45(0x2D)	-84dBm
78(0x4E)	-51dBm	44(0x2C)	-85dBm
77(0x4D)	-52dBm	43(0x2B)	-86dBm
76(0x4C)	-53dBm	42(0x2A)	-87dBm
75(0x4B)	-54dBm	41(0x29)	-88dBm
74(0x4A)	-55dBm	40(0x28)	-89dBm
73(0x49)	-56dBm	39(0x27)	-90dBm
72(0x48)	-57dBm	38(0x26)	-91dBm
71(0x47)	-58dBm	37(0x25)	-92dBm
70(0x46)	-59dBm	36(0x24)	-93dBm
69(0x45)	-60dBm	35(0x23)	-94dBm
68(0x44)	-61dBm	34(0x22)	-95dBm
67(0x43)	-62dBm	33(0x21)	-96dBm
66(0x42)	-63dBm	32(0x20)	-97dBm
65(0x41)	-64dBm	31(0x1F)	-98dBm