

# RFID Development guide

## Document revision record

version number	*Change state	Brief description	date	developer	Date of approval	approver
V1.0.0	C	initial version	2023/10/25	LJH		

\*Change status: C = create, A = increase, M = modify, D = delete

## Document approval record

order number	Examiner	role	Approval date	Sign	remarks

## catalogue

RFID Development guide .....	1
1. Abstract .....	5
2. matters need attention .....	5
2.1 Configure the android development environment .....	5
2.2 Read and write the basic process .....	6
3. interface .....	6
3.1 Connect / close the RFID .....	6
3.1.1 power 【Module upper / lower power】 .....	6
3.1.2 Connect 【Connect the serial port】 .....	7
3.1.3 Connect 【Connect the serial port】 .....	7
3.1.4 Disconnect 【Close the serial port】 .....	7
3.1.5 createProduct 【Set up the UHF module】 .....	8
3.1.6 createProduct 【Set up the UHF module】 .....	8
3.2 18000-6C command .....	8
3.2.1 StartRead 【Start the inventory】 .....	8
3.2.2 StopRead 【Stop inventory】 .....	9
3.2.3 ReadDataByEPC 【The EPC mask reads the data】 .....	9
3.2.4 ReadDataByTID 【The TID mask reads the data】 .....	10

3.2.5 WriteDataByEPC 【The EPC mask is used to write the data】 .....	11
3.2.6 WriteDataByTID 【The TID mask is to write the data】 .....	11
3.2.7 WriteEPCByTID 【The TID mask overwrites the EPC number】 .....	12
3.2.8 Lock 【Label lock】 .....	13
3.2.9 Kill 【Label destruction】 .....	14
3.3 Custom command.....	14
3.3.1 SetCallBack 【Set the callback interface】 .....	14
3.3.2 GetUHFInformation 【Obtain the module parameter information】 .	15
3.3.3 SetRfPower 【Set the power】 .....	16
3.3.4 SetRegion 【Set the frequency point】 .....	16
3.3.5 getInventoryTagMapList 【Get the read card tag array】 .....	17
3.3.6 getInventoryTagResultList 【Get the read card tag result array】 .....	17
3.3.7 SetPowerMode 【Set power consumption mode】 .....	17
3.3.8 MeasureTemperature 【 measure the temperature】 .....	18
3.3.9 MeasureReturnLoss 【Measure back damage】 .....	18
3.3.10 SetAntenna 【Set the antenna number】 .....	19
3.3.11 SetWorkMode 【Set the working mode】 .....	19
3.3.12 SetBaudRate 【Set the Porter rate】 .....	20
3.3.13 GetModuleVersion 【Get the module version】 .....	20

3.3.14 beginSound 【Turn on the sound】 .....	20
3.3.15 setSoundId 【set sound】 .....	21
3.3.16 playSound 【play Sound】 .....	21
4 Appendix 1 .....	22

# 1. Abstract

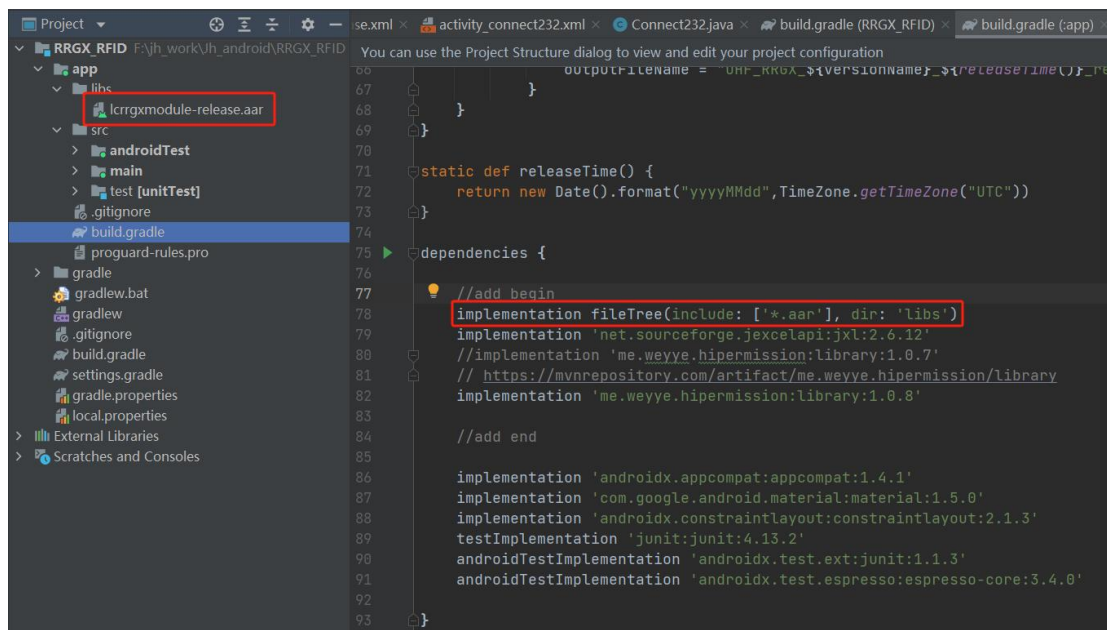
To facilitate secondary development, we provide a library of functions that can be run on the Java platform. The library was written in the Java language.

## 2. matters need attention

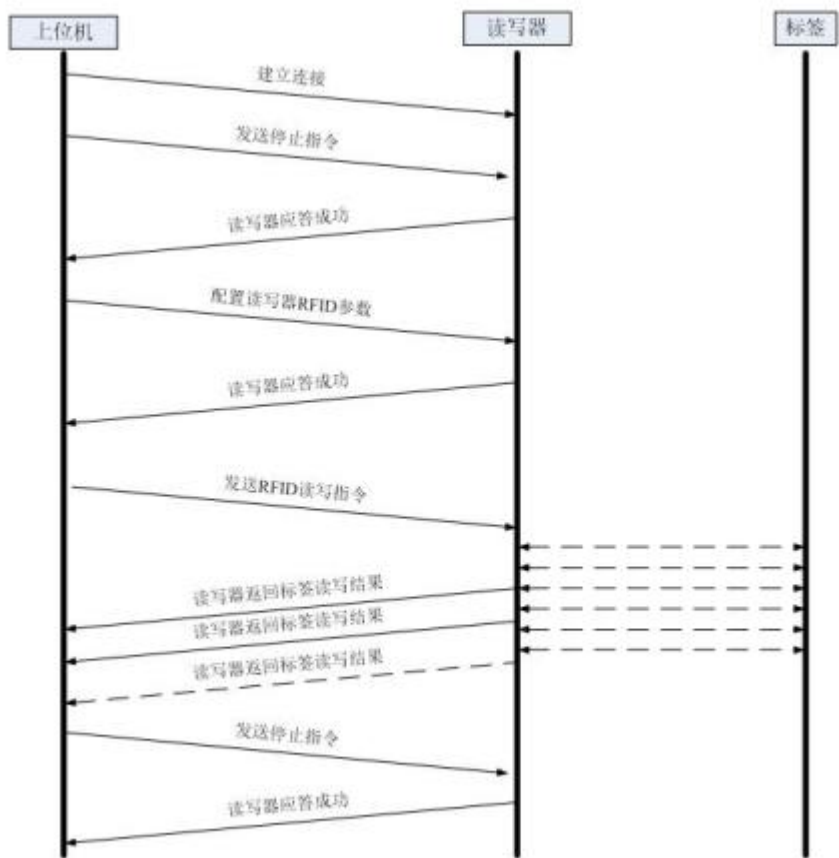
### 2.1 Configure the android development

#### environment

1. Development tools recommend versions Android Studio 4.0 and above.
2. SDK integration method: copy the lccrgxmodule-release.aar to the libs directory of your project, modify the build.gradle file, and add the lccrgxmodule-release.aar dependency.



## 2.2 Read and write the basic process



## 3. interface

### 3.1 Connect / close the RFID

#### 3.1.1 power [Module upper / lower power]

definition	void power(String state)		
illustration	UHF Module power up / down.		
parameter	name	type	remarks
	state	String	State. 1: power on, 0: power off.
return	none		

<b>identifying code</b>	PowerUtil.power( "1" )
-------------------------	------------------------

### 3.1.2 Connect 【Connect the serial port】

<b>definition</b>	int Connect();
<b>illustration</b>	Open the serial port connected to the reader and connect up.
<b>parameter</b>	<b>none</b>
<b>return(int)</b>	success: 0 ; fail: not-0 ; (View the return value error code table)
<b>identifying code</b>	none

### 3.1.3 Connect 【Connect the serial port】

<b>definition</b>	int Connect (String ComPort, int BaudRate);		
<b>illustration</b>	Open the serial port that connects to the reader.		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	ComPort	String	Serial port name, fixed as "/dev/ttyS3"
	BaudRate	int	Serial port communication rate (generally 115200)
<b>return(int)</b>	success: 0 ; fail: not-0 ; (View the return value error code table)		
<b>identifying code</b>	Connect().		

### 3.1.4 Disconnect 【Close the serial port】

<b>definition</b>	int Disconnect();
<b>illustration</b>	Undo the connection between the specified serial port and the reader and release the corresponding resource.
<b>parameter</b>	<b>none</b>

<b>return(int)</b>	success: 0 ; fail: not-0 ; (View the return value error code table)
<b>identifying code</b>	none

### 3.1.5 createProduct 【Set up the UHF module】

<b>definition</b>	ILcUhfProduct createProduct();
<b>illustration</b>	The system automatically determines the UHF module
<b>parameter</b>	none
<b>return(UHF module)</b>	ILcUhfProduct
<b>identifying code</b>	Reader.rllib = new LcModule(this).createProduct();

### 3.1.6 createProduct 【Set up the UHF module】

<b>definition</b>	ILcUhfProduct createProduct(int Type);
<b>illustration</b>	Manual value transfer to judge the UHF module
<b>parameter</b>	none
<b>return(UHF module)</b>	ILcUhfProduct
<b>identifying code</b>	Reader.rllib = new LcModule(this).createProduct(0x20);

## 3.2 18000-6C command

### 3.2.1 StartRead 【Start the inventory】

<b>definition</b>	Public int StartRead()
<b>illustration</b>	Start taking inventory of the labels, and the resulting label EPC or TID data is returned by callback. If the ReaderParameter parameter is TIDLen > 0, the TID



	<p>data is returned. Otherwise, the EPC data is returned.</p> <p>Note that for TID inventory, TID starting address and length are configured according to the actual label specification.</p>
<b>parameter</b>	none
<b>return(void )</b>	none
<b>identifying code</b>	None.

### 3.2.2 StopRead 【Stop inventory】

<b>definition</b>	Public void StopRead();
<b>illustration</b>	Stop the current ongoing inventory operation
<b>parameter</b>	none
<b>return(void )</b>	none
<b>identifying code</b>	none

### 3.2.3 ReadDataByEPC 【The EPC mask reads the data】

<b>definition</b>	public String ReadDataByEPC(String EPCStr,byte Mem,byte WordPtr,byte Num,byte Password[]);		
<b>illustration</b>	Read each storage area data by the EPC mask		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	EPCStr	String	The 16 decimal EPC number of the label
	Mem	byte	0- Storage area to be read, 1- Password area, the first 2 words are the destruction password, the last 2 words are the access password

			2- EPC distinguish 3- TID distinguish 4- USER distinguish
	WordPtr	byte	Starting word address for the read
	Num	byte	The length of the word read
	Password	byte[]	Access password for the tag, 4 bytes
<b>return(String)</b>	If the return string length is 2, represents the 16 decimal string of the error code, otherwise return the read data of the label.		
<b>identifying code</b>	none		

### 3.2.4 ReadDataByTID [The TID mask reads the data]

<b>definition</b>	public String ReadDataByTID(String TIDStr,byte Mem,byte WordPtr,byte Num,byte Password[]);		
<b>illustration</b>	Read each storage data by TID mask (TID must be the data from the starting address 0)		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	TIDStr	String	The 16th decimal TID number of the label
	Mem	byte	0- Storage area to be read, 1- Password area, the first 2 words are the destruction password, the last 2 words are the access password 2- EPC distinguish 3- TID distinguish 4- USER distinguish
	WordPtr	byte	Starting word address for the read
	Num	byte	The length of the word read
	Password	byte[]	Access password for the tag, 4 bytes
<b>return(String)</b>	If the return string length is 2, represents the 16 decimal string of the error code, otherwise return the read data of the label.		
<b>identifying code</b>	none		

### 3.2.5 WriteDataByEPC [The EPC mask is used to write the data]

<b>definition</b>	public int WriteDataByEPC(String EPCStr,byte Mem,byte WordPtr,byte Password[],String wdata);		
<b>illustration</b>	Write data to each storage through the EPC mask.		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	EPCStr	String	The 16 decimal EPC number of the label
	Mem	byte	0- Storage area to be written, 1- Password area, the first 2 words are the destruction password, the last 2 words are the access password 2- EPC distinguish 3- TID distinguish 4- USER distinguish
	WordPtr	byte	The word address of the write
	Password	byte[]	Access password for the tag, 4 bytes
	wdata	String	A 16 decimal string with written data must be an integer multiple of 4
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.2.6 WriteDataByTID [The TID mask is to write the data]

<b>definition</b>	public int WriteDataByTID(String TIDStr,byte Mem,byte WordPtr,byte Password[],String wdata);		
<b>illustration</b>	Write data to each storage via the TID mask. (Request a TID starting address starting from 0)		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	TIDStr	String	The 16th decimal TID number of the label

	Mem	byte	5- Storage area to be written, 6- Password area, the first 2 words are the destruction password, the last 2 words are the access password 7- EPC distinguish 8- TID distinguish 9- USER distinguish
	WordPtr	byte	The word address of the write
	Password	byte[]	Access password for the tag, 4 bytes
	wdata	String	Data to be written, a 16 decimal string, and the length must be an integer multiple of 4
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.2.7 WriteEPCByTID 【The TID mask overwrites the EPC number】

<b>definition</b>	public int WriteEPCByTID(String TIDStr,String EPCStr,byte Password[]);		
<b>illustration</b>	Overwrite the EPC number of the label with the TID mask. (Request a TID starting address starting from 0)		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	TIDStr	String	The 16th decimal TID number of the label
	EPCStr	String	16-EPC number to be rewritten
	Password	byte[]	Access password for the tag, 4 bytes
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	None		

### 3.2.8 Lock 【Label lock】

<b>definition</b>	public int Lock(String EPCStr,byte select,byte setprotect,String PasswordStr);		
<b>illustration</b>	Set the protection status of each area of the label		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	EPCStr	String	The 16 decimal EPC number of the label
	select	byte	<p>1 Bytes,</p> <p>0x00 controls the Kill password read / write protection settings.</p> <p>0x01 controls the access password read and write protection settings.</p> <p>0x02 Controls the EPC memory read and write protection settings.</p> <p>0x03 Controls the TID memory read and write protection settings.</p> <p>0x04 Controls the user memory read and write protection settings.</p> <p>Other values are reserved. If the reader receives other values, the message.</p>
	setprotect	byte	<p>1 Bytes,</p> <p>When Select is 0x00 or 0x01, the SetProtect value represents the following meaning:</p> <p>The 0x00 is set to be write-readable</p> <p>0x01 is set to always readable</p> <p>0x02 is set to be readable with a password</p> <p>0x03 is set to never read or write</p> <p>When Select is 0x02,0x03,0x04, the SetProtect value represents the following meaning:</p> <p>The 0x00 is set to be writable</p> <p>0x01 is set to forever writable</p> <p>0x02 is set to write with a password</p>

			0x03 is set to never write it.
	PasswordStr	String	Label's 16-wise string access password
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	None		

### 3.2.9 Kill 【Label destruction】

<b>definition</b>	public int Kill(String EPCStr,String PasswordStr)		
<b>illustration</b>	This command is used to destroy the labels. After the label is destroyed, reader commands are never processed.		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	EPCStr	String	The 16 decimal EPC number of the label
	PasswordStr	String	Label's 16-wise string access password
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

## 3.3 Custom command

### 3.3.1 SetCallBack 【Set the callback interface】

<b>definition</b>	public void SetCallBack(TagCallback callback)		
<b>illustration</b>	Set the callback interface after starting the inventory, and the label data is returned through the callback interface		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	callback	TagCallback	The callback interface for the label
<b>return(void )</b>	none		
<b>identifying code</b>	none		

### 3.3.2 GetUHFInformation 【Obtain the module parameter information】

<b>definition</b>	public int GetUHFInformation(byte Version[],byte Power[],byte band[],byte MaxFre[],byte MinFre[],byte BeepEn[],byte Ant[])		
<b>illustration</b>	Get the basic information about the UHF module.		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	Version	byte []	Output 2 bytes, the reader version information. The first byte is the version number, and the second byte is the subversion number handle.
	Power	byte []	Output 1 byte, the output power of the reader. Range is 0 to 30 in dBm.
	band	byte[]	Output 1 byte, frequency spectrum band.  1 - "Chinese band2";  2 - "US band";  3 - "Korean band";  4 - "EU band";  8 - "Chinese band1";
	MaxFre	byte[]	Output 1 byte, indicating the maximum frequency point of the current reader work.
	MinFre	byte[]	Output 1 byte, indicating the minimum frequency point of the current reader operation.
	BeepEn	byte[]	Output 1 byte, the buzzer call message.
	Ant	byte[]	Output 1 byte, and the antenna configuration information. Each bit bit represents an antenna number
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.3.3 SetRfPower [Set the power]

<b>definition</b>	public int SetRfPower(int Power)		
<b>illustration</b>	Set the reader power		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	Power	int	Output power of the reader. Range is 0 to 33, per unit of dBm.
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.3.4 SetRegion [Set the frequency point]

<b>definition</b>	public int SetRegion(int band,int maxfre,int minfre)		
<b>illustration</b>	Set the working frequency band of the reader		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	band	Int	1 byte, frequency spectrum band.  1 - "Chinese band2";  2 - "US band";  3 - "Korean band";  4 - "EU band";  8 - "Chinese band1";
	maxfre	Int	Represents the maximum frequency point where the current reader is working.
	minfre	Int	Represents the minimum frequency point where the current reader is working.
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		



### 3.3.5 getInventoryTagMapList 【Get the read card tag array】

<b>definition</b>	List<InventoryTagMap> getInventoryTagMapList();
<b>illustration</b>	Get the read card tag array
<b>parameter</b>	none
<b>return(List)</b>	Successfully returned to the label list.
<b>identifying code</b>	none

### 3.3.6 getInventoryTagResultList 【Get the read card tag result array】

<b>definition</b>	List<InventoryTagMap> getInventoryTagResultList();
<b>illustration</b>	Get the read card tag result array
<b>parameter</b>	none
<b>return(List)</b>	Successfully returned the label result list.
<b>identifying code</b>	none

### 3.3.7 SetPowerMode 【Set power consumption mode】

<b>definition</b>	int SetPowerMode(int OnOff);
<b>illustration</b>	none

parameter	name	type	remarks
	OnOff	int	Set the switch for the power consumption mode  Open set to 1  The closure is set to 0
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.3.8 MeasureTemperature 【measure the temperature】

<b>definition</b>	int MeasureTemperature(byte[] Temp);		
<b>illustration</b>	measure the temperature		
parameter	name	type	remarks
	Temp	byte[]	An array of the incoming byte type
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	None		

### 3.3.9 MeasureReturnLoss 【Measure back damage】

<b>definition</b>	int MeasureReturnLoss(byte[] ReturnLoss);		
<b>illustration</b>	Measure back damage		
parameter	name	type	remarks

	ReturnLoss	byte[]	Incoming in the byte array type data
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.3.10 SetAntenna 【Set the antenna number】

<b>definition</b>	int SetAntenna(byte AntCfg);		
<b>illustration</b>			
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	AntCfg	byte	Incoming the byte type data
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.3.11 SetWorkMode 【Set the working mode】

<b>definition</b>	int SetWorkMode(byte ReadMode);		
<b>illustration</b>	Set the working mode		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	ReadMode	byte	Incoming the byte type data
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.3.12 SetBaudRate 【Set the Porter rate】

<b>definition</b>	int SetBaudRate(byte BaudTate);		
<b>illustration</b>	Set the Porter rate		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	BaudTate	byte	Incoming the byte type data
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.3.13 GetModuleVersion 【Get the module version】

<b>definition</b>	int GetModuleVersion();		
<b>illustration</b>	Get the module version		
<b>parameter</b>	none		
<b>return(int)</b>	Successfully returned 0.		
<b>identifying code</b>	none		

### 3.3.14 beginSound 【Turn on the sound】

<b>definition</b>	Void beginSound (boolean sound);
-------------------	----------------------------------

<b>illustration</b>	Turn on the sound		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	sound	boolean	true:Turn on the sound
<b>return(int)</b>	none		
<b>identifying code</b>	none		

### 3.3.15 setSoundId **【set sound】**

<b>definition</b>	void setSoundId(int id, SoundPool soundPool);		
<b>illustration</b>	Turn on the sound		
<b>parameter</b>	<b>name</b>	<b>type</b>	<b>remarks</b>
	Id	int	sound id
	soundPool	SoundPool	Audio resources
<b>return(int)</b>	none		
<b>identifying code</b>	none		

### 3.3.16 playSound **【play Sound】**

<b>definition</b>	void playSound();		
<b>illustration</b>	Turn on the sound		

<b>parameter</b>	none
<b>return(void )</b>	none
<b>identifying code</b>	none

Calculation formula for each frequency band:

Chinese band2:  $F_s = 920.125 + N * 0.25$  (MHz) in  $N \in [0, 19]$ .  
US band:  $F_s = 902.75 + N * 0.5$  (MHz) in  $N \in [0, 49]$ .  
Korean band:  $F_s = 917.1 + N * 0.2$  (MHz) in  $N \in [0, 31]$ .  
EU band:  $F_s = 865.1 + N * 0.2$  (MHz) in  $N \in [0, 14]$ .  
Chinese band1:  $F_s = 840.125 + N * 0.25$  (MHz) in  $N \in [0, 19]$ .

## 4 Appendix 1

<b>error code</b>	<b>describe</b>
0x00	<b>Execution success.</b>
0x01	<b>No electronic tag was found.</b>
0x05	<b>Access password error.</b>
0x09	<b>Destroy the password error.</b>
0x0A	<b>The destruction password cannot be all 0.</b>
0x0B	<b>The electronic label does not support this command.</b>
0x0C	<b>The access password cannot be 0 for this command.</b>

0x0D	<b>The electronic tag has been set for read protection and cannot be set again.</b>
0x0E	<b>The electronic tag is not set for read protection and does not need to unlock.</b>
0x10	<b>The byte space is locked, and the write fails.</b>
0x11	<b>Can't lock.</b>
0x12	<b>Is already locked, and it cannot be locked again.</b>
0x13	<b>The parameter saving failed, but the set value is valid before the read and write module is powered off.</b>
0x14	<b>Unable to adjust.</b>
0xF8	<b>Antenna detection error</b>
0xF9	<b>Command execution error.</b>
0xFA	<b>There are electronic tags, but the communication is not smooth, can not operate.</b>
0xFB	<b>No electronic label is operable.</b>
0xFC	<b>The electronic tag returns the error code.</b>
0xFD	<b>Incorrect command length.</b>
0xFE	<b>Unlawful order.</b>
0xFF	<b>parameter error.</b>
0x30	<b>Communication error.</b>