Add dependent libraries:

Add dependencies in build.gradle (Project:)

```
allprojects {

    repositories {
        jcenter()
        maven {url 'https://maven.google.com'}
        maven { url 'https://www.jitpack.io' }
    }
}
```

Add dependencies in build.gradle (Module:)

```
dependencies {
api 'com.github.speedatag:uhf:7.8.7'
api 'com.github.SpeedataG:Device:1.6.7'
}
```

Note: The UHF module will stop working when the battery level is lower than 15%.

"**Android:sharedUserId="android.uid.system"**" has been added to "**AndroidManifest.xml**", please compile apk with "**System sigin**".

API

1. **UHFManager :Judging module instantiation**
2. openDev: **Power-on**
3. closeDev: **Power off**
4. setOnInventoryListener : **Set inventory data monitoring**
5. inventoryStart : **Start inventory**

6. inventoryStop : Stop inventory
7. setOnReadListener : Set read data monitoring
8. readArea: Reading card
9. setOnWriteListener: Set up write data monitoring
10. writeArea : Write card
11. selectCard: Selection card
12. setAntennaPower : Set antenna power
13. getAntennaPower: Read current antenna power value
14. setFreqRegion: Set frequency area
15. getFreqRegion : Reading frequency area
16. setPassword: set password
17. setLock : lock
18. setInvMode : Set inventory mode (R2000)
19. getInvMode : Get inventory mode information (R2000)

UHFManager

Function prototype	UHFManager.getUHFService(Context context)
Functional description	Judgment module instantiation
Parameter description	None
Return type	Instantiated object

Examples:

```
{
IUHFService iuhfService = UHFManager.getUHFService(this);
}
```

Function prototype	public int openDev( )
Functional description	Open the UHF module
Parameter description	None
Return type	0 represents success; -1 represents failure.

openDev

```
1.  iuhfService.openDev();
```

closeDev

Function prototype	public void closeDev()

Function prototype	public void closeDev()
Functional description	Closing the UHF module
Parameter description	None
Return type	None

Inventory

public void inventoryStart();

Function prototype	public void setOnInventoryListener(OnSpdInventoryListener onSpdInventoryListener)
Functional description	Start the inventory
Parameter description	OnSpdInventoryListener

Return type	None

inventoryStart Start the inventory

Function prototype	public void inventoryStart()
Functional description	Start the inventory
Parameter description	None
Return type	None

```
// Sample code
iuhfService.setOnInventoryListener(new OnSpdInventoryListener() {
@Override
public void getInventoryData(SpdInventoryData var1) {
//SpdInventoryData
```

```
6.  //String epc Card EPC (HEX)
7.  //String rssi signal intensity
8.  //String tid Store TID or USER data (only R2000 module support)
9.  }
10. });
11. iuhfService.inventoryStart();
```

inventoryStop

Function prototype	public void inventoryStop()
Functional description	Stop the inventory
Parameter description	None
Return type	None

Read tag

setOnReadListener

readArea

Function	
Function prototype	public int readArea(int area, int addr, int count, String passwd)
Functional description	Read the value of count (calculated in word) from the addr location in the label area area (calculated in word), and passwd is the access password.
Parameter description	1. the area to be read by area can only be 0 (representing reserved area), 1 (representing EPC area), 2 (representing TID area) and 3 (representing user area).
	2. addr should read the address and calculate it in word. The UHF tag read and write unit is word (HEX), so the address is also calculated in word.
	3. Count is a block number, and the unit is word
	4.Passwd, access password, if it is not locked, please fill in 00000000.
Return type	Successful return 0, not 0 as a parameter error.

```
1.  //sample code
2.  iuhfService.setOnReadListener(new OnSpdReadListener() {
3.  @Override
4.  public void getReadData(SpdReadData var1) {
5.  //SpdReadData
```

```
6.  //int status Read the state, succeed, or fail
7.  //byte[] EPCData The EPC of the tag
8.  //byte[] ReadData Read data
9.  //int EPCLen The length of EPC
10. //int DataLen Read the length of the data
11. //int RSS signal intensity
12. }
13. });
14. int readArea = iuhfService.readArea(1,2,6 , "00000000");
15. if (readArea != 0) {
16. // Parameter error
17. }
```

Write card

setOnWriteListener

Function prototype	public void setOnWriteListener(OnSpdWriteListener onSpdWriteListener)
Functional description	Setup write data listener

Function prototype	public void setOnWriteListener(OnSpdWriteListener onSpdWriteListener)
Parameter description	OnSpdWriteListener
Return type	None

writeArea

Function prototype	public int writeArea(int area, int addr, int count, String passwd, byte[] content)
Functional description	Write the data in content to the label area area, and write it in the position of addr (calculated by word).
Parameter description	1. the area to be read by area can only be 0 (representing reserved area), 1 (representing EPC area), 2 (representing TID area) and 3 (representing user area).
	2. addr should read the address and calculate it in word. The UHF tag read and write unit is word (HEX), so the address is also calculated in

	word.
	3. Count is a block number, and the unit is word
	4.Passwd, access password, if it is not locked, please fill in 00000000.
	5. content　Data to be written
Return type	The success returns 0; it is not a 0 parameter error.

```
1.  //sample code
2.  //↓↓↓↓↓↓↓↓↓↓↓↓Focus attention↓↓↓↓↓↓↓↓↓↓↓↓
3.  // The write operation will continue to be written, and the callback will sometimes return several times. When the
    callback status is written to 0, it will be written successfully.
4.  iuhfService.setOnWriteListener(new OnSpdWriteListener() {
5.  @Override
6.  public void getWriteData(SpdWriteData var1) {
7.  //SpdWriteData
8.  //int status Write status, success or failure
9.  //byte[] EPCData The EPC of the tag
10. //int EPCLen The length of EPC
11. //int RSS signal intensity
12. }
13. });
14. int writeArea = iuhfService.writeArea(1,2,2,"00000000",new                                    byte[4]);
15. if (writeArea != 0) {
16. // Parameter error
17. }
```

Select tag (mask)

selectCard

Function prototype	public int selectCard(int bank,String str,boolean mFlag)
Functional description	Select the tag that needs to be operated
Parameter description	Bank: the area of the label to mask, and 1: represents the EPC area. 2: representing the TID area. 3: representing the USER area.
Parameter description	str: The information of the label for the mask (HEX).
Parameter description	mFlag： Whether to select a card, select card to send true, do not select card operation, and then call this parameter to send false.
Return type	Successful return 0, failed to return to -1

```
1.  // Sample code
2.  iuhfService.selectCard(1, "", false); // cancel
3.  iuhfService.selectCard(1, epcStr, true); // select
```

Acquisition and setting of power / frequency band

setAntennaPower (The 3992 module is introduced into 0, representing false 1 for true)

---

getAntennaPower

Function prototype	**public int getAntennaPower()**
Functional description	Read the current antenna power
Parameter description	None
Return type	Failed to return to -1; successful return 0-30.

setFreqRegion

Function prototype	**public int setFreqRegion(int region)**

Functional description	Setting frequency area
Parameter description	The values can be 0 (840_845), 1 (920_925), 2 (902_928), 3 (865_868) (partial modules incomplete).
Return type	A successful return of 0; a failure to return to -1

getFreqRegion (The FLX module does not support this method)

Function prototype	public int getFreqRegion()
Functional description	Get the current frequency area
Parameter description	None
Return type	Failure to return to -1 successfully returned to one of 0 (840_845), 1 (920_925), 2 (902_928), 3 (865_868).

Set the password

setPassword

Function prototype	public int setPassword(int which, String cur_pass, String new_pass)
Functional description	Set the password
Parameter description	The value of which is 0 (Kill Password) 1 (Access Password).
	cur_pass initial password
	new_pass New password
Return type	The successful return of 0 is not a 0 parameter error. Is it successful to monitor callbacks through setOnWriteListener?

```
•
•  //↓↓↓↓↓↓↓↓↓↓↓↓Focus attention↓↓↓↓↓↓↓↓↓↓↓↓
•  // The write operation will continue to be written, and the callback will sometimes return several times. When the callback status is written to 0, it will be written successfully.

1.  iuhfService.setOnWriteListener(new OnSpdWriteListener() {
2.  @Override
3.  public void getWriteData(SpdWriteData var1) {
4.  //SpdWriteData
5.  //int status Write status, success or failure
6.  //byte[] EPCData The EPC of the tag
```

```
7.  //int EPCLen The length of EPC
8.  //int RSS signal intensity
9.  }
10. });
11. int writeArea = iuhfService.setPassword(1,"00000000","11111111");
12. if (writeArea != 0) {
13. // Parameter error
14. }
```

Lock tag

setLock

Function prototype	public int setLock(int type, int area, String passwd)
Functional description	Set the lock-in state. The access password must be set before this operation. When two areas of KillPassword and AccessPassword are locked, only a correct access password can be read, and EPC, TID, and users are locked in three regions, which can be read normally, but the write needs to provide the correct access password.
Parameter description	1. type operation type, the value can be 0 (unlock tag), 1 (lock tag), 2 (permanently unlock tag), 3 (permanently lock tag).
	2. area optional area, the value can be 0 (inactivated password area), 1 (access password area), 2 (EPC area), 3 (TID area), 4 (user area).
	3. passwd Access the password.
Return type	The successful return of 0 is not a 0 parameter error. Whether or not it is successful to listen to a callback through setOnWriteListener

```
1.
2.  //Sample code
3.  //↓↓↓↓↓↓↓↓↓↓↓↓Focus attention↓↓↓↓↓↓↓↓↓↓↓↓
4.  // The write operation will continue to be written, and the callback will sometimes return several times. When the
    callback status is written to 0, it will be written successfully.
5.  iuhfService.setOnWriteListener(new OnSpdWriteListener() {
```

```
@Override
public void getWriteData(SpdWriteData var1) {
//SpdWriteData
//int status Write status, success or failure
//byte[] EPCData The EPC of the tag
//int EPCLen The length of EPC
//int RSS signal intensity
}
});
int writeArea = iuhfService.setLock(1, 1, "00000000");
if (writeArea != 0) {
// Parameter error
}
```

Setting inventory mode

SetInvMode

Function prototype	public int SetInvMode(int invm, int addr, int length)
Functional description	Set up inventory mode, directly check out TID or user area data.
Parameter description	1. INVM mode 0 (EPC), 1 (EPC+TID), 2 (EPC+USER)
	2. addr in addition to EPC area second area inventory starting address

Function prototype	public int SetInvMode(int invm, int addr, int length)
	3. length except EPC area second regions inventory length (World).
Return type	Successful return 0, not 0 failure

Getting Inventory Pattern Information

GetInvMode

Function prototype	public int GetInvMode(int type)
Functional description	Getting Inventory Pattern Information
Parameter description	1. type 0 (mode) 1 (initial address) 2 (data length)
Return type	[type] is 0 hours (representing EPC mode), 1 (representing EPC+TID mode), and 2 (representing EPC+USER mode). [type] for 1 hours, except for second regions in the EPC area. [type] returned at 2 hours, representing second regions in addition to EPC area (World).