SUNMI RFID SDK Integration and Developer Documentation

Revision Record

No.	Updated At	Component Version	Revision	Written By
1.0.0	2020/02/28	2.6.x	Original version	曾冬阳(Darren Zeng)
1.0.1	2020/04/07	2.6.x	Complete the document, SDK integration, description on interface parameter return, etc.	曾冬阳(Darren Zeng)

1. Introduction

RFID-UHF is mainly used in the inventory taking in warehousing and freight, etc. SUNMI L2k supports RFID-UHF (an UHF handle is needed separately).

2. SDK Integration

2.1. Intro to SDK

SDK includes service connection, device connection status, tag inventory and tag operation interface encapsulation.

SDK directory:

com.sunmi.rfid.constant.CMD.java -- Operation type

com.sunmi.rfid.constant.ParamCts.java -- Callback parameter

com.sunmi.rfid.entity.DataParameter.java -- Data parameter

com.sunmi.rfid.ReaderCall.java – Interface definition

com.sunmi.rfid.RFIDHelper.java -- Tag operation interface

com.sunmi.rfid.ServicesHelper.java -- Service interface implementation

com.sunmi.rfid.RFIDManager.java -- RFID device connection management

2.2. How to Use SDK

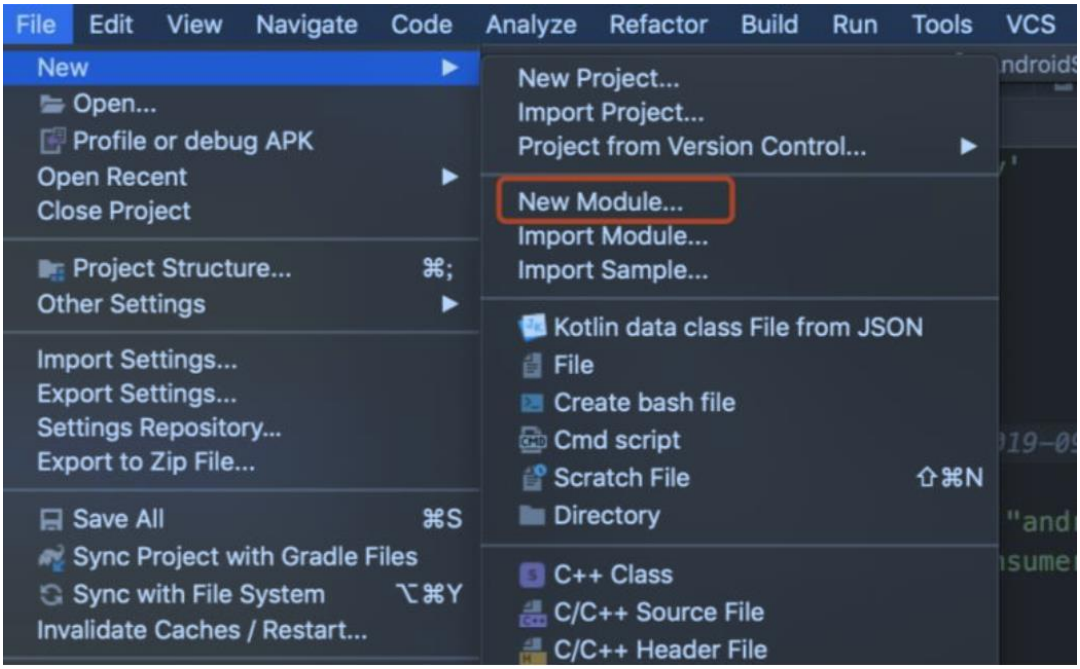
2.2.1. Resource File (AAR File)

Resource：SunmiRFID-SDK-release-v1.0.0.aar

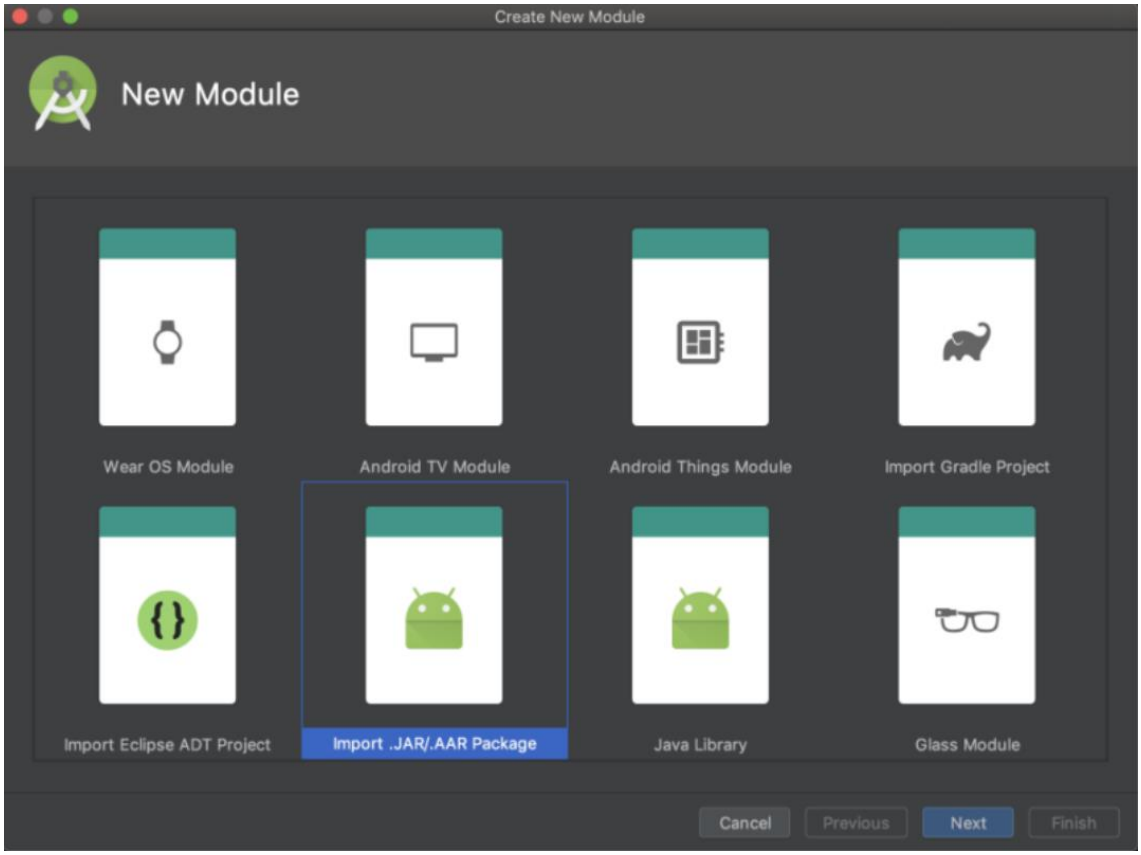
Reference：UHFDemo.zip

2.2.2. Import AAR Package

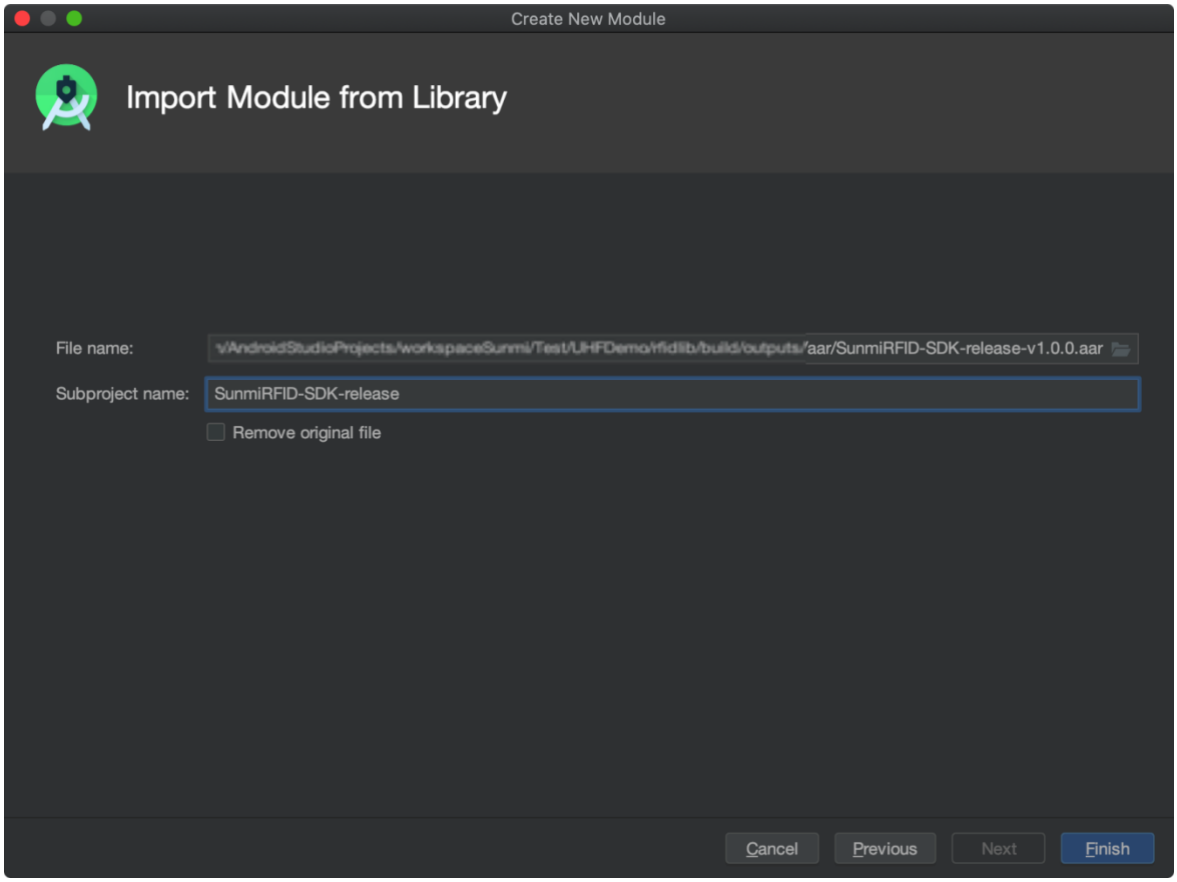
Open the item, then: File -> New -> New Module…



In the new dialogue box: New Module -> Import .JAR/.AAR Package -> Next



Next: Import Module from Library -> Select the SunmiRFID-SDK-release-vX.X.X.aar under the aar file -> Finish

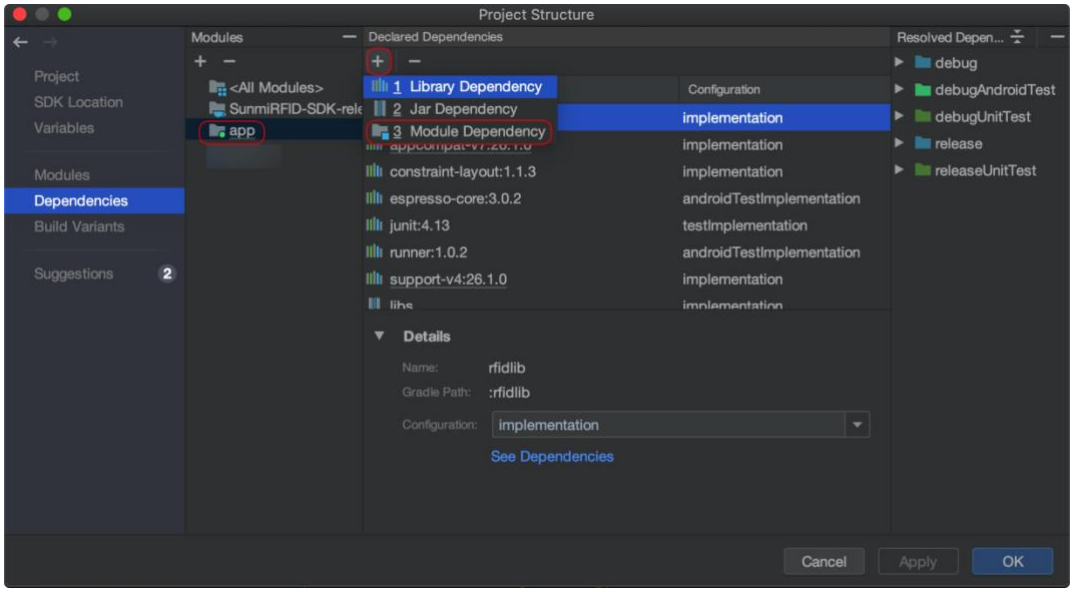


Complete:

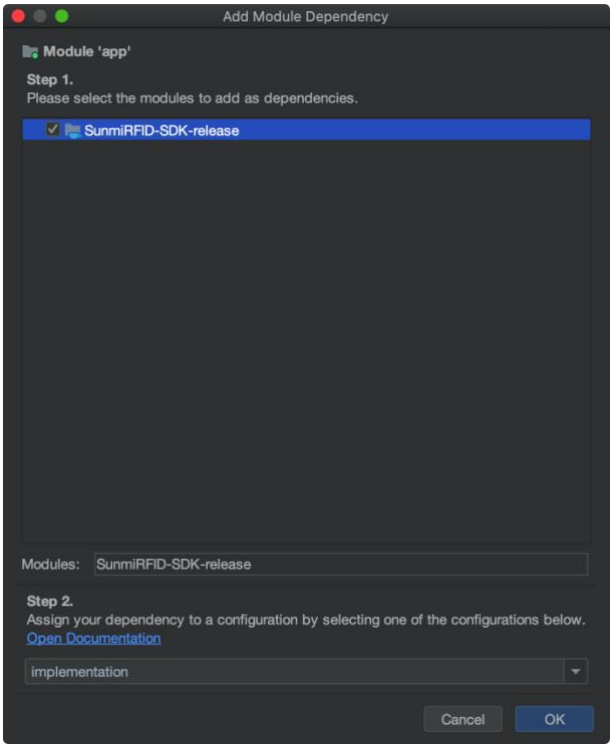
2.2.3. App Module Dependency

Click – open the module settings:

Setting dialogue box: select “app” -> click “+” -> select “Module Dependency”



Add Module Dependency: Check the corresponding module (SunmiRFID-SDK_vX.X.X-release/tes)-> click "OK", and the corresponding SDK can be successfully referenced.



2.3. RFID Manager

2.3.1. Connection Service

RFIDManager.getInstance().setPrintLog(true);//whether to output log
RFIDManager.getInstance().connect(this);

2.3.2. Disconnection Service

RFIDManager.getInstance().disconnect();

2.3.3. Get Helper

RFIDManager.getInstance().getHelper();

2.4. Descriptions of Helper Interface Definition

2.4.1. Basic Information and Data Reception Registration

No.	Method
1	int **getScanModel()** Get RFID type
2	void **registerReaderCall**(**ReaderCall** call) Register data reception
3	void **unregisterReaderCall**() Unregister data reception

1. Get RFID Type

Function: void **getScanModel()**

Parameter: None.

Return:

100 --> None;

101 --> UHF R2000；

2. Register Data Reception

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

Parameter:

call --> Interface callback implementation. Please refer to **3.1.**

Return: None.

3. Unregister Data Reception

Function: void **unregisterReaderCall**()

Parameter: None.

Return: None.

2.4.2. ISO18000-6C Tag Inventory

No.	Method
1	void **inventory(**byte btRepeat**)** Tag inventory – buffer pattern
2	void **realTimeInventory(**byte btRepeat**)**

	Tag inventory – real-time pattern (Auto)
3	void **customizedSessionTargetInventory(byte** btSession, **byte** btTarget**, byte** btSL, **byte** btPhase, **byte** btPowerSave, **byte** btRepeat**)** Tag inventory – real-time pattern (Session). A recommended inventory command.
4	void **fastSwitchAntInventory(byte** btA, **byte** btStayA, **byte** btB, **byte** btStayB, **byte** btC, **byte** btStayC, **byte** btD, **byte** btStayD, **byte** btInterval, **byte** btRepeat) Tag inventory – real-time pattern (Switch). A pattern in which antennas can be quickly switched.

1. 6C Tag Inventory – Buffer Pattern

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

Parameters:

btRepeat --> The number of times the inventory to be repeated. If it is 0xFF, the time used in this inventory round is the shortest time. If there is only one tag in the RF area, the time used in this inventory round is 30-50mS. This parameter value is normally used when quickly polling with several antennas in a four-channel device.

Please note: When the parameter is set as 255(0xFF), the algorithm specially designed for reading a small amount of tags will be enabled. It is more efficient and sensitive to read a small amount of tags. However, this parameter is not suitable to be used to simultaneously read a large amount of tags.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.**, the parameter **params** includes parameters: **ANT_ID, COUNT, READ_RATE, DATA_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to Appendix **4.2.**
2. Callback failure – please refer to **3.1. - 3.**

Description:

Buffer pattern: when a reader receives this command, it will read multiple tags simultaneously. The tag data will be cached into the reader's buffer, and the data can be obtained if using a get buffered tags command. Please refer to Buffer Operation in 2.4.5.

2. 6C Tag Inventory – Real-Time Pattern (Auto)

Function: void **realTimeInventory(byte** btRepeat**)**

Parameter:

btRepeat --> The number of times the inventory to be repeated. If it is 0xFF, the time used in this inventory round is the shortest time. If there is only one tag in the RF area, the time used in this inventory round is 30-50mS. This parameter value is normally used when quickly polling with several antennas in a four-channel device.

Please note: When the parameter is set as 255(0xFF), the algorithm specially designed for reading a small amount of tags will be enabled. It is more efficient and sensitive to read a small amount of tags. However, this parameter is not suitable to be used to simultaneously read a large amount of tags.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1**, the parameter **params** includes parameters**: READ_RATE, DATA_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to Appendix **4.2.**
2. Tag callback – refer to **3.1.-2.**， the parameter **tag** includes parameters: **ANT_ID, TAG_PC, TAG_EPC, TAG_RSSI, TAG_READ_COUNT, TAG_FREQ, TAG_TIME**. For the types of parameters, please refer to Appendix **4.2.**
3. Callback failure – refer to **3.1.-3.**

Description:

Multiple tags will be read and the tag data will be uploaded in real time instead of being cached in the buffer. With this command, the time used in an inventory round of is relatively long and it suits for the reading of a large amount of tags.

Because the hardware is equipped with a dual-CPU structure. The main CPU is used for polling tags and the sub CPU is used for data management. Tags polling and data sending can happen in parallel without occupying each other's time. Therefore, the data transmission of serial port won't affect the work efficiency of the reader.

3. 6C Tag Inventory – Real-Time Pattern (Session)

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

Parameters:

btSession --> The specified inventory session. 00 is S0, 01 is S1, 02 is S2, 03 is S3.
btTarget --> The specified inventory Inventoried Flag. 00 is A, 01 is B.
btSL --> Select Flag; range: 00,01,02,03.
btPhase --> Phase value: 00 – turn off this function; 01 – turn on this function.
btPowerSave --> Save energy – energy saving pattern: this shall be within 0~255.
btRepeat --> The number of times the inventory to be repeated.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1**. the parameter **params** include parameters: **READ_RATE, DATA_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to Appendix **4.2.**
2. Tag callback – refer to **3.1.-2.** the parameter **tag** includes parameters: **ANT_ID, TAG_PC, TAG_EPC, TAG_RSSI, TAG_READ_COUNT, TAG_FREQ, TAG_TIME**. For the types of parameters, please refer to Appendix **4.2.**
3. Callback failure – refer to **3.1.-3.**

Description:

Multiple tags will be read according to the specified session and inventoried flag. The tag data will be uploaded in real time and won't be cached into the reader's buffer. With this command, an inventory round will take a relatively short time, the S1 mode of this command is recommended to be used for common inventory taking.

4. 6C Tag Inventory – Real-Time Pattern (Switch). A Pattern in Which Antennas Can Be Quickly Switched

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

Parameters:

btA --> The 1st antenna (00 – 03) for polling. If the antenna number exceeds 3, there will be no polling.

btStayA/B/C/D --> The number of times a polling will be repeated by an antenna. This can be set respectively for each antenna.

btB --> The 2nd antenna (00 – 03) for polling. If the antenna number exceeds 3, there will be no polling.

btC --> The 3rd antenna (00 – 03) for polling. If the antenna number exceeds 3, there will be no polling.

btD --> The 4th antenna (00 – 03) for polling. If the antenna number exceeds 3, there will be no polling.

btInterval --> The time interval among each antenna's work (unit: mS). There will not be RF output if no antenna is working, and the power consumption can be lowered correspondingly.

btRepeat -->The number of times the inventory to be repeated.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1**. the parameter **params** include parameters: **COMMAND_DURATION, DATA_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to Appendix **4.2.**
2. Tag callback – refer to **3.1.-2.**. The parameter **tag** includes parameters: **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**. For the types of parameters, please refer to Appendix **4.2.**
3. Callback failure – refer to **3.1.-3.**

Description:

Multiple tags will be read at the same time. The tag data will be uploaded in real time and also will be cached in the reader's buffer. The reader will automatically switch antennas following the order of A->H. If there is no tag in the RF area or if there are only one or two tags in the RF area, each antenna will use about 30mS averagely. If there are more tags, the time used will be increased. This is suitable for the application where multiple antennas need to be quickly switched to read tags.

2.4.3. ISO18000-6C Tag Operation

No.	Method
1	void **readTag**(**byte** btMemBank, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryPassWord) Tag operation – read tags
2	void **writeTag**(**byte**[] btAryPassWord, **byte** btMemBank, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryData) Tag operation – write tags
3	void **lockTag**(**byte**[] btAryPassWord, **byte** btMemBank, **byte** btLockType)

	Tag operation – lock tags
4	void **killTag**(**byte**[] btAryPassWord) Tag operation – kill tags
5	void **setAccessEpcMatch**(**byte** btEpcLen, **byte**[] btAryEpc) Tag operation – set the matched EPC to be accessed (EPC match is valid until the next refresh)
6	void **cancelAccessEpcMatch**() Tag operation – clear the matched EPC to be accessed
7	void **getAccessEpcMatch**() Tag operation – get EPC match
8	void **setImpinjFastTid**(**boolean** blnOpen, **boolean** blnSave) Tag operation – set FastTID (only valid to some models of Impinj Monza tags)
9	void **getImpinjFastTid**() Tag operation – inquire FastTID

1. 6C Tag Operation – Read Tags

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

Parameters:

btMemBank --> Tag memory bank: 0x00: RESERVED; 0x01: EPC; 0x02: TID; 0x03: USER.

btWordAdd --> The first word address of the data read. For the value range, please refer to tag specs.

btWordCnt --> The data length, word length or WORD (16 bits) length of the data read; For the value range, please refer to tag specs.

btAryPassWord --> Tag access password. 4 bytes.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.** The parameter **params** include parameters: **TAG_PC, TAG_CRC, TAG_EPC, TAG_DATA, TAG_DATA_LEN, ANT_ID, TAG_READ_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Description:

2.4.3.-5. Set the Matched EPC To Be Accessed should be done first. Tags with the same EPC but different data read will be deemed as different tags.

2. 6C Tag Operation – Write Tags

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

Parameters:

btAryPassWord --> Tag access password. 4 bytes.

btMemBank --> Tag memory area: 0x00:RESERVED; 0x01:EPC; 0x02:TID; 0x03:USER.

btWordAdd --> The first word address of the data written. For the value range, please refer to tag specs; it usually starts from 02 if it was written into EPC memory bank, and PC + CRC are stored in the first four bytes of this area.

btWordCnt --> The data length, word length or WORD (16 bits) length of the data written; For the value range, please refer to tag specs.

btAryData --> The data written. The length is btWordCnt * 2.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.** The parameter **params** include parameters: **TAG_PC, TAG_CRC, TAG_EPC, ANT_ID, TAG_READ_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Description:

2.4.3.-5. Set the Matched EPC To Be Accessed should be done first. Tags with the same EPC but different data read will be deemed as different tags.

3. 6C Tag Operation – Lock Tags

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

Parameters:

btAryPassWord --> Tag access password. 4 bytes.

btMemBank --> The memory bank to be locked: 0x01: User Memory, 0x02: TID Memory, 0x03: EPC Memory, 0x04: Access Password, 0x05: Kill Password.

btLockType --> The types of lock operation: 0x00: open, 0x01: lock, 0x02: permanently open, 0x03: permanently locked.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.** **TAG_PC, TAG_CRC, TAG_EPC, ANT_ID, TAG_READ_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Description:

2.4.3.-5. Set the Matched EPC To Be Accessed should be done first. Tags with the same EPC but different data read will be deemed as different tags.

4. 6C Tag Operation – Kill Tags

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

Parameter:

btAryPassWord --> Tag access password. 4 bytes.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. 1. Successful callback – refer to **3.1. - 1.** The parameter **params** include parameters: **TAG_PC, TAG_CRC, TAG_EPC, ANT_ID, TAG_READ_COUNT, START_TIME, END_TIME.** For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Description:

2.4.3.-5. Set the Matched EPC To Be Accessed should be done first. Tags with the same EPC but different data read will be deemed as different tags.

5. 6C Tag Operation – Set the Matched EPC To Be Accessed (EPC Match Is Valid Until the Next Refresh)

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

Parameters:

btEpcLen --> EPC length.

btAryEpc --> EPC number, which is made up of EpcLen bytes.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.** The parameter **params** include parameters: **START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Callback:

Tags with the same EPC but with different data read will be deemed as different tags.

6. 6C Tag Operation – Clear the Matched EPC To Be Accessed

Function: void **cancelAccessEpcMatch**()

Parameter: None.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.** The parameter **params** include parameters: **START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Description:

Tags with the same EPC but with different data read will be deemed as different tags.

7. 6C Tag Operation – Get EPC Match

Function: void **getAccessEpcMatch**()

Parameter: None.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.** The parameter **params** include parameters: **START_TIME, TAG_ACCESS_EPC_MATCH, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Description:

Tags with the same EPC but with different data read will be deemed as different tags.

8. 6C Tag Operation – Set Fast TID

Function: void **setImpinjFastTid**(**boolean** blnOpen, **boolean** blnSave)

Parameters:

blnOpen --> FastTID on or off status.

blnSave --> Save the configuration into the internal Flash, which will prevent it from being lost due to an outage.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.** The parameter **params** include parameters: **START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Description:

This function is only effective to some models of Impinj Monza tags.

This function recognizes EPC and TID at the same time, thus drastically enhancing the efficiency of reading TID.

After turning on this function, tags of specific model will package TID into EPC during inventory. Therefore, the PC of a tag will be changed, and the original PC + EPC will become: the modified PC + EPC + (CRC of EPC) + TID.

If there is something goes wrong when recognizing a TID, the original PC + EPC will be uploaded. ★Please turn off this function if you do not need it to avoid unnecessary time used.

9. 6C Tag Operation – Inquire Fast TID

Function: void **getImpinjFastTid**()

Parameter: None.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1.** The parameter **params** include parameters: **START_TIME, TAG_MONZA_STATUS, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1. - 3.**

Description:

None.

2.4.4. ISO18000-6B Tag Inventory and Tag Operations

No.	Method
1	void **iso180006BInventory**() Tag inventory – real-time pattern
2	void **iso180006BReadTag**(**byte**[] btAryUID, **byte** btWordAdd, **byte** btWordCnt) Tag operation – read tags
3	void **iso180006BWriteTag**(**byte**[] btAryUID, **byte** btWordAdd, **byte** btWordCnt, **byte**[] btAryBuffer) Tag operation – write tags
4	void **iso180006BLockTag**(**byte**[] btAryUID, **byte** btWordAdd) Tag operation – lock tags
5	void **iso180006BQueryLockTag**(**byte**[] btAryUID, **byte** btWordAdd) Tag operation – locked tags query

1. 6B Tag Inventory – Real-Time Pattern

Function: void **iso180006BInventory**()

Parameter: None.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – please refer to **3.1. - 1**. The parameter **params** include parameters: **ANT_ID, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Tag callback – refer to **3.1.-2.** The parameter **tag** includes parameters: **ANT_ID, TAG_UID, TAG_READ_COUNT, TAG_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
3. Callback failure – refer to **3.1.-3.**

Description:

Multiple tags will be read and tag data will be uploaded in real time and won't be cached in the reader's buffer. With this command, an inventory round will take a relatively long time, and it is suitable for the reading of a large amount of tags.

For the hardware is equipped with a dual-CPU structure. The main CPU is used for polling tags and the sub CPU is used for data management. Tags polling and data sending can happen in parallel without occupying each other's time. Therefore, the data transmission of serial port won't affect the efficiency of the reader.

2. 6B Tag Operation – Read Tags

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

Parameters:

btAryUID --> The UID of the tag being operated. 8 bytes.
btWordAdd --> The first word address of the data read.
btWordCnt --> The length of the data read.

Callback:

Data reception registration is needed, please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1** The parameter **params** include parameters: **ANT_ID, TAG_DATA, TAG_DATA_LEN, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1.-3.**

Description:

None.

3. 6B Tag Operation -Write Tags

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

Parameters:

btAryUID --> The UID of the tag being operated. 8 bytes.
btWordAdd --> The first word address of the data written.
btWordCnt --> The length of the data written.
btAryBuffer --> Data.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1** The parameter **params** include parameters: **ANT_ID, TAG_DATA_LEN, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1.-3.**

Description:

Multiple bytes can be written at one time. Once the writing of a byte goes wrong, this command won't write the following data.

And the command will return the number of bytes that have been successfully written.

4. 6B Tag Operation – Lock Tags

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

Parameters:

btAryUID --> The UID of the tag to be operated. 8 bytes.

btWordAdd --> The address to be locked.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1** The parameter **params** include parameters: **ANT_ID, TAG_STATUS (0x00**: the operation lock is successfully conducted, **0xFE**: locked status, **0xFF**: failed to lock the tag**), START_TIME, END_TIME.** For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1.-3.**

Description:

None.

5. 6B Tag Operation – Locked Tag Query

Function: void **iso180006BQueryLockTag**(**byte**[] btAryUID, **byte** btWordAdd)

Parameters:

btAryUID --> The UID of the tag being operated. 8 bytes.

btWordAdd --> The address to be inquired.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1** The parameter **params** include parameters: **ANT_ID, TAG_STATUS (0x00: status - not locked, 0xFE: status - locked), START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1.-3.**

Description:

None.

2.4.5. ISO18000-6C Buffer Operation

No.	Method
1	void **getInventoryBuffer**() Buffer operation – get buffered tags

2	void **getAndResetInventoryBuffer**() Tag operation – get buffered tags and reset buffer
3	void **getInventoryBufferTagCount**() Tag operation – get the number of buffered tags
4	void **resetInventoryBuffer**() Tag operation – reset buffer

1. Buffer – Get Buffered Tags

Function: void **getInventoryBuffer**()

Parameter: None.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. 1. Successful callback – refer to **3.1. - 1** The parameter **params** include parameters: **TAG_PC, TAG_CRC, TAG_EPC, ANT_ID, TAG_RSSI, TAG_READ_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1.-3.**

Description:

Buffered tag data will be uploaded, and the number equals to the number of buffered tags (no repetitive data).

2. Buffer Operation – Get Buffered Tags and Reset Buffer

Function: void **getAndResetInventoryBuffer**()

Parameter: None.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1** The parameter **params** include parameters: **TAG_PC, TAG_CRC, TAG_EPC, ANT_ID, TAG_RSSI, TAG_READ_COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1.-3.**

Description:

The buffered tag data will be uploaded, and the number equals to the number of tags buffered (no repetitive data) and the buffered tags will be cleared.

3. Buffer Operation – Get the Number of Buffered Tags

Function: void **getInventoryBufferTagCount**()

Parameter: None.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1** The parameter **params** include parameters: **COUNT, START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1.-3.**

Description:

None.

4. Buffer Operation – Reset Buffer

Function: void **resetInventoryBuffer**()

Parameter: None.

Callback:

Data reception registration is needed. Please refer to **2.4.1.-2.** and **2.4.1.-3.**

1. Successful callback – refer to **3.1. - 1** The parameter **params** include parameters: **START_TIME, END_TIME**. For the types of parameters, please refer to **Appendix 4.2.**
2. Callback failure – refer to **3.1.-3.**

Description:

None.

3. Callback Descriptions

3.1. ReaderCall – Description

No.	Method
1	void **onSuccess**(**byte** cmd**, DataParameter** params) throws RemoteException Successful callback – the operation is successful
2	void **onTag**(**byte** cmd, **byte** state, **DataParameter** tag) throws RemoteException Tag operation – tag callback
3	void **onFiled**(**byte** cmd, **byte** errorCode, **String** msg) throws RemoteException Callback failure – the operation failed

1. Successful Callback – The Operation Is Successful

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

Parameters:

cmd --> Operation type. Refer to **2.4.** or **Appendix 4.1.**

params --> Data parameter. Refer to **2.4.** or **4.2.** Example: to get the number of 6C tags after an inventory round: params.getInt(ParamCts.DATA_COUNT, 0)

Description:

Buffered tag data will be uploaded, and the number equals to the number of buffered data (no repetitive data).

2. Tag Operation – Tag Callback

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

Parameter:

cmd --> Operation type. Refer to **2.4.** or **Appendix 4.1.**

state --> Tag status:

ParamCts.FOUND_TAG,(0x01) – New tag;

ParamCts.UPDATE_TAG,(0x02) – Updated tag.

tag --> Tag parameter. For details, please refer to **2.4.** and **4.2.** Example: to get the data of the EPC memory bank of a 6C tag, tag.getString(ParamCts.TAG_EPC);

Description:

Tag data will be returned by this interface.

3. Callback Failure – The Operation Failed

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

Parameter:

cmd --> Operation type. Refer to **2.4.** or **Appendix 4.1.**

errorCode --> Error code. Refer to **Appendix 4.3.**

msg --> Error message.

Description:

None.

3.2 Table of Actions, Descriptions and Parameters

Action	Description	Parameter (Refer to 4.2.)
com.sunmi.rfid.unFoundReader	No relevant device has been found	
com.sunmi.rfid.onLostConnect	The connection is disconnected	
com.sunmi.rfid.batteryLowElec	Low battery	ParamCts.BATTERY_REMAINING_PERCENT

4. Appendix

4.1. Table of Parameters, Values and Descriptions

Parameter (CMD)	Value	Description
CMD.INVENTORY	0x80	6C Tag inventory (buffer pattern)
CMD.READ_TAG	0x81	6C Tag operation – read tags
CMD.WRITE_TAG	0x82	6C Tag operation – write tags
CMD.LOCK_TAG	0x83	6C Tag operation – lock tags
CMD.KILL_TAG	0x84	6C Tag operation – kill tags
CMD.SET_ACCESS_EPC_MATCH	0x85	6C Tag operation – set the matched EPC to be accessed 6C Tag operation – clear the matched EPC to be accessed
CMD.GET_ACCESS_EPC_MATCH	0x86	6C Tag operation – get the matched EPC to be accessed
CMD.REAL_TIME_INVENTORY	0x89	6C Tag inventory – real-time pattern (Auto)
CMD.FAST_SWITCH_ANT_INVENTORY	0x8A	6C Tag inventory – real-time pattern (Switch)
CMD.CUSTOMIZED_SESSION_TARGET_INVENTORY	0x8B	6C Tag inventory – real-time pattern (Session)
CMD.SET_IMPINJ_FAST_TID	0x8C	6C Tag operation – set Fast TID
CMD.SET_AND_SAVE_IMPINJ_FAST_TID	0x8D	6C Tag operation – set Fast TID and save
CMD.GET_IMPINJ_FAST_TID	0x8E	6C Tag operation – inquire Fast TID
CMD.ISO18000_6B_INVENTORY	0xB0	6B Tag inventory – real-time
CMD.ISO18000_6B_READ_TAG	0xB1	6B Tag operation – read tags
CMD.ISO18000_6B_WRITE_TAG	0xB2	6B Tag operation – fto be tags
CMD.ISO18000_6B_LOCK_TAG	0xB3	6B Tag operation – lock tags
CMD.ISO18000_6B_QUERY_LOCK_TAG	0xB4	6B Tag operation – Locked tag query
CMD.GET_INVENTORY_BUFFER	0x90	6C Buffer operation – get buffered tags
CMD.GET_AND_RESET_INVENTORY_BUFFER	0x91	6C Buffer operation – get buffered tags and reset buffer
CMD.GET_INVENTORY_BUFFER_TAG_COUNT	0x92	6C Buffer operation – get the number of buffered tags
CMD.RESET_INVENTORY_BUFFER	0x93	6C Buffer operation – reset buffer

4.2. Table of Parameters, Types and Descriptions

Parameter (ParamCts)	Type	Description
Broadcast Parameter Part		
BATTERY_REMAINING_PERCENT	int	Remaining battery (1-100)
Tag Inventory Part		
ANT_ID	byte	The current antenna ID (0x00-0x03)
COUNT	int	The number of buffered tags
READ_RATE	int	The tag recognition rate
DATA_COUNT	int	The number of tags read
START_TIME	long	The start time of an operation (unit: ms)
END_TIME	long	The end time of an operation (unit: ms)
COMMAND_DURATION	int	The total time used in 6C inventory (Switch pattern). Unit: ms.
Tag Part		
TAG_UID	String	6B Tag UID data
TAG_PC	String	6C Tag PC data
TAG_EPC	String	6C Tag EPC data
TAG_CRC	String	6C Tag CRC data
TAG_RSSI	String	6C Tag RSSI data
TAG_READ_COUNT	int	The number of times a tag to be read
TAG_FREQ	String	The RF frequency of tag reading
TAG_TIME	long	When the tag was last updated. Unit: ms.
TAG_DATA	String	Tag data (tag operation)
TAG_DATA_LEN	int	Tag data length (tag operation)
TAG_ANT_1	int	6C inventory (Switch pattern) – the number of times a tag to be recognized by Ant1
TAG_ANT_2	int	6C inventory (Switch pattern) – the number of times a tag to be recognized by Ant2
TAG_ANT_3	int	6C inventory (Switch pattern) – the number of times a tag to be recognized by Ant3
TAG_ANT_4	int	6C inventory (Switch pattern) – the number of times a tag to be recognized by Ant4
TAG_ACCESS_EPC_MATCH	String	6C tag operation – the matched EPC to be accessed
TAG_MONZA_STATUS	byte	6C tag operation – Fast TID
TAG_STATUS	byte	6B the status of a locked tag

4.3. Error Codes and Descriptions

Num	Code	Name	Description
1	0x10	success	The operation has been completed successfully
2	0x11	fail	The execution of the command failed
3	0x20	mcu_reset_error	CPU reset error
4	0x21	cw_on_error	CW enabling error
5	0x22	antenna_missing_error	The antenna is not connected
6	0x23	write_flash_error	Writing Flash error
7	0x24	read_flash_error	Reading Flash error
8	0x25	set_output_power_error	Transmit power setting error
9	0x31	tag_inventory_error	Tag inventory error
10	0x32	tag_read_error	Tag reading error
11	0x33	tag_write_error	Tag writing error
12	0x34	tag_lock_error	Tag locking error
13	0x35	tag_kill_error	Tag killing error
14	0x36	no_tag_error	Error – no operable tag
15	0x37	inventory_ok_but_access_fail	Inventory succeeded but access failed
16	0x38	buffer_is_empty_error	The buffer is empty
17	0x3C	nxp_custom_command_fail	NXP chip customization command error
18	0x40	access_or_password_error	Tag access error or access password error
19	0x41	parameter_invalid	Invalid parameter
20	0x42	parameter_invalid_wordCnt_too_long	wordCnt parameter exceeds the specified length
21	0x43	parameter_invalid_membank_out_of_range	MemBank parameter exceeds the specified range
22	0x44	parameter_invalid_lock_region_out_of_range	The parameter Lock data area exceeds the range
23	0x45	parameter_invalid_lock_action_out_of_range	LockType parameter exceeds the range
24	0x46	parameter_reader_address_invalid	Invalid reader address
25	0x47	parameter_invalid_antenna_id_out_of_range	Antenna_id exceeds the range
26	0x48	parameter_invalid_output_power_out_of_range	The output power parameter exceeds the range
27	0x49	parameter_invalid_frequency_region_out_of_range	The RF range parameter exceed the specified range
28	0x4A	parameter_invalid_baudrate_out_of_range	The baud rate parameter exceeds the range
29	0x4B	parameter_beeper_mode_out_of_range	The beeper setting parameter exceeds the range
30	0x4C	parameter_epc_match_len_too_long	EPC match length exceeds the range
31	0x4D	parameter_epc_match_len_error	EPC match length error
32	0x4E	parameter_invalid_epc_match_mode	EPC match parameter exceeds the range

33	0x4F	parameter_invalid_frequency_range	Invalid frequency range setting parameter
34	0x50	fail_to_get_RN16_from_tag	Failed to receive the RN16 of the tag
35	0x51	parameter_invalid_drm_mode	The parameter DRM setting error
36	0x52	pll_lock_fail	Failed to lock PLL
37	0x53	rf_chip_fail_to_response	No response from the RF chip
38	0x54	fail_to_achieve_desired_output_power	Failed to achieve the desired output power
39	0x55	copyright_authentication_fail	Failed to pass the copyright authentication
40	0x56	spectrum_regulation_error	Spectrum regulation setting error
41	0x57	output_power_too_low	The output power is too low
42	0xEE	fail_to_get_rf_port_return_loss	Failed to measure the return loss
43	0x03	un_check_reader	No RFID device has been found

4.4. RSSI Parameters and Signal Strengths

RSSI Parameter	Corresponding Signal Strength (Logarithm)	RSSI Parameter	Corresponding Signal Strength (Logarithm)
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