

蓝牙背夹API说明

一、开发环境搭建

- 1.1 根据开发资料中“集成开发需要文件”文件夹内的jar文件完成开发环境搭建。
- 1.2 在Android项目下新建“myjar”文件夹，将SR_ANDROID_JAR.jar及SR_UHF_JAR.jar拷贝至文件夹内，右键项目构建路径，配置构建路径，添加jar包。
- 1.3 清理,刷新项目。

二、程序流程说明

启动Android程序 => 扫描蓝牙设备 => 连接蓝牙设备 => 连接并且注册服务成功 => 进行操作 => 操作结束 => 停止读标 => 关闭蓝牙连接 => 退出程序

三、开发说明

3.1 定义对象

```
public static Reader Readercontroller;
```

3.2 实例化对象

```
Readercontroller = new Reader("BlueTooth",this);
```

3.3 注册回调使能

```
Readercontroller.SetCallBack(this);//注册回调,循环读标数据通过回调传到method函数
```

3.4 开启定时下发心跳

```
ReaderController.SetHeartThread(true);
```

3.5 实现 MultiLableCallBck 接口

```
public class MainHandleActivity extends Activity
    implements MultiLableCallBack
```

添加未实现的方法

```
@Override
public void method(byte[] data)//接收 常规群读&带密码群读 标签数据。数据解析见5.1
```

```
@Override
public void CmdRespond(String[] data)//接收 温度标签读取&带User群读 标签数据。数据解析见5.2
```

```
@Override
public void BlueToothBtnNew(int state)//蓝牙手柄按键回调接口(0:停止读标 1:开始读标)
```

```
@Override
public void BlueToothVoltageNew(int voltage, byte BAT_STATUS,
    byte WORK_STATUS, int Voltseg1, int Voltseg0)//蓝牙手柄电压回调
//此处传出手柄电池状态、RFID模块工作状态、手柄电池电量信息，处理方式参见源码
```

```
@Override
public void SdkActionResult(String ResultStr)//蓝牙设备连接结果接口。数据解析见5.3
```

注:Android 10版本手机需开启GPS权限并打开GPS功能才可扫描蓝牙。

安卓蓝牙权限

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

GPS权限

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

四、函数说明

4.1 扫描蓝牙设备

接口函数调用示例

```
Readercontroller._bles.ScanTime = 6000;//设置扫描时间；单位为ms
Readercontroller._bles.scanBle();//执行扫描
```

- Readercontroller._bles.isScanFinsh为扫描是否完成标志:为true时表示扫描结束，为false表示正在扫描中

得到当前扫描到蓝牙对象

```
List<BleDevice> BleList = ReaderController._bles.GetBleList();
```

- 两次扫描蓝牙设备间隔时间建议>=6s

4.2 停止扫描蓝牙设备

```
ReaderController._bles.StopsCanBle();
```

4.3 连接蓝牙设备

接口函数调用示例

```
Readercontroller._bles.ConnectDev(bleDevice);
```

4.4 与蓝牙设备断开连接

接口函数调用示例

```
Readercontroller._bles.disconnect();
```

4.5 获取模块版本

接口函数调用示例

```
Ware mWare = new Ware(CommandType.GET_FIRMWARE_VERSION, 0, 0, 0);
Boolean ret = ReaderController.UHF_CMD(CommandType.GET_FIRMWARE_VERSION, mWare);
```

返回值说明

- 为true:获取成功; 为false:获取失败
如为true,则版本为mWare.major_version + “.” + mWare.minor_version + “.” + mWare.revision_version

4.6 开始循环读标

4.6.1 常规群读

接口函数调用示例

```
Multi_query_epc mMulti_query_epc = new Multi_query_epc();
mMulti_query_epc.query_total = 0;
ReaderController.UHF_CMD(CommandType.MULTI_QUERY_TAGS_EPC, mMulti_query_epc);
```

4.6.2 带密码群读

接口函数调用示例

```
Multi_query_epc mMulti_query_epc = new Multi_query_epc();
mMulti_query_epc.query_total = 0;
mMulti_query_epc.inventory_type = 0x06;
mMulti_query_epc.access_pwd = DemoConfig.AccessPwd;
ReaderController.UHF_CMD(CommandType.MULTI_QUERY_TAGS_EPC, mMulti_query_epc);
```

参数说明

参数名	必选	类型	说明
query_total	是	int	为0即可
inventory_type	是	byte	群读方式;0x06表示为带密码读标
access_pwd	是	String	标签访问密码

4.6.3 带User群读

接口函数调用示例(user起始地址<=255)

```
Multi_query_epc mMulti_query_epc = new Multi_query_epc();
mMulti_query_epc.query_total = 0;
mMulti_query_epc.inventory_type = 0x08;
mMulti_query_epc.access_pwd = DemoConfig.AccessPwd;
mMulti_query_epc.user_startAddr = (byte)DemoConfig.UserStartAddr;
mMulti_query_epc.user_Len = (byte)DemoConfig.UserLen;
ReaderController.UHF_CMD(CommandType.MULTI_QUERY_TAGS_EPC, mMulti_query_epc);
```

接口函数调用示例(user起始地址>255)

```
Multi_query_epc mMulti_query_epc = new Multi_query_epc();
mMulti_query_epc.query_total = 0;
mMulti_query_epc.inventory_type = 0x09;
mMulti_query_epc.access_pwd = DemoConfig.AccessPwd;
mMulti_query_epc.Int_User_startAddr = DemoConfig.UserStartAddr;
mMulti_query_epc.user_Len = (byte)DemoConfig.UserLen;
ReaderController.UHF_CMD(CommandType.MULTI_QUERY_TAGS_EPC, mMulti_query_epc);
```

参数说明

参数名	必选	类型	说明
query_total	是	int	为0即可
inventory_type	是	byte	群读方式
access_pwd	是	String	标签访问密码
user_startAddr	是	byte	User数据读取起始地址
Int_User_startAddr	是	int	User数据读取起始地址
user_Len	是	byte	User数据读取长度

4.6.4 读温度标签

接口函数调用示例

```
Multi_TempTags mMulti_temptags = new Multi_TempTags();
mMulti_temptags.TempTagsType = 0x02;
ReaderController.UHF_CMD(CommandType.Read_TagTemp_Start, mMulti_temptags);
```

参数说明

参数名	必选	类型	说明
TempTagsType	是	byte	温度标签类型; 0x01:LTU27, 0x02:LTU3x, (byte)0x20:RFM, (byte)0x30:NMV2D

4.7 停止循环读标

4.7.1 常规群读/带密码群读/带User群读 => 停止

接口函数调用示例

```
Boolean ret = ReaderController.UHF_CMD(CommandType.STOP_MULTI_QUERY_TAGS_EPC, null);
```

返回值说明

- 为true:停止成功;为false:停止失败

4.7.2 温度标签读取 => 停止

接口函数调用示例

```
Multi_TempTags mMulti_temptags = new Multi_TempTags();
mMulti_temptags.TempTagsType = 0x02;
ReaderController.UHF_CMD(CommandType.Read_TagTemp_Stop, mMulti_temptags);
```

参数说明

参数名	必选	类型	说明
TempTagsType	是	byte	温度标签类型; 0x01:LTU27, 0x02:LTU3x, (byte)0x20:RFM, (byte)0x30:NMV2D

返回值说明

- 为true:停止成功;为false:停止失败

4.8 获取功率

接口函数调用示例

```
Power mPower = new Power();
mPower.com_type = CommandType.GET_POWER;
mPower.loop = 0;
mPower.read = 0;
mPower.write = 0;
Boolean ret = ReaderController.UHF_CMD(CommandType.GET_POWER, mPower);
```

返回值说明

- 为true:获取成功; 为false:获取失败
如为true,则功率为mPower.read

4.9 设置功率

接口函数调用示例

```
Power mPower = new Power();
mPower.com_type = CommandType.SET_POWER;
mPower.loop = 0;
mPower.read = ReadPower;
mPower.write = ReadPower;
Boolean ret = ReaderController.UHF_CMD(CommandType.SET_POWER, mPower);
```

参数说明

参数名	必选	类型	说明
ReadPower	是	int	ReadPower为你要设置的功率值。 值范围为5-30或5-25， 根据设备类型而定。

返回值说明

- 为true:设置成功; 为false:设置失败

4.10 获取频率区域

接口函数调用示例

```
Frequency_region mFrequency_region = new Frequency_region(CommandType.GET_FREQUENCY_REGION, 0, 0);
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.GET_FREQUENCY_REGION, mFrequency_region);
```

返回值说明

- 为true:获取成功; 为false:获取失败
如为true, 则频率区域根据mFrequency_region.region而定:为1时表示为China1, 为2是表示为China2, 为3时表示为Europe, 为4时表示为USA, 为5时表示为Korea, 为6时表示为Japan。

4.11 设置频率区域

接口函数调用示例

```
Frequency_region mFrequency_region = new Frequency_region(CommandType.SET_FREQUENCY_REGION, int_save,int_region_tmp);
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.SET_FREQUENCY_REGION, mFrequency_region);
```

参数说明

参数名	必选	类型	说明
int_save	是	int	保存标志; 为0时为不断电保存, 为1时为断电保存
int_region_tmp	是	int	频率区域; 为1时表示为China1, 为2是表示为China2, 为3时表示为Europe, 为4时表示为USA, 为5时表示为Korea, 为6时表示为Japan

返回值说明

- 为true:设置成功; 为false:设置失败

4.12 获取RF链路

接口函数调用示例

```
RfLink mRfLink = new RfLink();
mRfLink.com_type = CommandType.GET_RF_LINK;
mRfLink.rflink_Type = 0;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.GET_RF_LINK, mRfLink);
```

返回值说明

- 为true:获取成功; 为false:获取失败
如为true, 则RF链路根据mRfLink.rflink_Type而定, 为0时表示为DSB/FM0/40KHz, 为1时表示为PR/M4/250KHz, 为2时表示为PR/M4/300KHz, 为3时表示为DSB/FM0/400KHz。

4.13 设置RF链路

接口函数调用示例

```
RfLink mRfLink = new RfLink();
mRfLink.com_type = CommandType.SET_RF_LINK;
mRfLink.save = int_save;
mRfLink.rflink_Type = int_rflink_temp;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.SET_RF_LINK, mRfLink);
```

参数说明

参数名	必选	类型	说明
int_save	是	int	保存标志; 为0时为不断电保存, 为1时为断电保存
int_rflink_temp	是	int	RF链路; 为0时表示为DSB/FM0/40KHz , 为1时表示为PR/M4/250KHz, 为2时表示为PR/M4/300KHz, 为3时表示为DSB/FM0/400KH Z

返回值说明

- 为true:设置成功; 为false:设置失败

4.14 获取工作时间及间隔时间

接口函数调用示例

```
Multi_interval mMulti_interval = new Multi_interval();
mMulti_interval.com_type =CommandType.GET_MULTI_QUERY_TAGS_INTERVAL;
mMulti_interval.work_time = 0;
mMulti_interval.interval = 0;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.GET_MULTI_QUERY_TAGS_INTERVAL,mMulti_interval);
```

返回值说明

- 为true:获取成功; 为false:获取失败
如为true, 则工作时间为mMulti_interval.work_time, 间隔时间为mMulti_interval.interval。

4.15 设置工作时间及间隔时间

接口函数调用示例

```
Multi_interval mMulti_interval = new Multi_interval();
mMulti_interval.com_type = CommandType.SET_MULTI_QUERY_TAGS_INTERVAL;
mMulti_interval.work_time = int_worktime_temp;
mMulti_interval.interval = int_interval_temp;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.SET_MULTI_QUERY_TAGS_INTERVAL,mMulti_interval);
```

参数说明

参数名	必选	类型	说明
int_worktime_temp	是	int	工作时间；值范围为0-65535。
int_interval_temp	是	int	间隔时间；值范围为0-65535。

返回值说明

- 为true:设置成功; 为false:设置失败

4.16 获取同时读取EPC及TID状态

接口函数调用示例

```
EPCAndTID mepcandtid = new EPCAndTID();
mepcandtid.com_type = CommandType.GET_EPCAndTID;
mepcandtid.state = 0;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.GET_EPCAndTID, mepcandtid);
```

返回值说明

- 为true:获取成功; 为false:获取失败
如为true, 则同时读取EPC及TID状态根据mepcandtid.state值而定。值为0时表示为关闭状态，为1时表示为开启状态。

4.17 设置同时读取EPC及TID状态

接口函数调用示例

```
EPCAndTID mepcandtid = new EPCAndTID();
mepcandtid.com_type = CommandType.SET_EPCAndTID;
mepcandtid.state = int_startq_temp;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.SET_EPCAndTID, mepcandtid);
```

参数说明

参数名	必选	类型	说明
int_startq_temp	是	int	同时读取EPC和TID功能状态； 为0时表示为关闭，为1时表示为开启

返回值说明

- 为true:设置成功; 为false:设置失败

4.18 获取 9200模块/-L模块 读标Rssi参数

接口函数调用示例

```
ModuleConfig moduleConfig = new ModuleConfig();
moduleConfig.com_type = CommandType.MODULE_CONFIG;
moduleConfig.subCmd = 0x01;
moduleConfig.funcCmd = 0x01;
```

```
moduleConfig.rssiState = 0x00;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.MODULE_CONFIG, moduleConfig);
```

参数说明

参数名	必选	类型	说明
subCmd	是	byte	命令类型; 0x01为9200Rssi参数
funcCmd	是	byte	操作类型; 0x01为获取, 0x00为设置
rssiState	是	byte	9200Rssi开启状态; 获取时赋初始值0x00

返回值说明

- 为true:获取成功; 为false:获取失败
如为true, 则9200 Rssi状态根据moduleConfig.rssiState值而定。值为(byte)0x55时表示为关闭状态, 为(byte)0xAA时表示为开启状态。

4.19 设置 9200模块/-L模块 读标Rssi参数

接口函数调用示例

```
ModuleConfig moduleConfig = new ModuleConfig();
moduleConfig.com_type = CommandType.MODULE_CONFIG;
moduleConfig.subCmd = 0x01;
moduleConfig.funcCmd = 0x00;
moduleConfig.rssiState = rssiState;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.MODULE_CONFIG, moduleConfig);
```

参数说明

参数名	必选	类型	说明
subCmd	是	byte	命令类型; 0x01为9200Rssi参数
funcCmd	是	byte	操作类型; 0x01为获取, 0x00为设置
rssiState	是	byte	9200Rssi开启状态; 值为 (byte)0x55时表示为关闭, 为 (byte)0xAA时表示为开启

返回值说明

- 为true:设置成功; 为false:设置失败

4.20 读指定标签数据

接口函数调用示例

```
Tags_data mTags_data = new Tags_data();
mTags_data.password = pwd;
mTags_data.FMB = int_filter;
mTags_data.filterData_len = data.length;
mTags_data.filterData = filterdata;
mTags_data.start_addr = int_offset;
mTags_data.data_len = int_length;
mTags_data.mem_bank = int_bank;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.READ_TAGS_DATA, mTags_data);
```


参数说明

参数名	必选	类型	说明
pwd	是	String	访问密码；默认为“00000000”
int_filter	是	int	过滤/指定 类型；为0时表示过滤/指定 EPC，为1时表示过滤/指定TID
data.length	是	int	过滤/指定 数据字节长度；如无过滤，则该值为0
filterdata	否	byte[]	过滤/指定 数据；如无过滤，则该值无需赋值
int_offset	是	int	读数据区域起始地址；单位为字(1字=2字节)
int_length	是	int	读数据区域长度；单位为字(1字=2字节)
int_bank	是	int	读数据区域类型；0为RFU区，1为EPC区，2为TID区，3为USER区

返回值说明

- 为true:读取成功; 为false:读取失败
如为true, 则mTags_data.data为读取数据

4.21 写指定标签数据

接口函数调用示例

```
Tags_data mTags_data = new Tags_data();
mTags_data.password = psd;
mTags_data.FMB = int_filter;
mTags_data.filterData_len = data.length;
mTags_data.filterData = filterdata;
mTags_data.start_addr = int_offset;
mTags_data.data_len = int_length;
mTags_data.mem_bank = int_bank;
mTags_data.data = data;
Boolean ret = MainActivity.ReaderController.UHF_CMD(CommandType.WRITE_TAGS_DATA, mTags_data);
```

参数说明

参数名	必选	类型	说明
pwd	是	String	访问密码；默认为“00000000”
int_filter	是	int	过滤/指定 类型；为0时表示过滤/指定 EPC，为1时表示过滤/指定TID
data.length	是	int	过滤/指定 数据字节长度；如无过滤，则该值为0
filterdata	否	byte[]	过滤/指定 数据；如无过滤，则该值无需赋值
int_offset	是	int	读数据区域起始地址；单位为字(1字=2字节)

参数名	必选	类型	说明
int_length	是	int	读数据区域长度；单位为字(1字=2字节)
int_bank	是	int	读数据区域类型；0为RFU区，1为EPC区，2为TID区，3为USER区
data	是	byte[]	写入数据

返回值说明

- 为true:写入成功; 为false:写入失败

五、回调接口函数解析说明

5.1 回调接口函数method数据解析示例

```
byte msb = data[0];
byte lsb = data[1];
int pc = (msb & 0x00ff) << 8 | (lsb & 0x00ff);
pc = (pc & 0xf800) >> 11;

byte[] epc = new byte[pc * 2];
System.arraycopy(data, 2, epc, 0, epc.length);
String str_epc = ShareData.CharToString(epc, epc.length);
str_epc = str_epc.replace(" ", "");

byte[] tid = new byte[data.length - 2 - (pc * 2) - 3];
System.arraycopy(data, 2 + (pc * 2), tid, 0, tid.length);
String str_tid = ShareData.CharToString(tid, tid.length);
str_tid = str_tid.replace(" ", "");

String str_rssi = "" + (~((short) (((data[2 + pc * 2 + tid.length] & 0xFF) << 8)
    | (data[2 + pc * 2 + 1 + tid.length] & 0xFF) - 1))) / -10.0);
```

5.2 回调接口函数CmdRespond数据解析说明

值	说明
String[0]	保留
String[1]	Cmd命令类型；“ F7” 表示为LTU27温度标签数据，“ F8” 表示为 LTU3x/RFM/NMV2D 温度标签数据，“ 79” 表示为带User群读标签数据
String[2]	标签EPC
String[3]	标签Rssi场强值
String[4]	温度标签温度值
String[5]	温度标签类型
String[6]	标签TID

5.3 回调接口函数SdkActionResult数据解析说明

先将String转换为String[]数组

```
String[] result = ResultStr.split("\\\\,");
```

值	说明
---	----

值	说明
String[0]	消息类型; 该值为BlueToothTypes.BlueToothMsgType时表示此条消息为 蓝牙连接结果相关消息
String[1]	消息内容; 该值为BlueToothTypes.CONNECT_SUCCESS时表示为连接成功 , 为BlueToothTypes.NOTIFY_SUCCESS时表示注册收发服务成功 , 为BlueToothTypes.NOTIFY_FAIL时表示注册收发服务失败

- 执行连接动作后，如收到连接成功消息和注册收发服务成功消息，代表本次连接通讯OK，可以开始执行操作；如收到连接成功和注册收发服务失败消息，则代表本次连接通讯异常，应用层需提示用户等待约10s后再次进行连接操作。

API Ver : 0.1.0

API Compile Time : 2021-06-24