MRXC2H2SDK

Reference Guide

Asterisk,Inc.

2023-04-28

Revision History

Ver.	Contents	Date
1.0	Initial version	2023/4/28

Contents

Introduction

Main purposes of this paper:

- Guide developers to build the development environment so that developers can use the MRXC2H2SDK library to develop Android applications.
- Explain the SDK library to the users.

Development tools:

- Android Studio Arctic Fox | 2020.3.1
- Android SDK 24
- Android Gradle 5.4 - 2.1

System requirements:

- Android 7.0+

1 SDK Import and Usage

1.1 Import SDK

Click the project file "libs" in the app folder, and right-click "Reveal in Finder" (as shown in FIG. 1-1-1).

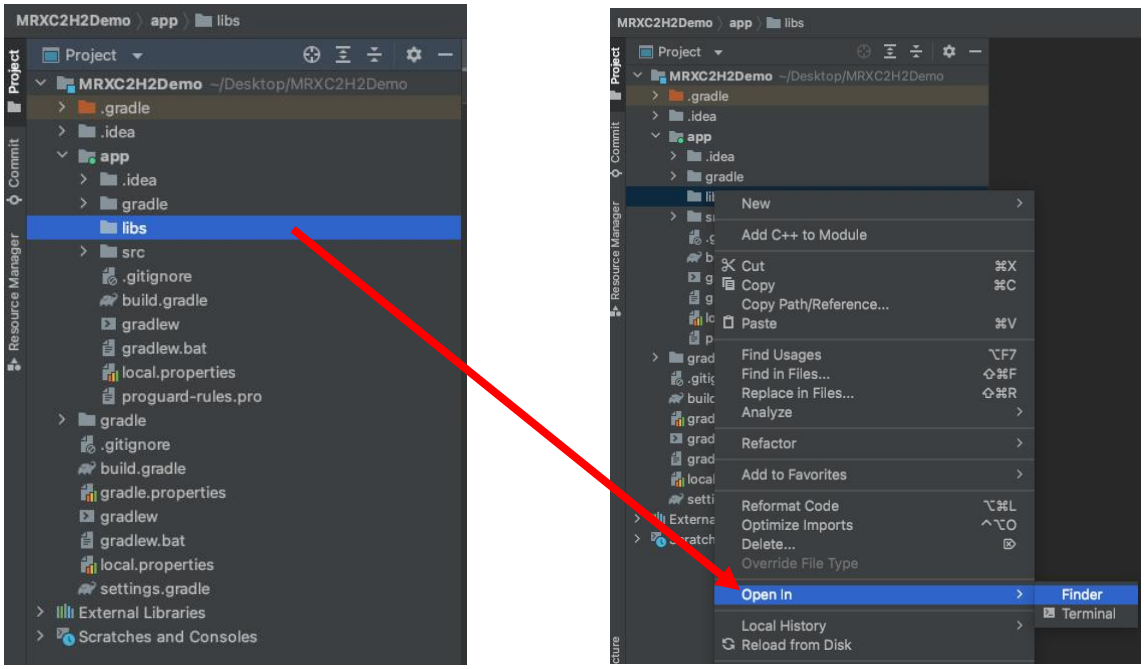

FIG. 1-1-1

In the pop-up window, select the "libs" directory, and paste " mrxc2h2sdk.aar " into it (as shown in FIG. 1-1-2). After the above operation, " mrxc2h2sdk.aar " will appear under "libs" of this project (as shown in FIG. 1-1-3).

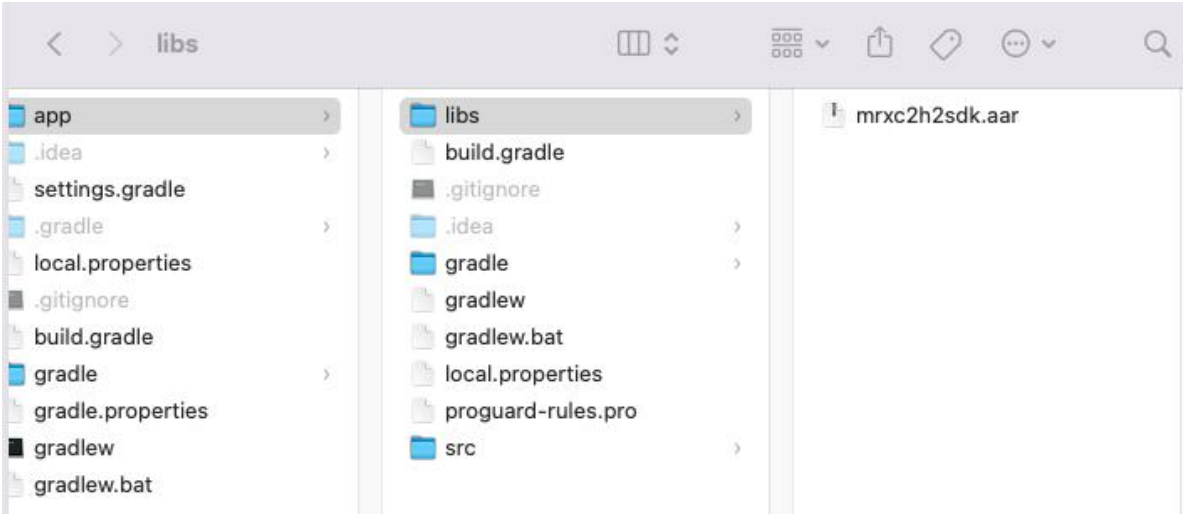

FIG. 1-1-2

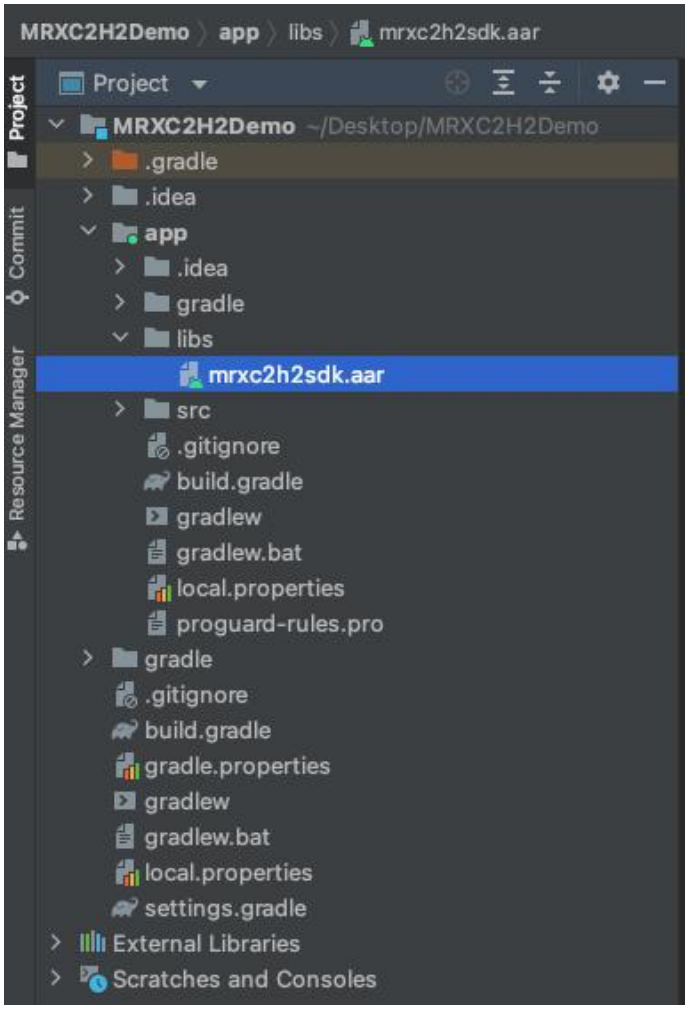
MRXC2H2Demo
app
libs
mrxc2h2sdk.aar
Project
Commit
Resource Manager
MRXC2H2Demo ~/Desktop/MRXC2H2Demo
.gradle
.idea
app
.idea
gradle
libs
mrxc2h2sdk.aar
src
.gitignore
build.gradle
gradlew
gradlew.bat
local.properties
proguard-rules.pro
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat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

FIG. 1-1-3

Double-click to open "build.gradle" in the project (as shown in FIG. 1-1-4). Add the repositories and dependencies as shown in step 1 of FIG. 1-1-5, and click 'Sync Now'.

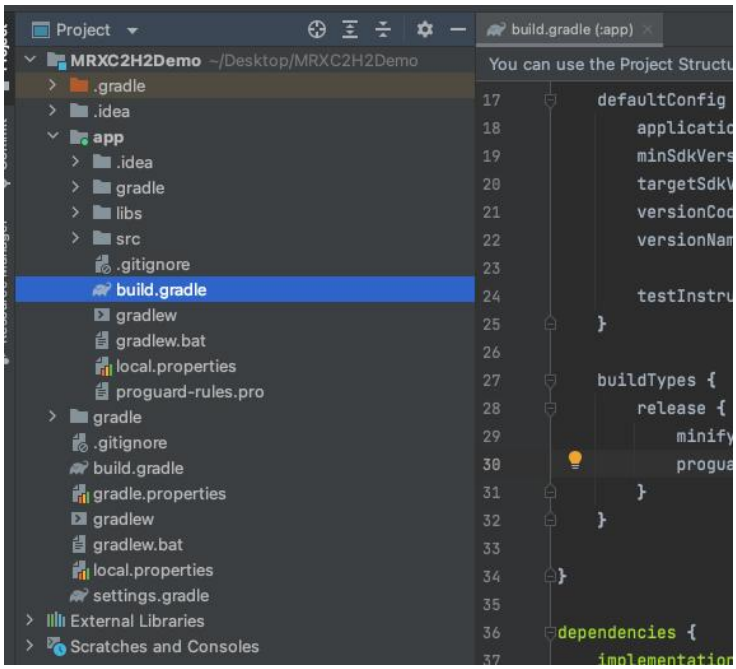

FIG. 1-1-4

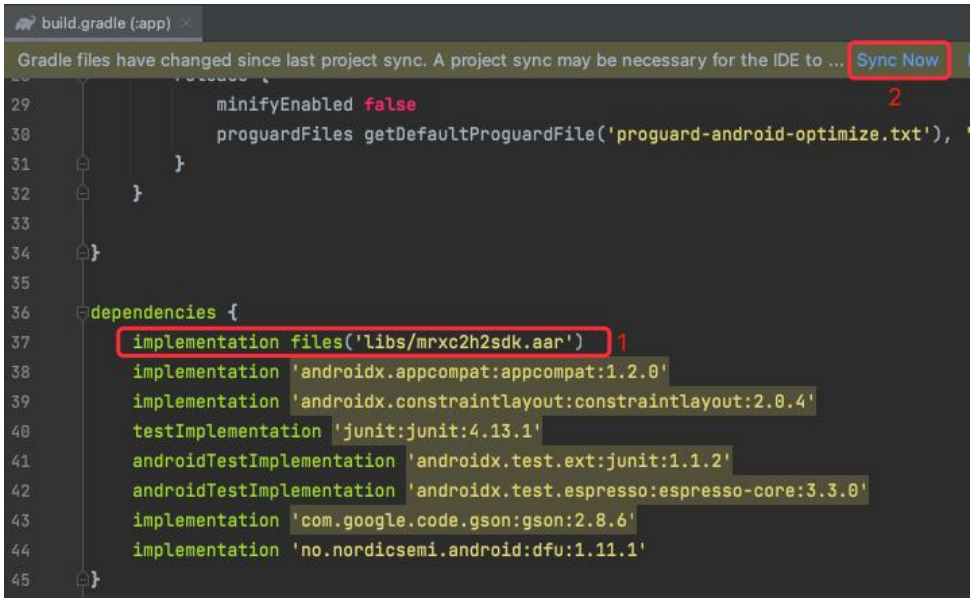

FIG. 1-1-5

Successful synchronization is shown below. Thus, SDK import is successful.

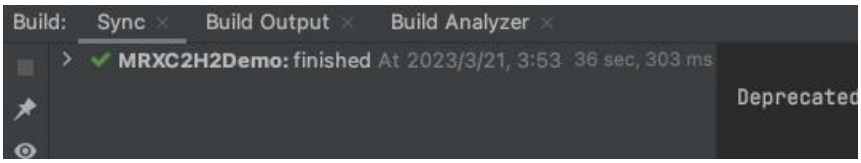

FIG. 1-1-6

1.2 SDK Usage

In the class to use the SDK, use the "import" statement to reference the MRXC2H2SDK (as shown in FIG. 1-2-1).

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

FIG. 1-2-1

Follow the steps below:
Create an object for MRXC2H2SDK and pass in the Activity object (this) for it.

```
private MRXC2H2SDK mMRXC2H2SDK;  1
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mMRXC2H2SDK = MRXC2H2SDK.getInstance();  2

    mMRXC2H2SDK.setActivity(this);  3
```

FIG. 1-2-2

Call API: Take connect(BluetoothDevice device, MRXC2H2DeviceType type) as an example, follow the steps in FIG. 1-2-3:

1) Reference BluetoothAdapter, BluetoothDevice, and MRXC2H2DeviceType library files by the "import" statement (Mark 1).
2) Get the local Bluetooth adapter, BluetoothAdapter objects. (Mark3)
3) Get the BluetoothDevice object. (Mark4)
4) Connect. (Mark 5)

```
import com.asreader.mrxc2h2sdk.MRXC2H2SDK;
import com.asreader.mrxc2h2sdk.type.MRXC2H2DeviceType;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;  1
public class MainActivity extends AppCompatActivity{
    private BluetoothAdapter mBtAdapter;  2
    private MRXC2H2SDK mMRXC2H2SDK;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mMRXC2H2SDK = MRXC2H2SDK.getInstance();

        mMRXC2H2SDK.setActivity(this);

        mBtAdapter = BluetoothAdapter.getDefaultAdapter();  3
        String deviceAddress = "E8:D1:38:BC:96:B3";

        BluetoothDevice mLastConnectedBluetoothDevice = mBtAdapter.getRemoteDevice(deviceAddress);  4

        mMRXC2H2SDK.connect(mLastConnectedBluetoothDevice, MRXC2H2DeviceType.ble);  5
```

FIG. 1-2-3

Create the MRXC2H2EventListener object and override the function in the MRXC2H2EventListener interface. FIG. 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() {}
};
```

FIG. 1-2-4

Before connecting to the Bluetooth device, the function setMRXC2H2EventListener(MRXC2H2EventListener listener) must be called and passed in the MRXC2H2EventListener object to listen for the status of the Bluetooth connection and to receive data returned from the MRXC2H2 device. (see FIG. 1-2-5)

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

FIG. 1-2-5

Take the functions onStateChanged and receivedBarcodeData for example: (see FIG. 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
}
```

FIG. 1-2-6

1.3 Add project permissions

You need to define project permissions in the APP's AndroidManifest.xml file before using the SDK (see FIG. 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" />
```

FIG. 1-3-1

2 Function Instructions

2.1 MRXC2H2SDK

2.1.1 getInstance

Function Name	public static MRXC2H2SDK getInstance()		
Parameter Name	IN/OUT	Type	Descriptions
	OUT	MRXC2H2SDK	MRXC2H2SDK object
■Function Description: Gets the object of AsFingerSDK. AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); ■Sample Code: //Create and initialize an object of AsFingerSDK. AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance();			

2.1.2 setMRXC2H2SDKEventListener

Function Name	public void setMRXC2H2SDKEventListener(MRXC2H2EventListener mrxC2H2EventListener)		
Parameter Name	IN/OUT	Type	Descriptions
mrxC2H2Event Listener	IN	MRXC2H2EventListener	MRXC2H2EventListener object (see 2.2)
■Function Description: Sets AsFingerEventListener. **■Sample Code: (**Note: mMRXC2H2SDK is an object of MRXC2H2SDK; mrxC2H2EventListener is an object of MRXC2H2EventListener) mMRXC2H2SDK.setMRXC2H2SDKEventListener (mrxC2H2EventListener);			

2.1.3 connect

Function Name	public void connect(BluetoothDevice device, MRXC2H2DeviceType type)		
Parameter Name	IN/OUT	Type	Descriptions
device	IN	BluetoothDevice	BluetoothDevice object
type	IN	MRXC2H2DeviceType	Connection mode Enumeration type (see: 3.1).
■Function Descriptions: This function is used to connect the MRXC2 or MRX-H2 device (via Bluetooth). Once this function is called, the connection status of the MRX-C2 or MRX-H2 device will be returned via invoking function onStateChanged (see 2.2.1). **■Sample Code: (**Note: mMRXC2H2SDK is an object of MRXC2H2SDK. mDevice is an object of BluetoothDevice.) mMRXC2H2SDK.connect(mDevice, MRXC2H2DeviceType.ble);			

2.1.4 setActivity

Function Name	public void setActivity(Activity activity)		
Parameter Name	IN/OUT	Type	Descriptions
activity	IN	Activity	context

■**Function Description:**
Set up a same context as the application lifecycle.
■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK, this is an object of Activity.)
mMRXC2H2SDK.setActivity (this);

2.1.5 disconnect

Function Name	public void disconnect()

■**Function Descriptions:**
This function is used to disconnect from the MRX-C2 or MRX-H2 device.

Once this function is called, the connection status of the MRX-C2 or MRX-H2 device will be returned via invoking function onStateChanged (see 2.2.1).

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.disconnect();

2.1.6 getSdkVersion

Function Name	public String getSdkVersion()		
Parameter Name	IN/OUT	Type	Descriptions
-	OUT	String	SDK version

■**Function Descriptions:**
This function is used to get the MRXC2H2SDK version.

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.getSdkVersion();

2.1.7 getFirmwareVersion

Function Name	public void getFirmwareVersion()

■**Function Descriptions:**
This function is used to get the firmware version of the MRX-C2 or MRX-H2 device.

Once this function is called, the firmware version will be returned via invoking function receivedFirmwareVersion (see 2.2.4).

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.getFirmwareVersion();

2.1.8 getBattery

Function Name	public void getBattery()

■**Function Descriptions:**
Get the battery remaining of the MRX-C2 or MRX-H2 device.

Once this function is called, the battery remaining of the MRX-C2 or MRX-H2 device will be returned via invoking function receivedBattery (see 2.2.3).

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.getBattery();

2.1.9 startScan

Function Name	public void startScan()

■**Function Descriptions:**
Start to scan barcodes.

Once this function is called, the barcode data scanned by MRX-C2 or MRX-H2 device will be returned via invoking function receivedBarcodeData (see 2.2.2), and the start-scan status will be returned via invoking the function startScanStatus (see 2.2.13).

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.startScan();

2.1.10 stopScan

Function Name	public void stopScan()

■**Function Descriptions:**
To stop scanning.

Once this function is called, the stop-scan status will be returned via invoking the function stopScanStatus (see 2.2.14).

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.startScan();

2.1.11 startDiscovery

Function Name	public void startDiscovery()

■**Function Descriptions:**
Start to search for the Bluetooth devices.

Once this function is called, the Bluetooth devices that be found will be returned via invoking function receivedDevice. (see 2.2.6).

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK. startDiscovery();

2.1.12 stopDiscovery

Function Name	public void stopDiscovery()

■**Function Descriptions:**
Stop searching for Bluetooth devices.

Once this function is called, the function receivedFoundDeviceFinished will be invoked to notify that the search is stopped. (see 2.2.7).

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK. stopDiscovery();

2.1.13 getPairedDevices

Function Name	public Set<BluetoothDevice> getPairedDevices()		
Parameter Name	IN/OUT	Type	Descriptions
-	OUT	Set<Bluetooth Device>	The list of the paired Bluetooth devices

■**Function Descriptions:**
Get the list of the paired Bluetooth devices stored in the phone.

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
Set<BluetoothDevice> pairedDevices = mMRXC2H2SDK. getPairedDevices();

2.1.14 deviceType

Function Name	public MRXC2H2DeviceType deviceType()		
Parameter Name	IN/OUT	Type	Descriptions
-	OUT	MRXC2H2DeviceType	Connection mode Enumeration type (see: 3.1).

■**Function Descriptions:**
Get the Bluetooth connection mode of the current connected MRX-C2 or MRX-H2 device.

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
DeviceType mDeviceType = mMRXC2H2SDK.deviceType();

2.1.15 connectedDevice

Function Name	public BluetoothDevice connectedDevice()		
Parameter Name	IN/OUT	Type	Descriptions
-	OUT	BluetoothDevice	Bluetooth device

■**Function Descriptions:**
Get the object of the Bluetooth device that is currently connected.

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
BluetoothDevice mBluetoothDevice = mMRXC2H2SDK.connectedDevice();

2.1.16 getBeepStatus

Function Name	public void getBeepStatus()

■**Function Descriptions:**
Gets the buzzer status of the MRX-C2 or MRX-H2 device.

Once this function is called, the function receivedBeepIsOn (see 2.2.11) will be invoked to receive the buzzer status of the MRX-C2 or MRX-H2 device.

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.getBeepStatus ();

2.1.17 setBeepOn

Function Name	public void setBeepOn(boolean isOn)		
Parameter Name	IN/OUT	Type	Descriptions
isOn	IN	boolean	The buzzer status of the MRX-C2 or MRX-H2 device True: on. False: off.

■**Function Descriptions:**
Sets the buzzer status of the MRX-C2 or MRX-H2 device.

Once this function is called, the function setBeepStatus (see 2.2.15) will be invoked to receive whether the buzzer status of the MRX-C2 or MRX-H2 device is set successfully.

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.setBeepStatus (isOn);

2.1.18 getVibrationStatus

Function Name	public void getVibrationStatus()

■**Function Descriptions:**
Gets the vibration status of the MRX-C2 or MRX-H2 device.

Once this function is called, the function receivedVibrationIsOn (see 2.2.10) will be invoked to receive the vibration status of the MRX-C2 or MRX-H2 device.

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)
mMRXC2H2SDK.getVibrationStatus();

2.1.19 setVibrationOn

Function Name	public void setVibrationOn(boolean isOn)		
Parameter Name	IN/OUT	Type	Descriptions
isOn	IN	boolean	The vibration status of the MRX-C2 or MRX-H2 device True: on. False: off.

■**Function Descriptions:**
Sets the vibration status of the MRX-C2 or MRX-H2 device.

Once this function is called, the function setVibrationStatus (see 2.2.16) will be invoked to receive whether the vibration status of the MRX-C2 or MRX-H2 device is set successfully.

■**Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.)

mMRXC2H2SDK.getVibrationStatus();

2.1.20 getBarcodeTimeout

Function Name	public void getBarcodeTimeout()
■Function Descriptions: Gets the timeout value of the MRX-C2 or MRX-H2 device. Once this function is called, the function receivedBarcodeTimeoutValue (see 2.2.12)will be invoked to receive the timeout value of the MRX-C2 or MRX-H2 device. **■Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.) mMRXC2H2SDK.getBarcodeTimeout ();	

2.1.21 setBarcodeTimeout

Function Name	public void setBarcodeTimeout(MRXC2H2BarcodeTimeout barcodeTimeout)		
Parameter Name	IN/OUT	Type	Descriptions
	IN	MRXC2H2BarcodeTimeout	MRXC2H2BarcodeTimeout object Barcode scanning timeout Enum (see 3.4).
■Function Descriptions: Sets the barcode scanning timeout of the MRX-C2 or MRX-H2 device. Once this function is called, the function setBarcodeTimeoutStatus (see 2.2.17) will be invoked to receive whether the barcode scanning timeout of the MRX-C2 or MRX-H2 device is set successfully. **■Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.) mMRXC2H2SDK.setBarcodeTimeout (MRXC2H2BarcodeTimeout. *BarcodeTimeout_4S*);			

2.1.22 getHardwareVersion

Function Name	public void getHardwareVersion()
■Function Descriptions: Gets the hardware version of the MRX-C2 or MRX-H2 device. Once this function is called, the function receiveHardwareVersion (see 2.2.8) will be invoked to receive the hardware version of the MRX-C2 or MRX-H2 device. **■Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.) mMRXC2H2SDK.getHardwareVersion();	

2.1.23 getManufatureName

Function Name	public void getManufacturerName()
■Function Descriptions: Gets the manufacture name of the MRX-C2 or MRX-H2 device. Once this function is called, the function receiveManufacturerName (see 2.2.9) will be invoked to receive the manufacture name of the MRX-C2 or MRX-H2 device. **■Sample Code:** (Note: mMRXC2H2SDK is an object of MRXC2H2SDK.) mMRXC2H2SDK. getManufacturerName();	

2.2 MRXC2H2EventListener

2.2.1 onStateChanged

Function Name	void onStateChanged(MRXC2H2ConnectionType connectionType);		
Parameter Name	IN/OUT	Type	Descriptions
connectionType	OUT	MRXC2H2ConnectionType	Connection status with the Bluetooth device Enumeration type (see: 3.2)

■Function Descriptions:
Listen the connection status with the MRX-C2 or MRX-H2 device.
This function will be called back to returned the connection status once the function connect (see 2.1.3)or disconnect (see 2.1.5) is called.

■Sample Code:

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

2.2.2 receivedBarcodeData

Function Name	void receivedBarcodeData(byte[] bytes, MRXC2H2BarcodeType barcodeType);		
Parameter Name	IN/OUT	Type	Descriptions
bytes	OUT	byte[]	The scanned barcode data
barcodeType	OUT	MRXC2H2BarcodeType	Barcode type Enumeration type MRXC2H2BarcodeType object (see 3.3).

■Function Descriptions:
Receives the data scanned by the MRX-C2 or MRX-H2 device.
This function will be called back to returned the scanned data once the function startScan (see 2.1.9) is called and there is any barcode has been scanned.

■Sample Code:

```
    @Override
    public void receivedBarcodeData(byte[] barcodeData, MRXC2H2BarcodeType barcodeType) {
        // Barcode data
        // Barcode type
    }
```

2.2.3 receivedBattery

Function Name	void receivedBattery(int battery);		
Parameter Name	IN/OUT	Type	Descriptions
battery	OUT	Int	Battery power value The value is an integer ranging from 1 to 4.

■Function Descriptions:
Receives the battery power value returned by the MRX-C2 or MRX-H2 device.
This function will be called back to returned the battery power value once the function getBattery (see 2.1.8) is called..

■Sample Code:

```
@Override
public void receivedBattery(int battery) {
    //The remaining battery power value
}
```

2.2.4 receivedFirmwareVersion

Function Name	void receivedFirmwareVersion (String firmwareVersion);		
Parameter Name	IN/OUT	Type	Descriptions
firmwareVersion	OUT	String	The firmware version

■Function Descriptions:
Receives the firmware version returned by the MRX-C2 or MRX-H2 device.
This function will be called back to returned the firmware version once the function getFirmwareVersion() (see 2.1.7) is called..

■Sample Code:

```
@Override
public void receivedFirmwareVersion(String firmwareVersion) {
    //Firmware version
}
```

2.2.5 receivedData

Function Name	void receivedData(byte[] data);		
Parameter Name	IN/OUT	Type	Descriptions
data	OUT	byte[]	Data returned by the MRX-C2 or MRX-H2 device

■Function Descriptions:
Receives data returned by the MRX-C2 or MRX-H2 device.
This function will be called back to received data returned by the MRX-C2 or MRX-H2 device.

■Sample Code:

```
@Override
public void receivedData(byte[] data) {
    // Data returned by the MRX-C2 or MRX-H2 device (data)
}
```

2.2.6 receivedDevice

Function Name	void receivedDevice(BluetoothDevice bluetoothDevice);

Parameter Name	IN/OUT	Type	Descriptions
bluetoothDevice	OUT	BluetoothDevice	Bluetooth devices

■Function Descriptions:
Receives the searched Bluetooth devices.
This function will be called back to returned the searched Bluetooth devices once the function startDiscovery (see 2.1.11) is called.

■Sample Code:

```
@Override
public void receivedDevice(BluetoothDevice bluetoothDevice) {
    // Bluetooth devices
}
```

2.2.7 receivedFoundDeviceFinished

Function Name	void receivedFoundDeviceFinished();

■Function Descriptions:
Receives the notice that the searching to the Bluetooth device has stopped.
This function will be called back to returned the notice that the searching to the Bluetooth device has stopped once the function stopDiscovery (see 2.1.12) is called.

■Sample Code:

```
@Override
public void receivedFoundDeviceFinished(){
    //stopDiscovery function finished.
}
```

2.2.8 receiveHardwareVersion

Function Name	void receiveHardwareVersion(String hardwareVersion);		
Parameter Name	IN/OUT	Type	Descriptions
hardwareVersion	OUT	String	Hardware version

■Function Descriptions:
Receives the hardware version.
This function will be called back to returned the hardware version once the function getHardwareVersion (see 2.1.22) is called.

■Sample Code:

```
@Override
public void receiveHardwareVersion (String hardwareVersion){
    //The hardwareVersion
}
```

2.2.9 receiveManufacturerName

Function Name	void receiveManufacturerName(String manufacturerName);		
Parameter Name	IN/OUT	Type	Descriptions
manufacturerName	OUT	String	Manufacturer name

■**Function Descriptions:**
Receives the manufacture name.
This function will be called back to returned the manufacture name once the function getManufacturerName (see 2.1.23) is called.
■**Sample Code:**

```
@Override
public void receiveManufacturerName(String manufacturerName){
    //Manufacturer name
}
```

2.2.10 receivedVibrationIsOn

Function Name	void receivedVibrationIsOn(boolean isOn);		
Parameter Name	IN/OUT	Type	Descriptions
isOn	OUT	boolean	Vibration status True: vibration on. False: vibration off.

■**Function Descriptions:**
Receives the vibration status of the MRX-C2 or MRX-H2 device.
This function will be called back to returned the vibration status once the function getVibrationStatus (see 2.1.18) is called.

■**Sample Code:**

```
@Override
public void receivedVibrationIsOn (boolean isOn){
    if(isOn){
        // The vibrator is on.
    }else{
        // The vibrator is off.
    }
}
```

2.2.11 receivedBeepIsOn

Function Name	void receivedBeepIsOn(boolean isOn);		
Parameter Name	IN/OUT	Type	Descriptions
isOn	OUT	boolean	buzzer status True: buzzer on. False: buzzer off.

■**Function Descriptions:**
Receives the buzzer status of the MRX-C2 or MRX-H2 device.
This function will be called back to returned the buzzer status once the function getBeepStatus (see 2.1.16) is called.
■**Sample Code:**

```
@Override
public void receivedBeepIsOn(boolean isOn){
    if(isOn){
        // The buzzer is on.
    }else{
        // The buzzer is off.
    }
}
```

2.2.12 receivedBarcodeTimeoutValue

Function Name	void receivedBarcodeTimeoutValue(MRXC2H2BarcodeTimeout barcodeTimeout);		
Parameter Name	IN/OUT	Type	Descriptions
barcodeTimeout	OUT	MRXC2H2BarcodeTimeout	Barcode timeout Enumeration type (see 3.4).

■Function Descriptions:
Receives the timeout value of the current MRX-C2 or MRX-H2 device.
This function will be called back to returned the timeout value once the function getBarcodeTimeout (see 2.1.20) is called.
■Sample Code:

```
@Override
public void receivedBarcodeTimeoutValue (BarcodeTimeout barcodeTimeout){
    // Barcode timeout
}
```

2.2.13 startScanStatus

Function Name	void startScanStatus(boolean status);		
Parameter Name	IN/OUT	Type	Descriptions
status	OUT	boolean	The execution results of the starting scanning. True: suceed. False: failed.

■Function Descriptions:
This function is used to call back the execution results when the scan begins.
This function will be called back to returned the execution results once the function startScan() (see 2.1.9) is called or the trigger key is pressed to start to scan.
■Sample Code:

```
@Override
public void startScanStatus (boolean status){
    if(status){
        // Suceed to start the scan.
    }else{
        // Failed to start the scan.
    }
}
```

2.2.14 stopScanStatus

Function Name	void stopScanStatus(boolean status);		
Parameter Name	IN/OUT	Type	Descriptions
status	OUT	boolean	The execution results ofthe stopping scanning. True: suceed. False: failed.

■**Function Descriptions:**
This function is used to call back the execution results when the scan stops.
This function will be called back to returned the execution results once the function stopScan (see 2.1.10) is called or the trigger key is released to stop scanning.

■**Sample Code:**

```
@Override
public void stopScanStatus(boolean status){
    if(status){
        // Suceed to stop scanning.
    }else{
        // Failed to stop scanning.
    }
}
```

2.2.15 setBeepStatus

Function Name	void setBeepStatus(boolean status);		
Parameter Name	IN/OUT	Type	Descriptions
status	OUT	boolean	The execution results of the function setBeepOn (see 2.1.17) True: succeed. False: failed.

■**Function Descriptions:**
This function will be called back to returned the execution results once the function setBeepOn (see 2.1.17) is called.

■**Sample Code:**

```
@Override
public void setBeepStatus (boolean status){
    //if(status){
        // The function setBeepOn is executed successfully.
    }else{
        // The function setBeepOn is failed to execute.
    }
}
```

2.2.16 setVibrationStatus

Function Name	void setVibrationStatus(boolean status);		
Parameter Name	IN/OUT	Type	Descriptions
status	OUT	boolean	To return the execution results of the function setVibrationOn (see 2.1.19). True: succeed. False: failed.

■**Function Descriptions:**
This function will be called back to returned the execution results once the function setVibrationOn (see 2.1.19) is called.

■**Sample Code:**

```
@Override
public void setVibrationStatus (boolean status){
    if(status){
        // The function setVibrationOn is executed successfully.
    }else{
        // The function setVibrationOn is failed to execute.
```

```
        }
    }
```

2.2.17 setBarcodeTimeoutStatus

Function Name	void setBarcodeTimeoutStatus(boolean status);		
Parameter Name	IN/OUT	Type	Descriptions
status	OUT	boolean	The execution results of the function setBarcodeTimeout (see 2.1.21): True: succeed. False: failed.

■Function Descriptions:

This function will be called back to returned the execution results once the function setBarcodeTimeout (see 2.1.21) is called.

■Sample Code:

```
@Override
public void setBarcodeTimeoutStatus (boolean status){
    if(status){
        // The function setBarcodeTimeout has beed executed successfully.
    }else{
        // The function setBarcodeTimeout has beed executed failed.
    }
}
```

3 Enum

3.1 MRXC2H2DeviceType

Definitions	Descriptions
unknow = 0	no connection
ble = 1	ble connection
spp = 2	spp connetion

3.2 MRXC2H2ConnectionType

Definitions	Descriptions
Connected = 0	connected
Connecting = 1	connecting
Disconnected = 2	disconnected

3.3 MRXC2H2BarcodeType

Definitions	Descriptions
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

Definitions	Descriptions
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