MRXC2H2SDK

Reference Guide 开发手册

Asterisk,Inc.

2023-04-28

修改日志

版本	修订内容	修订日期
1.0	初稿	2023/4/28

目 录

简介

本文的主要目的：

- 指导开发人员构建开发环境，以便开发人员可以使用 MRXC2H2SDK 库来开发 Android 应用程序。
- 向用户说明 SDK 库。

开发工具：

- Android Studio Arctic Fox | 2020.3.1
- Android SDK 24
- Android Gradle 8.1

系统要求：

- Android 7.0+

1 库文件导入及使用

1.1 导入库文件

找到 app 目录下项目文件“libs”，右键点击选择“Reveal in Finder”（如图 1-1-1）

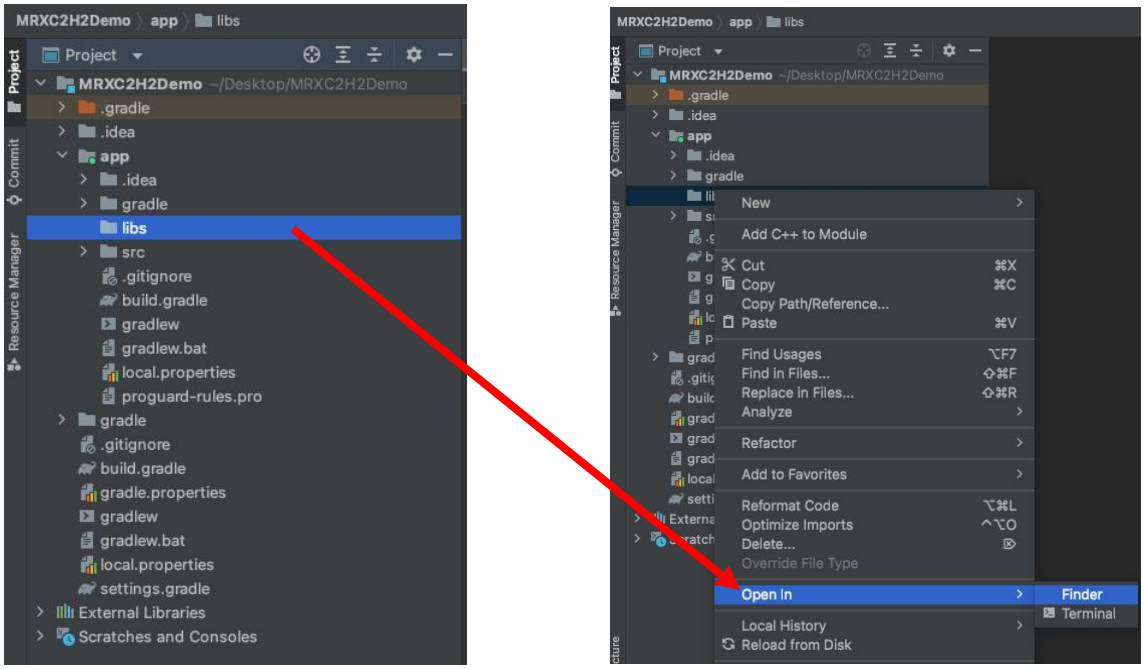

图 1-1-1

在弹出的窗口中选择“libs”目录，然后将“mrxc2h2sdk.aar”粘贴在此文件夹下（如图1-1-2），此时“mrxc2h2sdk.aar”将显示在本工程“libs”下（如图 1-1-3）。

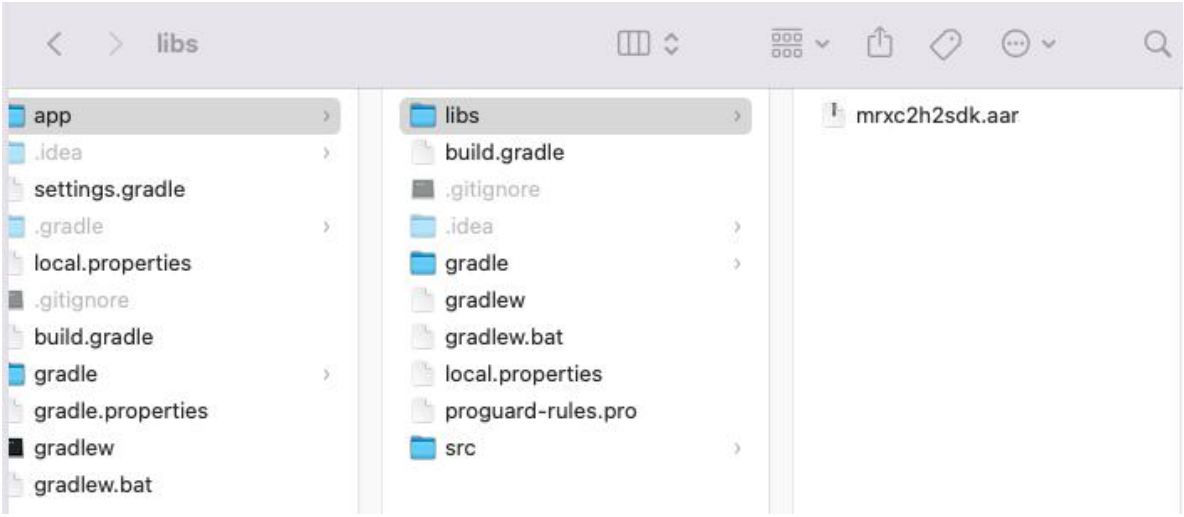

图 1-1-2

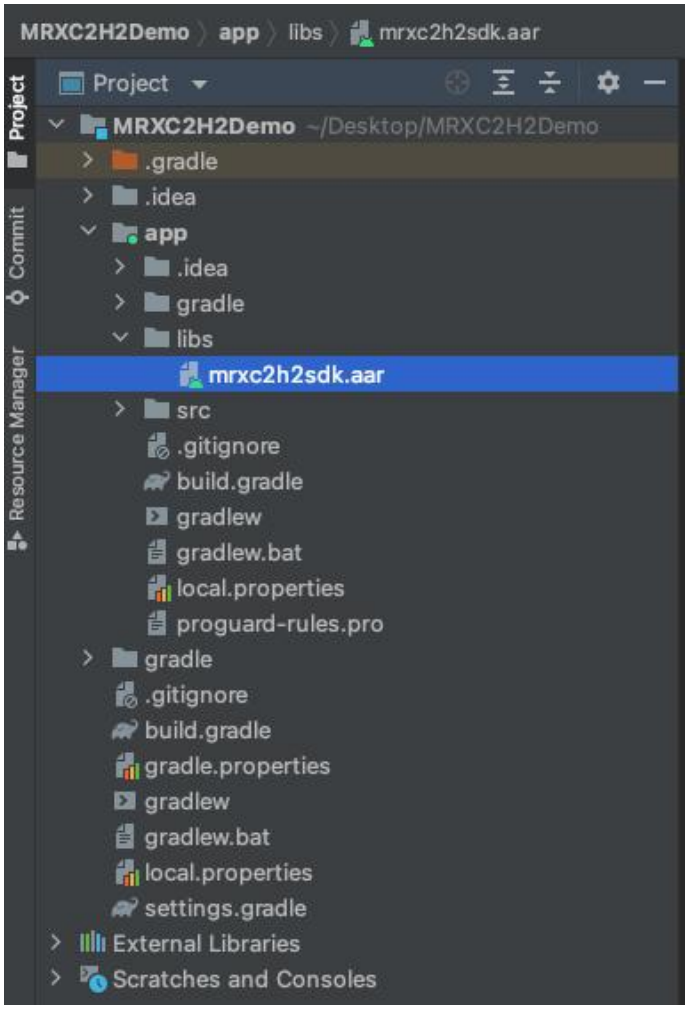
MRXC2H2Demo
app
libs
mrxc2h2sdk.aar
Project
Commit
Resource Manager
MRXC2H2Demo ~/Desktop/MRXC2H2Demo
.gradle
.idea
app
.idea
gradle
libs
mrxc2h2sdk.aar
src
.gitignore
build.gradle
gradlew
gradlew.bat
local.properties
proguard-rules.pro
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
External Libraries
Scratches and Consoles

图 1-1-3

在工程中双击打开“build.gradle”（如图 1-1-4）。按照（如图 1-1-5）（标注 1）添加仓库与依赖，按照（标注 2）点击“Sync Now”进行同步。

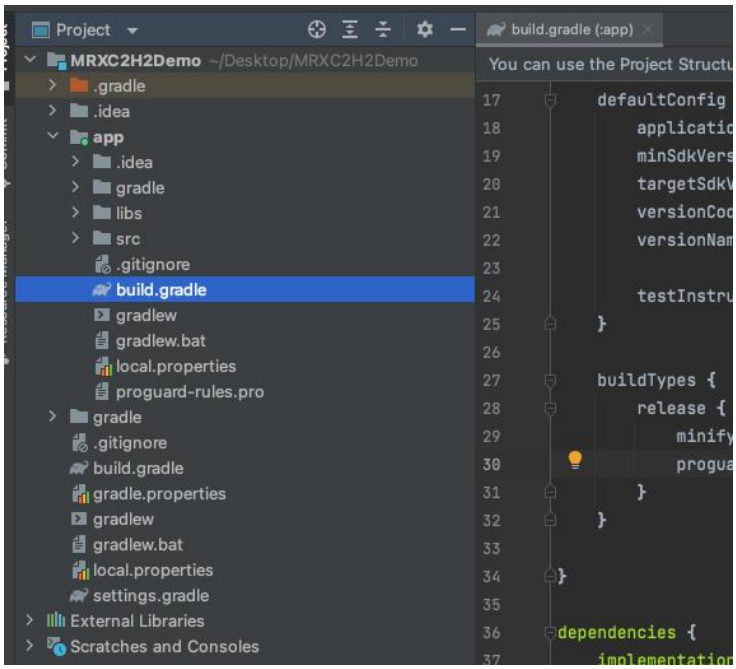

图 1-1-4

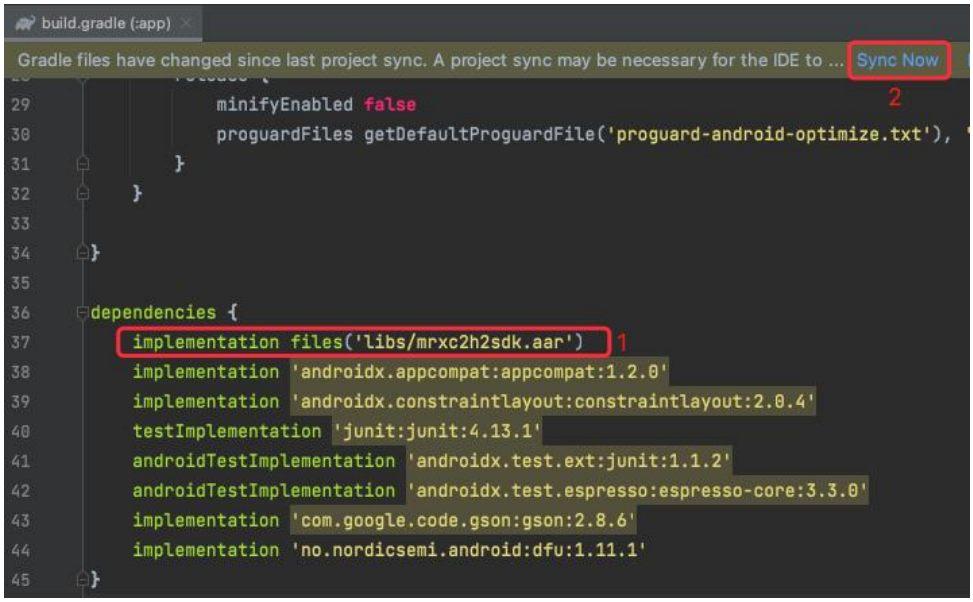

图 1-1-5

同步成功(如图 1-1-6 所示)，至此，SDK 导入成功。

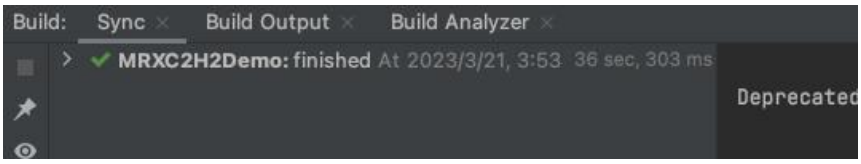

图 1-1-6

1.2 使用库文件

在要使用库文件的类中，使用“import”语句，来引用 MRXC2H2SDK 库文件（如图 1-2-1）

```
import com.asreader.mrxc2h2sdk.MRXC2H2SDK;
```

图 1-2-1

按照（如图 1-2-2）中操作：

1) 创建并初始化“MRXC2H2SDK”对象并传入 Activity 对象（this）。（标记 1，2，3）

图 1-2-2

调用 API：以“connect(BluetoothDevice device, MRXC2H2DeviceType type)”为例，按照（如图 1-2-3）中（标记）完成：

1) 使用“import”语句引用“BluetoothAdapter”、“BluetoothDevice”、MRXC2H2DeviceType 库文件（标记 1）。
2) 获得本地蓝牙适配器“BluetoothAdapter”对象（标记 3）。
3) 获取 BluetoothDevice 对象（标记 4）。
4) 进行连接（标记 5）。

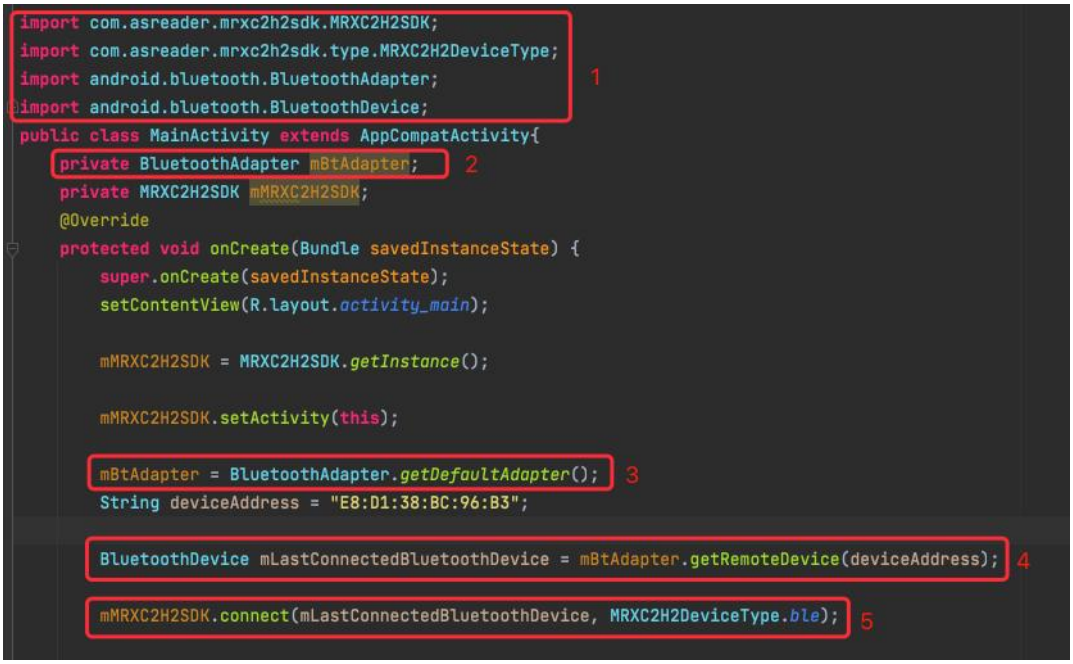

图 1-2-3

创建并初始化 MRXC2H2EventListener 对象并重写 MRXC2H2EventListener 接口中的方法（如图 1-2-4）。

```
private MRXC2H2EventListener mMRXC2H2EventListener = new MRXC2H2EventListener() {
    @Override
    public void onStateChanged(final MRXC2H2ConnectionType connectionType) {}
    @Override
    public void receiveFirmwareVersion(String firmwareVersion) {}
    @Override
    public void receiveHardwareVersion(String hardwareVersion) {}
    @Override
    public void receiveManufacturerName(String manufacturerName) {}
    @Override
    public void receivedBattery(int battery) {}
    @Override
    public void receivedBarcodeData(byte[] barcodeData,MRXC2H2BarcodeType barcodeType) {}
    @Override
    public void startScanStatus(boolean status) {}
    @Override
    public void stopScanStatus(final boolean status) {}
    @Override
    public void setBeepStatus(boolean status) {}
    @Override
    public void setVibrationStatus(boolean status) {}
    @Override
    public void setBarcodeTimeoutStatus(boolean status) {}
    @Override
    public void receivedBeepIsOn(boolean isOn) {}
    @Override
    public void receivedVibrationIsOn(boolean isOn) {}
    @Override
    public void receivedBarcodeTimeoutValue(MRXC2H2BarcodeTimeout barcodeTimeout) {}
    @Override
    public void receivedData(byte[] data) {}
    @Override
    public void receivedDevice(BluetoothDevice device) {}
    @Override
    public void receivedFoundDeviceFinished() {}
};
```

图 1-2-4

连接蓝牙设备前，需调用“setMRXC2H2EventListener(MRXC2H2EventListener listener)”方法并传入MRXC2H2EventListener对象，来进行蓝牙连接状态的监听和接收从MRXC2H2设备返回的数据。（如图 1-2-5）。

```
mMRXC2H2SDK.setMRXC2H2SDKEventListener(mMRXC2H2EventListener);
```

图 1-2-5

以接口中 onStateChanged 和 receivedBarcodeData 方法作为示例（如图 1-2-6）。

```
@Override
public void onStateChanged(MRXC2H2ConnectionType connectionType) {
    switch (connectionType) {
        case Connected:
            break;
        case Connecting:
            break;
        case Disconnected:
            break;
    }
}

@Override
public void receivedBarcodeData(byte[] barcodeData,MRXC2H2BarcodeType barcodeType) {
    // One the function startScan() is called,the device returns the barcode data scanned.
    // Encoding
    // Charset mCharset = Charset.forName("ASCII");
    // String str = "";
    // try {
    //      str = new String(barcodeData,mCharset);
    // }catch (Exception e){}
    // barcodeType: Barcode type
}
```

图 1-2-6

1.3 添加项目权限

在使用 sdk 前, 需在 app 的 AndroidManifest.xml 文件中定义蓝牙相关权限 (如图 1-3-1)。

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

图 1-3-1

2 方法说明

2.1 MRXC2H2SDK

2.1.1 getInstance

函数名	*public static MRXC2H2SDK getInstance()*		
参数名	IN/OUT	类型	说明
	OUT	MRXC2H2SDK	MRXC2H2SDK 实例对象
■方法说明： 获取 MRXC2H2SDK 实例对象 ■示例代码： //创建并初始化 MRXC2H2SDK 对象 MRXC2H2SDK mMRXC2H2SDK = MRXC2H2SDK.getInstance();			

2.1.2 setMRXC2H2SDKEventListener

函数名	public void setMRXC2H2SDKEventListener(MRXC2H2EventListener mrxC2H2EventListener)		
参数名	IN/OUT	类型	说明
mrxC2H2EventListener	IN	MRXC2H2EventListener	MRXC2H2EventListener 对象（参照 2.2）
■方法说明： 设置 MRXC2H2EventListener。 ■示例代码：（注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象, mrxC2H2EventListener 为 MRXC2H2EventListener 的实类对象） mMRXC2H2SDK.setMRXC2H2SDKEventListener (mrxC2H2EventListener);			

2.1.3 connect

函数名	public void connect(BluetoothDevice device, MRXC2H2DeviceType type)		
参数名	IN/OUT	类型	说明
device	IN	BluetoothDevice	BluetoothDevice 对象
type	IN	MRXC2H2DeviceType	连接模式 枚举类型（参照：3.1）
■方法说明： 连接 MRXC2H2 设备。 此方法执行后，通过 onStateChanged（参照：2.2.1）回调方法接收连接状态。 ■示例代码：（注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象，mDevice 为 BluetoothDevice 类的实类对象） mMRXC2H2SDK.connect(mDevice, MRXC2H2DeviceType.*ble*);			

2.1.4 setActivity

函数名	public void setActivity(Activity activity)		
参数名	IN/OUT	类型	说明
activity	IN	Activity	上下文环境

■方法说明：
设置一个与应用生命周期一样的上下文环境。
■示例代码：(注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象,this 为 Activity 对象)
mMRXC2H2SDK.setActivity (this);

2.1.5 disconnect

函数名	public void disconnect()

■方法说明：
断开与 MRXC2H2 设备的连接。
此方法执行后，通过 onStateChanged（参照：2.2.1）回调方法接收连接状态。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.disconnect();

2.1.6 getSdkVersion

函数名	public String getSdkVersion()		
参数名	IN/OUT	类型	说明
-	OUT	String	SDK 版本号

■方法说明：
获取 MRXC2H2SDK 的版本号。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.getSdkVersion();

2.1.7 getFirmwareVersion

函数名	public void getFirmwareVersion()

■方法说明：
获取 MRXC2H2 设备的固件版本号。
此方法执行后，通过 receivedFirmwareVersion（参照 2.2.4）回调方法接收固件版本信息。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.getFirmwareVersion();

2.1.8 getBattery

函数名	public void getBattery()

■方法说明：
获取 MRXC2H2 设备的电量值。
此方法执行后，通过 receivedBattery（参照 2.2.3）回调方法接收电量值数据。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.getBattery();

2.1.9 startScan

函数名	public void startScan()
■方法说明： 开始扫描。 此方法执行后，通过 receivedBarcodeData（参照 2.2.2）回调方法接收扫描条码数据.通过 startScanStatus（参照 2.2.13）回调方法接收开始扫描的状态。 ■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象） mMRXC2H2SDK.startScan();	

2.1.10 stopScan

函数名	public void stopScan()
■方法说明： 停止扫描。 此方法执行后，通过 stopScanStatus（参照 2.2.14）回调方法接收停止扫描的状态。 ■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象） mMRXC2H2SDK.startScan();	

2.1.11 startDiscovery

函数名	public void startDiscovery()
■方法说明： 检索蓝牙设备。 此方法执行后，通过receivedDevice（参照2.2.6）回调方法接收检索到的蓝牙设备。 ■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象） mMRXC2H2SDK. startDiscovery();	

2.1.12 stopDiscovery

函数名	public void stopDiscovery()
■方法说明： 停止检索蓝牙设备。 此方法执行后，通过receivedFoundDeviceFinished（参照2.2.7）回调方法接收检索蓝牙设备停止信息。 ■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象） mMRXC2H2SDK. stopDiscovery();	

2.1.13 getPairedDevices

函数名	public Set<BluetoothDevice> getPairedDevices()		
参数名	IN/OUT	类型	说明
-	OUT	Set<Bluetooth Device>	蓝牙设备列表

■方法说明：
获取手机端已匹配的蓝牙设备列表。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
Set<BluetoothDevice> pairedDevices = mMRXC2H2SDK. getPairedDevices();

2.1.14 deviceType

函数名	public MRXC2H2DeviceType deviceType()		
参数名	IN/OUT	类型	说明
-	OUT	MRXC2H2DeviceType	连接模式 枚举类型（参照：3.1）

■方法说明：
获取当前已连接 MRXC2H2 设备的蓝牙连接类型。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
DeviceType mDeviceType = mMRXC2H2SDK.deviceType();

2.1.15 connectedDevice

函数名	public BluetoothDevice connectedDevice()		
参数名	IN/OUT	类型	说明
-	OUT	BluetoothDevice	蓝牙设备

■方法说明：
获取已连接的蓝牙设备

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
BluetoothDevice mBluetoothDevice = mMRXC2H2SDK.connectedDevice();

2.1.16 getBeepStatus

函数名	public void getBeepStatus()

■方法说明：
获取 MRXC2H2 设备的蜂鸣器状态。
此方法执行后，通过 receivedBeepIsOn（参照2.2.11）回调方法接收MRXC2H2设备的蜂鸣器状态。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.getBeepStatus ();

2.1.17 setBeepOn

函数名	public void setBeepOn(boolean isOn)		
参数名	IN/OUT	类型	说明
isOn	IN	boolean	MRXC2H2 设备蜂鸣状态： True：开启。 False：关闭。

■方法说明：
设置 MRXC2H2 设备的蜂鸣器状态。
此方法执行后，通过setBeepStatus（参照2.2.15）回调方法接收MRXC2H2设备的蜂鸣器状态是否设置成功。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.setBeepStatus (isOn);

2.1.18 getVibrationStatus

函数名	public void getVibrationStatus()

■方法说明：
获取 MRXC2H2 设备的振动器状态。
此方法执行后，通过receivedVibrationIsOn（参照2.2.10）回调方法接收MRXC2H2设备的振动器状态。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.getVibrationStatus();

2.1.19 setVibrationOn

函数名	public void setVibrationOn(boolean isOn)		
参数名	IN/OUT	类型	说明
isOn	IN	boolean	MRXC2H2 设备振动状态： True：开启。 False：关闭。

■方法说明：
设置 MRXC2H2 设备的振动器状态。
此方法执行后，通过setVibrationStatus（参照2.2.16）回调方法接收MRXC2H2设备的蜂鸣器状态是否设置成功。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.getVibrationStatus();

2.1.20 getBarcodeTimeout

函数名	public void getBarcodeTimeout()

■方法说明：
获取 MRXC2H2 设备的扫描超时时间。

此方法执行后，通过receivedBarcodeTimeoutValue（参照2.2.12）回调MRXC2H2设备的扫描超时时间。

■示例代码： （注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象）
mMRXC2H2SDK.getBarcodeTimeout ();

2.1.21 setBarcodeTimeout

函数名	public void setBarcodeTimeout(MRXC2H2BarcodeTimeout barcodeTimeout)		
参数名	IN/OUT	类型	说明

	IN	MRXC2H2BarcodeTimeout	MRXC2H2BarcodeTimeout 对象 扫描超时时间 枚举类型（参照 3.4）
■方法说明： 设置当前 MRXC2H2 设备扫码超时时间。 此方法执行后，通过setBarcodeTimeoutStatus（参照2.2.17）回调方法接收MRXC2H2设备扫码超时时间是否设置成功。 ■示例代码：（注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象） mMRXC2H2SDK.setBarcodeTimeout (MRXC2H2BarcodeTimeout. *BarcodeTimeout_4S*);			

2.1.22 getHardwareVersion

函数名	public void getHardwareVersion()
■方法说明： 获取 MRXC2H2 设备的硬件版本号。 此方法执行后，通过receiveHardwareVersion（参照2.2.8）回调方法接收MRXC2H2设备的硬件版本号。 ■示例代码：（注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象） mMRXC2H2SDK.getHardwareVersion();	

2.1.23 getManufatureName

函数名	public void getManufacturerName()
■方法说明： 获取 MRXC2H2 设备的制造商名称。 此方法执行后，通过receiveManufacturerName （参照2.2.9）回调方法接收MRXC2H2设备的制造商名称。 ■示例代码：（注：mMRXC2H2SDK 为 MRXC2H2SDK 类的实类对象） mMRXC2H2SDK. getManufacturerName();	

2.2 MRXC2H2EventListener

2.2.1 onStateChanged

函数名	void onStateChanged(MRXC2H2ConnectionType connectionType);		
参数名	IN/OUT	类型	说明
connectionType	OUT	MRXC2H2ConnectionType	蓝牙设备连接状态 枚举类型：（参照：3.2）

■方法说明：
监听 MRXC2H2 设备连接状态。
当调用 connect（参照 2.1.3）方法或 disconnect（参照 2.1.5）方法时，通过该方法回调连接状态。

■示例代码：

```
@Override
public void onStateChanged(MRXC2H2ConnectionType connectionType) {
    switch (connectionType){
        //Connected
        case Connected:
            break;
        //Disconnected
        case Disconnected:
            break;
        //Connecting
        case Connecting:
            break;
    }
}
```

2.2.2 receivedBarcodeData

函数名	void receivedBarcodeData(byte[] bytes, MRXC2H2BarcodeType barcodeType);		
参数名	IN/OUT	类型	说明
bytes	OUT	byte[]	扫描条码的数据
barcodeType	OUT	MRXC2H2BarcodeType	条码类型 枚举类型 MRXC2H2BarcodeType 对象（参照 3.3）

■方法说明：
接收 MRXC2H2 设备扫描到的数据。
当调用 startScan（参照 2.1.9）方法并扫描到条码时，通过该方法回调扫描到的数据。

■示例代码：

```
@Override
public void receivedBarcodeData(byte[] barcodeData, MRXC2H2BarcodeType barcodeType) {
    // 扫描条码的数据（barcodeData）
    // 条码类型（barcodeType）
}
```

2.2.3 receivedBattery

函数名	void receivedBattery(int battery);		
参数名	IN/OUT	类型	说明
battery	OUT	Int	电量值 数值范围为 1~4 的整数。

■方法说明：
接收 MRXC2H2 设备返回的电量值。
当调用 getBattery（参照 2.1.8）方法时，通过该方法回调当前电量值。

■示例代码：

```
@Override
public void receivedBattery(int battery) {
    //设备电量值（battery）
}
```

2.2.4 receivedFirmwareVersion

函数名	void receivedFirmwareVersion (String firmwareVersion);		
参数名	IN/OUT	类型	说明
firmwareVersion	OUT	String	固件版本号

■方法说明：
接收 MRXC2H2 设备返回的固件版本号。
当调用 getFirmwareVersion()（参照 2.1.7）方法时，通过该方法回调固件版本号。

■示例代码：

```
@Override
public void receivedFirmwareVersion(String firmwareVersion) {
    //固件版本号（firmwareVersion）
}
```

2.2.5 receivedData

函数名	void receivedData(byte[] data);		
参数名	IN/OUT	类型	说明
data	OUT	byte[]	MRXC2H2 设备返回的数据

■方法说明：
接收 MRXC2H2 设备返回的数据。
当 MRXC2H2 设备返回数据时，通过该方法回调返回的数据。

■示例代码：

```
@Override
public void receivedData(byte[] data) {
    // MRXC2H2设备返回的数据 （data）
}
```

2.2.6 receivedDevice

函数名	void receivedDevice(BluetoothDevice bluetoothDevice);		
参数名	IN/OUT	类型	说明
bluetoothDevice	OUT	BluetoothDevice	蓝牙设备

■方法说明:
接收检索到的蓝牙设备。
当调用startDiscovery（参照2.1.11）方法时，通过该方法回调索到蓝牙设备。

■示例代码:

```
@Override
public void receivedDevice(BluetoothDevice bluetoothDevice) {
    // 蓝牙设备（bluetoothDevice）
}
```

2.2.7 receivedFoundDeviceFinished

函数名	void receivedFoundDeviceFinished();

■方法说明:
接收检索蓝牙设备停止的信息。
当调用stopDiscovery（参照2.1.12）方法停止检索蓝牙设备时该方法会回调。
■示例代码:

```
@Override
public void receivedFoundDeviceFinished(){
    //stopDiscovery方法执行完成
}
```

2.2.8 receiveHardwareVersion

函数名	void receiveHardwareVersion(String hardwareVersion);		
参数名	IN/OUT	类型	说明
hardwareVersion	OUT	String	硬件版本号

■方法说明:
接收固件的硬件版本号。
当调用getHardwareVersion（参照2.1.22）方法时，通过该方法回调固件的硬件版本号。
■示例代码:

```
@Override
public void receiveHardwareVersion (String hardwareVersion){
    //硬件版本号 （hardwareVersion）
}
```

2.2.9 receiveManufacturerName

函数名	void receiveManufacturerName(String manufacturerName);		
参数名	IN/OUT	类型	说明
manufacturerName	OUT	String	制造商名称

■方法说明:
接收固件的制造商名称。
当调用getManufacturerName（参照2.1.23）方法时，通过该方法回调硬件制造商名称。
■示例代码:

```
@Override
public void receiveManufacturerName(String manufacturerName){
    //制造商名称（manufacturerName）
}
```

2.2.10 receivedVibrationIsOn

函数名	void receivedVibrationIsOn(boolean isOn);		
参数名	IN/OUT	类型	说明
isOn	OUT	boolean	振动器开关状态： True：振动器开启。 False： 振动器关闭。

■方法说明：
接收 MRXC2H2 设备的振动器状态。
当调用getVibrationStatus（参照2.1.18）方法时，通过该方法回调MRXC2H2设备振动器的状态。
■示例代码：
@Override
public void receivedVibrationIsOn (boolean isOn){
if(isOn){
//震动器为开启状态
}else{
//震动器为关闭状态
}
}			

2.2.11 receivedBeepIsOn

函数名	void receivedBeepIsOn(boolean isOn);		
参数名	IN/OUT	类型	说明
isOn	OUT	boolean	蜂鸣器开关状态： True：蜂鸣开启。 False：蜂鸣关闭。

■方法说明：
接收 MRXC2H2 设备的蜂鸣器状态。
当调用getBeepStatus（参照2.1.16）方法时，通过该方法回调MRXC2H2设备蜂鸣的状态。
■示例代码：
@Override
public void receivedBeepIsOn(boolean isOn){
if(isOn){
//蜂鸣器为开启状态
}else{
//蜂鸣器为关闭状态
}
}			

2.2.12 receivedBarcodeTimeoutValue

函数名	void receivedBarcodeTimeoutValue(MRXC2H2BarcodeTimeout barcodeTimeout);		
参数名	IN/OUT	类型	说明
barcodeTimeout	OUT	MRXC2H2BarcodeTimeout	扫描超时时间 枚举类型（参照 3.4 ）

■方法说明：
接收当前 MRXC2H2 设备扫码超时时间。
当调用getBarcodeTimeout（参照2.1.20）方法时，通过该方法回调MRXC2H2设备扫码超时时间。
■示例代码：

```
@Override
public void receivedBarcodeTimeoutValue (BarcodeTimeout barcodeTimeout){
    // 返回扫码超时时间（barcodeTimeout）
}
```

2.2.13 startScanStatus

函数名	void startScanStatus(boolean status);		
参数名	IN/OUT	类型	说明
status	OUT	boolean	开始扫码的执行结果： True：开始扫描成功。 False：开始扫描失败。

■方法说明：
开始扫描时通过该方法回调执行结果
当调用startScan()（参照2.1.9）方法或按下trigger键后触发开始扫描事件。
■示例代码：

```
@Override
public void startScanStatus (boolean status){
    if(status){
        //开始扫描成功
    }else{
        //开始扫描失败
    }
}
```

2.2.14 stopScanStatus

函数名	void stopScanStatus(boolean status);		
参数名	IN/OUT	类型	说明
status	OUT	boolean	停止扫描的执行结果： True：方法执行成功。 False：停止扫描失败。

■方法说明：
停止扫描时通过该方法回调执行结果。
当调用stopScan（参照2.1.10）方法、抬起trigger键或扫码条码超时后触发停止扫描事件。
■示例代码：

```
@Override
public void stopScanStatus(boolean status){
    if(status){
        //停止扫描成功
    }else{
        //停止扫描失败
    }
}
```

2.2.15 setBeepStatus

函数名	void setBeepStatus(boolean status);		
参数名	IN/OUT	类型	说明
status	OUT	boolean	调用 setBeepOn（参照 2.1.17）方法的执行结果： True：方法执行成功。 False： 方法执行失败。

■方法说明：
当调用setBeepOn（参照2.1.17）方法后，通过该方法回调执行结果。
■示例代码：

```
@Override
public void setBeepStatus (boolean status){
    //if(status){
        //setBeepOn方法执行成功
    }else{
        //setBeepOn方法执行失败
    }
}
```

2.2.16 setVibrationStatus

函数名	void setVibrationStatus(boolean status);		
参数名	IN/OUT	类型	说明
status	OUT	boolean	调用 setVibrationOn（参照 2.1.19）方法的执行结果： True：方法执行成功。 False： 方法执行失败。

■方法说明：
当调用setVibrationOn（参照2.1.19）方法后，通过该方法回调执行结果。
■示例代码：

```
@Override
public void setVibrationStatus (boolean status){
    if(status){
        //setVibrationOn方法执行成功
    }else{
        //setVibrationOn方法执行失败
    }
}
```

2.2.17 setBarcodeTimeoutStatus

函数名	void setBarcodeTimeoutStatus(boolean status);		
参数名	IN/OUT	类型	说明
status	OUT	boolean	调用 setBarcodeTimeout（参照 2.1.21）方法的执行结果： True：方法执行成功。 False： 方法执行失败。

■方法说明：
当调用setBarcodeTimeout（参照2.1.21）方法后，通过该方法回调执行结果。
■示例代码：

```
@Override
public void setBarcodeTimeoutStatus (boolean status){
    if(status){
        //setBarcodeTimeout方法执行成功
    }else{
        //setBarcodeTimeout方法执行失败
    }
}
```

3 Enum

3.1 MRXC2H2DeviceType

定义	说明
unknow = 0	no connection
ble = 1	ble connection
spp = 2	spp connetion

3.2 MRXC2H2ConnectionType

定义	说明
Connected = 0	connected
Connecting = 1	connecting
Disconnected = 2	disconnected

3.3 MRXC2H2BarcodeType

定义	说明
Code39 = 0x01	Code 39
Code11 = 0x0C	Code 11
Codabar = 0x02	Codabar
EAN13 = 0x0B	EAN-13
Code128 = 0x03	Code 128
EAN13With2Supps = 0x4B	EAN 13 with 2 Supps.
Discrete2Of5 = 0x04	Discrete 2 of 5
EAN13With5Supps = 0x8B	EAN 13 with 5 Supps.
IATA2Of5 = 0x05	IATA 2 of 5
MSI = 0x0E	MSI
Interleaved2Of5 = 0x06	Interleaved 2 of 5
EAN128 = 0x0F	EAN 128
Code93 = 0x07	Code 93
UPCE1 = 0x10	UPC E1
UPCA = 0x08	UPC A
UPCE1With2Supps = 0x50	UPC E1 with 2 Supps.
UPCAWith2Supps = 0x48	UPC A with 2 Supps.
UPCE1With5Supps = 0x90	UPC E1 with 5 Supps.
UPCAWith5Supps = 0x88	UPC A with 5 Supps.
TriopticCode39 = 0x15	Trioptic Code 39
UPCE0 = 0x09	UPC E0
BooklandEAN = 0x16	Bookland EAN
UPCE0With2Supps = 0x49	UPC E0 with 2 Supps.
CouponCode = 0x17	Coupon Code
UPCE0With5Supps = 0x89	UPC E0 with 5 supps.
GS1DataBarLimitedRSSLimited = 0x31	GS1 DataBar Limited (RSS-Limited)
EAN8 = 0x0A	EAN 8
GS1DataBarRSS14 = 0x30	GS1 DataBar (RSS-14)

EAN8With2Supps = 0x4A	EAN 8 with 2 Supps.
GS1DataBarExpandedRSSExpanded = 0x32	GS1 DataBar Expanded (RSS-Expanded)
EAN8With5Supps = 0x8A	EAN 8 with 5 Supps.
Matrix2Of5 = 0x0D	Matrix 2 of 5
ChinaPostChinese2Of5 = 0x72	China Post (Chinese 2 of 5)
Code32 = 0x20	Code 32
UKPlessey = 0x13	UK Plessey
ISBT128 = 0x19	ISBT 128
PDF417 = 0x11	PDF417
Aztec = 0x2D	Aztec
MicroPDF417 = 0x1A	MicroPDF417
QR = 0x1C	QR
DataMatrix = 0x1B	DataMatrix
MicroQR = 0x2C	Micro QR
HanXinCode = 0xFF	Han Xin Code
Maxicode = 0x25	Maxicode
ITF14 = 0xc0	ITF-14
ITF6 = 0xc1	ITF-6
AIM128 = 0xc2	AIM 128
ISSN = 0xc3	ISSN
ISBN = 0xc4	ISBN
GS1Databar = 0xc5	GS1-Databar

3.4 MRXC2H2BarcodeTimeout

定义	说明
BarcodeTimeout_4S = 0	4s
BarcodeTimeout_8S = 1	8s
BarcodeTimeout_16S = 2	16s
BarcodeTimeout_24S = 3	24s
BarcodeTimeout_30S = 4	30s
BarcodeTimeout_1Min = 5	1min
BarcodeTimeout_1Min30S = 6	1.5min
BarcodeTimeout_2Min = 7	2min
BarcodeTimeout_5Min = 8	5min