ModuleAPI_Android_BT API

MODEL: MUBR01

Developers manual

Version1.7

Contents

Release notes

Date	Version	Description
2011-3-7	Version 1.0	The original version supports the main reader type of the core module, supporting the basic operation of the reader and performance parameter settings.
2011-5-5	Version 1.1	* Add self-developed module reader support * Complete more reader parameter settings
2012-10-15	Version 1.2	* Type modification M1S changed to MT series
2013-2-19	Version 1.3	* Add an enumeration, add an example
2015-5.28	Version 1.4	* Unified interface 2.0, 4.0BLE support
2016-9-30	Version 1.5	* Library function internal optimization adjustment, SetFrameParams changes
2017-5-3	Version 1.5	* add the ResetBlueTooth method, and pay attention to the Connect function.
2019-8-30	Version 1.6	*add quick mode *amend sdk
2019-11-15	Version1.7	* Optimize internal exception handling * Add permanent save interface

1 Introduction

Android is a very popular platform for smart devices, and its free and open resources make it increasingly useful in the industry. Using Java language as the application layer development on Android makes the development function richer and faster, especially on the network. An API developed to help customers of the Android platform application environment to quickly and easily develop the secondary application of the reader application. The API supports almost all of the SL module Bluetooth reader products. The API is given in the form of a development package called ModuleAPI_Android_BT.jar, because there is a version difference, so be sure to pay attention to the version number when using the development package.

2 Reference development kit

After the user creates a new Android project, add an external library on its property page. Add ModuleAPI_Android_BT.jar to the project. After adding, you need to import the package name in the reference class:

Import reader class

import com.silionmodule.*;

import Bluetooth communication class

import com.bth.api.cls.

All user interface classes are defined in these two packages, and the user only needs to know the classes and methods in the two packages.

Bluetooth reader, Bluetooth Class (2.0) or 4.0 BLE communication, use com.bth.api.cls to complete Bluetooth communication, and then initialize the reader object.

3 Interface class and method

3.1 Comm_Bluetooth class

Bluetooth communication general class. Integrated Bluetooth Class (2.0) and Bluetooth 4.0 BLE protocol communication methods provide a unified communication interface.

3.1.1 Construction method

Comm_Bluetooth(Context act)

Description: Construct Bluetooth communication class

Define: public Comm_Bluetooth(Context act)

Parameter Description

name	Type	Description
act	Context	Pass in the Context object to accept the event

Return: No

3.1.2 ResetBlueTooth Function

Description: Reset Bluetooth, reset Bluetooth if there is a search failure or an abnormal connection

definition: public void ResetBlueTooth()

It is best to search before connecting a Bluetooth reader, if the Bluetooth is connected. It is not searchable. In addition, the internal will make some judgments if the Bluetooth is abnormally reset for about 5 seconds. Of course, if the search connection has an exception or fails multiple times, you can also call the reset function ResetBlueTooth.

Parameter Description: No

Return: No

3.1.3 Bluetooth search Function

3.1.3.1 StartSearch Function

Description: Start the search function, after found the device, it will enter the callback function processing.

definition: public int StartSearch(int searchoption,SearchCallback Scallback)

Parameter Description

Name	Type	Description
searchoption	int	Search options
Scallback	SearchCallback	Callback function after searching for device

Returns: 0 for success, other values for failure

example:

```
int selectoption=0;
    selectoption|=Comm_Bluetooth.SearchOption.Blue.value();
    search 2.0/3.0 set
    selectoption|=Comm_Bluetooth.SearchOption.BLE.value();
    search 4.0（BLE）set


 callback:
private Comm_Bluetooth.SearchCallback Cbscallback=new
           Comm_Bluetooth.SearchCallback()
    {
        @Override
        public  void OnSearch(CommBlueDev device) {
      }
  }
```

3.1.3.2 CommBlueDev Class

Description: Bluetooth device class

definition: public CommBlueDev(BluetoothDevice bd,int ri)

Member function description

Name	Type	Description
getName()	String	Get a Bluetooth device Name
getAddress()	String	Get the address of the Bluetooth device

getType()	int	Get the Bluetooth device type, view it in detail in Android SDK document
RssiValue()	int	Get the Bluetooth device signal strength

3.1.4 StopSearch Function

Description: Stop searching for Bluetooth devices

definition: public int StopSearch()

Parameter Description: No

Return: No

3.1.5 ToMatch Function

Description: Match Bluetooth Function

definition: public int ToMatch(String adrorpwd)

Parameter Description

Name	Type	Description
adrorpwd	String	Note that if Bluetooth 4.0 is a password, and uuid is set before use. If it is Bluetooth 2, it is the address.

Return：value is BluetoothDevice.BOND_BONDED Then the match is successful.

3.1.6 CancleMatch Function

Description: Unpair Bluetooth, note only for Bluetooth 2.0

definition: public boolean CancleMatch(String address)

Parameter Description

Name	Type	Description
address	String	Bluetooth mac address

Return: true ,Indicates successful cancellation

3.1.7 Connect Function

Description: Create a new connection, connect to the target Bluetooth device, cancel or disconnect the connection needs to be called

DisConnect Function

Note: Do not put the Connect method in the search callback function of Bluetooth, because the Bluetooth communication class is in the main thread, the search and its callback function is another thread. Once the Bluetooth device is searched, it should be handled by the main thread through a mechanism such as event wakeup. Usually handled with the Hander class.

definition: public int Connect(String address,int bluetype)

Parameter Description

Name	Type	Description
address	String	Bluetooth mac address
bluetype	Int	2 is Class Bluetooth 2.0,4 is 4.0 BLE

Return: 0 means success, non-zero means failure

3.1.8 ConnectState Function

Description: Get Bluetooth connection status

definition: public int ConnectState()

Parameter Description: No

Return：

Comm_Bluetooth.DISCONNECTED disconnect

Comm_Bluetooth.CONNECTED connected

Comm_Bluetooth.CONNECTING connecting

3.1.9 GetConnectAddr Function

Description: Get the connected Bluetooth mac address

definition: public String GetConnectAddr()

Parameter Description: No

Returns: mac address string

3.1.10 getRemoveType Function

Description: Get the connected Bluetooth type

definition: public String GetConnectAddr()

Parameter Description: No

Back: 2 means Bluetooth Class 2.0, 4 means Bluetooth 4.0 BLE

3.1.11 FindServices Function

Description: Search for Bluetooth BLE service, only for Bluetooth 4.0 BLE

definition: public List<BLEServices> FindServices(int millsecond)

Parameter Description

Name	Type	Description
millsecond	int	Search time generally takes 1000~6000 milliseconds

Return：

List<BLEServices> BLE Service list

3.1.12 SetServiceUUIDs Function

Description: Set uuid for BLE service, only for Bluetooth 4.0 BLE

definition: public boolean SetServiceUUIDs(String serviceuuid,String readuuid,String writeuuid)

public boolean SetServiceUUIDs(String serviceuuid,String readuuid,String writeuuid,String passuuid)

Parameter Description

Name	Type	Description
serviceuuid	String	Servie uuid

Readuuid	String	Data notify characteristic uuid
Writeuuid	String	Write data characteristic uuid
Passuuid	String	Password Characteristic uuid

Return: true is success false is failure

3.1.13 GetUUID Function

Description: Get the UUID object of Bluetooth 4.0 BLE

definition: public UUID GetUUID(String uuid)

Parameter Description:

Name	Type	Description
Uuid	String	Uuid string

Return: UUID object

3.1.14 IssetUUID Function

Description: Is it set UUID

definition: public boolean IssetUUID()

Parameter Description: No

Return: True indicates that it has been set, false means that it has not been set

3.1.15 ReConnect Function

Description: Reconnecting the Bluetooth reader, the connection must have been established before

definition: public int ReConnect()

Parameter Description: No

Return:

3.1.16 ResetUUID Function

Description: Reset UUID, clear all configured UUIDs

definition: public void ResetUUID()

Parameter Description: No

Return:

3.1.17 SetFrameParams Function

Description: Set the send data one frame size and timeout, only for Bluetooth 4.0 BLE.

Define:

Parameter Description

Name	Type	Description
fcount	int	One frame data size, default 20
ftime	Int	Timeout, default 100 milliseconds

Return：No

3.1.18 Comm_SetParam Function

Description: Set communication parameters

definition: public boolean Comm_SetParam(Object obj,String key)

Parameter Description

Name	Type	Description
Obj	Object	Parameter value
key	String	Parameter Name

Parameters List

Name	Value	Description
ParamNames.Communicaton_module	HardWareDetector.Module_Type Enumeration type	Specify communication module type
ParamNames.Communicaton_mode	0 or 1	Working mode 0 is passive mode (command mode) 1 is active mode (trigger mode) 2 is update firmware

Return: True means success

note: ParamNames.Communicaton_mode Will affect the function used by the Reader class

3.1.19 Comm_GetParam Function

Description: Get communication parameter values

definition: public Object Comm_GetParam(String key)

Parameter Description

Name	Type	Description
key	String	Parameter Name

Parameter list: Same as above

Return: Object

3.1.20 DisConnect Function

Description: Disconnect or cancel the Bluetooth connection.

definition: public int DisConnect()

Parameter Description: No

Return: No

3.2 Bluetooth connection step instructions:

For Bluetooth Class 2.0 and Bluetooth 4.0 BLE, the connection steps are different due to protocol differences.

3.2.1 Bluetooth 2.0 connection steps:

1) Initialize Bluetooth

2) Search for Bluetooth

3) Get a Bluetooth device

4) Connect a Bluetooth device

5) Determine if you want password matching--→ enter password

6) connection succeeded

3.2.1 Bluetooth 4.0 BLE connection steps:

1) Initialize Bluetooth

2) Search for Bluetooth

3) Get a Bluetooth device

4) Connect a Bluetooth device

5) Search UUID service

6) If need to enter a password for the password characteristic

7) set write and read UUID characteristic

8) connection succeeded

3.3 Reader Class

The Reader class is a high-level abstract class that encapsulates all the different types of readers. Therefore, the user only needs to use the Reader class to complete, construct, set, operate, close, and so on. The reader life cycle is from the time the Reader class is created until the reader is closed.

3.3.1 Create Function：

Description: Create a reader method 1: All reader applications start with the creation of a reader, instantiating a Reader object with the Create method. Once created, you can perform various operations on the reader.

Define: public static Reader Create(AntTypeE antsnum,Communication comm)

Parameter Description

Name	Type	Description
type	ReaderTypeE	Reference enumeration constant type
comm	Communication	Communication object, for example: Comm_Bluetooth object

Return：Returns a Reader instance that is the object of the specific reader.

Note: Creating a reader function may throw a ReaderException, if you want to handle the exception yourself;

If an exception occurs while creating the reader, an exception should be caught. Common reasons are address errors, physical connection problems, and so on.

example:

int antc=1;

```
Comm_Bluetooth CommBth=new Comm_Bluetooth(this);
......Eliminate the CommBth connection process here
Reader reader=Reader.Create(AntTypeE.valueOf(antc), CommBth);
```

3.3.2 DisConnect Function：

Description: Close the reader method: If you do not need to use the reader, calling this method will release the reader resource.

Define: public abstract void DisConnect();

Parameter Description: No

Return： void

3.3.3 addStatusListenerFunction

Description: Monitors the status of the Bluetooth reader and returns the status string.

definition: `public void addStatusListener(StatusEventListener listener)`

Parameter Description:

Name	Type	Description
listener	StatusEventListener	Listener object

Return: No

example:

```
StatusEventListener SL=new StatusEventListener()
    {
        @Override
        public void StatusCatch(Object t) {
      System.out.println(t.toString());}
    }
```

3.3.4 Passive mode function description

Bluetooth reader works in active mode and passive mode. Setting different working modes in communication parameters will use different functions. Passive mode means that the Bluetooth reader is in standby after power-on, and needs to connect to the host to send commands to it to work. The following is a description of all functions in passive mode.

3.3.4.1 InitTag Function：

Description: Initialize the Gen2 tag method: Gen2 tag initialization refers to initializing the EPC area. This way of writing data is the same as writing the function WriteTagMemWords to the EPC bank, and there are different places. Both methods must specify an operating antenna. Therefore, the operating antenna must have been set before using this method. InitTag is the length of the id that can change the tags epc id length (the length of the tags at the time of inventory), while WriteTagMemWords only changes the data without changing the length. This operation must first set the operating antenna.

Define:public void InitTag(TagFilter target, TagData epc) throws ReaderException

Parameter Description

Name	Type	Description
Target	TagFilter	Filter condition, type reference structure data type
Epc	TagData	Reference structure data type

Return：void

example:

```
reader.InitTag(null, new TagData("F3888333"));
```

3.3.4.2 KillTag Function：

Description: Destroy the tag method: The so-called destruction of the tag is to invalidate the tag. Once a tag is destroyed, the tag can no longer be used. If you want to destroy a tag, you must first set the kill password to non-zero, and the kill password exists in the reserved block 0~1. This operation must first set the operating antenna.

Define: public void KillTag(TagFilter target, int password) throws ReaderException

Parameter Description:

Name	Type	Description
Target	TagFilter	Filter condition, type reference structure data type
Password	Int	Password 8 hexadecimal characters, 32 bits

Return：void

3.3.4.3 LockTag Function：

Description: Lock tag method: Lock the tag, which depends on the bank of the tag. The Gen2 tag lock is unlocked (temporary lock), temporarily locked, and permanently locked. Different banks have different performances after being locked by different types.

Temporary lock, that is, lock a bank, but this lock can be released. For the reserved area, no matter what kind of lock, it is inaccessible and can't be changed. For epc, user is readable and unmodifiable. The correct password must be provided for modification. Or unlock this lock bank, operation again. The Tid bank is generally permanently locked from the factory.

Permanently locked, doing permanent operations on a zone, once permanently locked, cannot be lifted. The correct password must be provided to access or modify. This operation must first set the operating antenna.

Define: public void LockTag(TagFilter target, TagLockAction action) throws ReaderException

Parameter Description

Name	Type	Description
Target	TagFilter	Filter condition, type reference structure data type
Action	TagLockAction	Lock type, reference structure data type

Return：void

example:

```
Gen2LockAction g2la = new Gen2LockAction(Gen2.Gen2LockAction.EPC_LOCK);
        Gen2Password g2pw = new Gen2Password("11111111");
        reader.paramSet(ParamNames.Reader_Gen2_AccessPassword, g2pw);
        reader.LockTag(null, g2la);
```

3.3.4.4 paramGet Function：

Description: Get a parameter value method: view the value of a parameter of the reader

Define: public Object paramGet(String key) throws ReaderException

Parameter Description

Name	Type	Description
Key	String	Configuration parameter Name, reference enumeration constant type

Return：Ojbect The function returns an abstract object, and the user can convert according to the specific return type of the configuration parameter table.

example:
```
Gen2.SessionE g2se = (SessionE) reader
        .paramGet(ParamNames.Reader_Gen2_Session);
```

3.3.4.5 paramSet Function：

Description: Set a parameter value method: set the value of a parameter of the reader

Define: public void paramSet(String key,Object value) throws ReaderException

Parameter Description

Name	Type	Description
Key	String	Configuration parameter Name, reference enumeration constant type
Value	Object	A parameter value, corresponding to the specific type reference configuration parameter table

Return：void

example:
```
reader.paramSet(ParamNames.Reader_Gen2_Session,Gen2.SessionE.Session1
);
g2se = (SessionE) reader.paramGet(ParamNames.Reader_Gen2_Session);
```

3.3.4.6 Read Function：

Description: Inventory tags method: The so-called inventory tags refers to tags that can be identified by one or more antennas in a certain query order. Corresponding to the 6b tag is generally the first 8 blocks of data, and the Gen2 tag is the pc length data in the Epc bank. Therefore, before calling this method, you need to configure the ReadPlan parameter. Use ReadPlan to command the reader to do the inventory tag.

Define: public TagReadData[] Read(int milliseconds) throws ReaderException

Parameter Description

Name	Type	Description
Milliseconds	Int	Counting timeout period, the time after the reader reads from reading to returning

Return：TagReadData Array, each element is the data of the tag to the tag. Detailed reference structure data type

example:

```
SimpleReadPlan srp= new SimpleReadPlan(new int[]{1,2,3});
reader.paramSet(ParamNames.Reader_Read_Plan,srp);
TagReadData[] trd = reader.Read(600);
```

3.3.4.7 ReadTagMemWords Function：

Description: Single antenna operation read Gen2 tag memory method: read the memory block of Gen2 tag, a 16-bit, only one bank at a time. This operation must first set the operating antenna.

Define: public short[] ReadTagMemWords(TagFilter target, MemBankE bank, int wordAddress, int wordCount) throws ReaderException

Parameter Description

Name	Type	Description
Target	TagFilter	Filter condition, reference structure data type
Bank	MemBankE	Gen2 tag bank reference enumeration constant type
Wordaddress	Int	Starting block number
Wordcount	Int	Blocks count

Return：short[]

example:

```
short[] epddata = reader.ReadTagMemWords(null, MemBankE.EPC, 2, 6);
```

3.3.4.8 WriteTagMemWords Function：

Description: Single antenna operation write Gen2 tag memory method: reserved bank, user bank starts from 0 block, Epc area starts from the second block, Tid can't write. The operation must first set the operation antenna.

Define: public void WriteTagMemWords(TagFilter target,MemBankE bank, int wordaddress, short[] data) throws ReaderException

Parameter Description

Name	Type	Description
Target	TagFilter	Filter condition, reference structure data type
Bank	MemBankE	Gen2 tag bank reference enumeration constant type
Wordaddress	Int	Starting block number
Data	Short	Hexadecimal data

Return：void

example:

```
reader.WriteTagMemWords(null, MemBankE.USER, 0, Functional
.hexstr_Short16s("094E2D008C4E2D00004E04C882D02C7B330101FFF"));
```

3.3.4.9 ReadTagMemWords Function：

Description: Single antenna operation read Gen2 tag memory method: read the memory block of Gen2 tag, a 16-bit, only one bank at a time. This operation must first set the operating antenna.

Define:public short[] ReadTagMemWords(TagFilter target, MemBankE bank, int wordAddress, int wordCount) throws ReaderException

Parameter Description

Name	Type	Description
Target	TagFilter	Filter condition, reference structure data type
Bank	MemBankE	Gen2 tag area, reference enum constant type

Wordaddress	Int	Starting word address
Wordcount	Int	Word count

return：short[]

example：

```
short[] epddata = reader.ReadTagMemWords(null, MemBankE.EPC, 2, 6);
```

3.3.4.10 WriteTagMemBytesFunction：

Description: Single antenna operation to write ISO180006B tag memory method: can only be written from the 8th block, 0~7 block fixed. This operation must first set the operating antenna

Define:public void WriteTagMemBytes(TagFilter target,ISO180006B.MemBankE bank, int wordaddress, byte[] data)throws ReaderException

Parameter Description

Name	Type	Description
Target	TagFilter	Filter condition, reference structure data type
Bank	Iso180006B.MemBankE	bank, reference enum constant type
Bygeaddress	Int	Starting block
Data	Byte[]	Hexadecimal data

return：void

```
reader.WriteTagMemBytes(iso6tf, ISO180006B.MemBankE.ISO180006B, 8,
Functional.hexstr_Bytes("AAAAAA"));
```

3.3.4.11 WriteTagMemWordsFunction：

Description: Single antenna operation Write Gen2 tag memory method: reserved bank, user bank starts from 0 block, Epc bank starts from second block, Tid can't write normally. This operation must first set the operating antenna.

Define:public void WriteTagMemWords(TagFilter target,MemBankE bank, int wordaddress, short[] data) throws ReaderException

Parameter Description

Name	Type	Description
Target	TagFilter	Filter condition, reference structure data type
Bank	MemBankE	Gen2 bank，reference enum constant type
Wordaddress	Int	Starting word address
Data	Short	Hexadecimal data

return：void

example：

```
reader.WriteTagMemWords(null, MemBankE.USER, 0, Functional
.hexstr_Short16s("094E2D008C4E2D00004E04C882D02C7B330101FFF"));
```

The following is a reader support function based on the Arm7 control board.

3.3.4.12 SetIP_arm7b

Define:void SetIP_arm7b(String ip,String subnet,String gateway)throws ReaderException;

Set the IP address method: set the ip address of the Arm7 board.

Parameter Description

Name	Type	Description
Ip	String	Ip address
Subnet	String	Subnet mask
Gateway	String	Gateway

return：void

example：

```
reader.SetIP_arm7b("192.168.1.188", "255.255.255.0", "192.168.1.1");
```

3.3.4.13 GetIP_arm7bFunction

Description: Read the IP address method: if you read the string array that normally returns three elements, in turn：ip ， Subnet mask，Gateway。

Define:String[] GetIP_arm7b() throws ReaderException;

Parameter Description:

example:

```
String[] nets = reader.GetIP_arm7b();
```

3.3.4.14 GpioGet_arm7bFunction

Description: Read extended GPIO method: similar to GpioGet method, the pin is on the arm7 board

Define: GPioPin[] GpioGet_arm7b() throws ReaderException;

Parameter Description:

Return:

3.3.4.15 GpioSet_arm7bFunction

Description: Set the extended GPIO method: support 4 pin numbers 1, 2, 3, 4, similar to the GpioSet method

Define: boolean GpioSet_arm7b(GPioPin[] gp) throws ReaderException;

Parameter Description:

Return:

3.3.4.16 ResetModule_arm7b Function

Description: Restart the reader method

Define: void ResetModule_arm7b() throws ReaderException;

Parameter Description: none

Return:

3.3.5 Active mode function description

Bluetooth reader works in active mode and passive mode. Setting different working modes in communication parameters will use different functions. Active mode means that after the Bluetooth reader is powered on, it will be saved in the reader. The internal parameters are initialized and in standby mode. Press the trigger button to scan the label and transfer the data to the host. The following is the description of all functions in active mode.

3.3.5.1 Active inventory receiving data

Description: Listen to the data returned by the Bluetooth reader and

Return `TagReadData[] array`

Define: `public void addDataListener(DataListener listener)`

Parameter Description:

Name	Type	Description
listener	DataListener	Data listener object

return：void

example：

```
DataListener DL=new DataListener()
    {
        @Override
        public void ReadData(TagReadData[] t) {
            // TODO Auto-generated method stub
}
}
```

3.3.5.2 Active mode permanently saves parameter settings

In active mode, the parameters can also be configured before pressing the keyboard point label. The parameters are permanently saved. Once the button is pressed, the Bluetooth reader will not respond to any command. It must be powered back on to respond to the command.

Configure and get the parameters, still use the paramGet and paramSet functions. But the parameter names are prefix with

`ParamNames.InitMode_`

3.3.6 Quick mode function description

In the passive mode of operation, the UHF module of the R2000 chip supports the fast inventory mode. Once the quick inventory mode is turned on, it will not be counted until the callback status is abnormal or a stop command is received. Tag data listens for tag events through the DataListener listener.

3.3.6.1 Start fast mode

Description: Start a quick inventroy

Define: `public int AsyncStartReading(int option)`

Parameter Description:

Name	Type	Description
option	int	Quick inventory option value

Returns: 0, the exception returns an error code

The value of Option can be defined by the class QuickModeOption, which is obtained by configuring the object property and then calling the method getcmdotpion(). By default, you can pass 0.

3.3.6.2 Stop quick mode Function

Description: stop quick mode inventory

Define: `public int AsyncStopReading()`

Returns: 0, the exception returns an error code

3.3.7 Permanently save parameter description

UHF The module supports permanent saving of parameters, that is, power-on will configure default values as before. The permanent save parameter configuration is consistent with the normal parameter configuration using functions. Set the call to paramSet and get the call to paramGet. The parameter key value is *Reader_Perma_* prefix.

3.3.7.1 Configure permanent save

Call the paramSet function to complete, note that if you pass in a null value, it will be set to the default value.

define：`paramSet(ParamNames.Reader_Perma_Radio_PortPowerList, value);`

example1：Permanently save antenna power

Value for the AntPower[] array

```
AntPower[] ap=new AntPower[1];
ap[0]=new AntPower(1,2460,2480);
 reader.paramSet(ParamNames.Reader_Perma_Radio_PortPowerList, ap);
```

example2: Permanently save frequency and frequency switching time

```
int[] fre2 = (int[])reader
                    .paramGet(ParamNames.Reader_Region_HopTable);

 int[] fre3=new int[fre2.length-1];
         for(int i=0;i<fre3.length;i++)
             fre3[i]=fre2[i];
       reader.paramSet(ParamNames.Reader_Perma_Region_HopTime,fre3);
```

frequency switching time `reader.paramSet(ParamNames.Reader_Perma_Region_HopTime,200);`

example3：set region，session，target

```
com.silionmodule.Region.RegionE re = com.silionmodule.Region.RegionE.NA;
reader.paramSet(ParamNames.Reader_Perma_Region_Id, re);

reader.paramSet(ParamNames.Reader_Perma_Gen2_Session,
com.silionmodule.Gen2.SessionE.Session1);

reader.paramSet(ParamNames.Reader_Perma_Gen2_Target,
                com.silionmodule.Gen2.TargetE.B);
```

example4：whether to check antenna

```
reader.paramSet(ParamNames.Reader_Perma_Antenna_CheckPort,null);
```

Set the inspection antenna, then if the module detects that the antenna is not recognized before reading, it will report an error. If the antenna is known to be undetectable, the check item should be cancelled.

example5：set Qvalue

```
reader.paramSet(ParamNames.Reader_Perma_Gen2_Q,null);
```

Set the Q value, note that -1 indicates the dynamic Q value.

3.3.7.2 Get permanent save configuration

Call the paramGet function to complete.

example1： Obtain antenna power and antenna power for permanently saved configurations

```
AntPower[] ap = (AntPower[])
 reader.paramGet(ParamNames.Reader_Perma_Radio_PortPowerList);
AntPower[] ap2 = (AntPower[])reader
                        .paramGet(ParamNames.Reader_Radio_PortPowerList);
LogD.LOGD("get  save  rpow:"+ap[0].Readpower()+"  wpow:"+ap[0].Writepower()+"
after connect get rpower:"+ap2[0].Readpower()+"wpower:"+ap2[0].Writepower());
```

example2：Get permanent save configuration target and Q value

```
com.silionmodule.Gen2.TargetE tar1=(TargetE)
reader.paramGet(ParamNames.Reader_Perma_Gen2_Target);
                com.silionmodule.Gen2.TargetE tar2=(TargetE)
reader.paramGet(ParamNames.Reader_Gen2_Target);
                LogD.LOGD("get save tar:"+tar1+" after connect tar:"+tar2);
Gen2.Q g2q1=(Q) reader.paramGet(ParamNames.Reader_Perma_Gen2_Target);
```

3.3.7.1 Restore power-on defaults

Call paramSet function to complete, used to clear all previous configurations saved permanently。

example：

```
reader.paramSet(ParamNames.Reader_Perma_reset, true);
```

4 Data structure type

4.1 AntPower Class

This class mainly stores the read power and write power of one antenna. Some readers read and write power must be set to be consistent.（MT100）。So there are two ways to construct

Constructor1

public AntPower(int ant,int pow)

Used to construct power parameters with consistent read and write power

Constructor2

public AntPower(int ant,int readpow,int writepow)

Used to construct power parameters with different read and write powers

4.2 Q Class

There are Dynamic Q and static Q in Q class, static Q needs to pass Q value, Q value refers to Gen2 protocol.

4.3 Gen2Password Class

Password used to store the Gen2 protocol tag, either a numeric value or an 8 hex string

Constructor1

public Gen2Password(int value)

Constructor2

public Gen2Password(String value) throws ReaderException

it will throw an exception if the format of passed string is incorrect

4.4 Gen2TagFilter Class

Conditional parameter for filtering of Gen2 tags, inheriting the TagFilter interface

Constructor

public Gen2TagFilter(MemBankE mbe,int filteraddress,byte[] filterdata,int bitdatalength)

Parameter Description

Name	Type	Description
Mbe	MemBankE	Filter bank
Filteraddress	Int	Filter address, unit block
Filterdata	Byte[]	Filter data ,bytes
bitdatalength	int	Filter data length, unit in bit

Filter tags can be accurately placed up to 255 bits

The reader can also support matching and mismatching operations, the default match

public void SetInvert(boolean isinvert)

if the isinvert value is true, then the filter condition is a match, otherwise the filter is reversed.

4.6 GpioPin Class

GPIO The class stores a pin and its level.

Constructor：

public GPioPin(int id,boolean high)

Parameter Description

Name	Type	Description
Id	Int	Gpio pin number
High	Boolean	Ture is high level， false is low level

4.7 ISO180006B. ISO6BTagData Class

Data for storing Iso180006B tags

Constructor1：

public ISO6bTagData(byte[] bEPC) throws ReaderException

Constructor2：

public ISO6bTagData(String sEPC) throws ReaderException

Constructor3：

public ISO6bTagData(byte[] bEPC, byte[] crc) throws ReaderException

Constructor4：

public ISO6bTagData(String sEPC, String sCRC) throws ReaderException

4.8 ISO180006B.ISO6BtagFilter Class

Conditional parameter for filtering 6B tags, inheriting the TagFilter interface

Constructor：

public ISO6BTagFilter(byte[] data)

Filter according to label Id, so data is 8 bytes of data

4.9 ISO18000.LockAction Class

Used to lock 6B tags

Constructor：

public LockAction(int address)

address should between 0 and 255

4.10 SimpleReadPlan Class

Used to specify the reader mode, which can be called a single mode because it supports a protocol

The protocol for specifying the tag of the inventory when inventorying, the antenna used for inventory

Constructor1：

public SimpleReadPlan()

Use default value

Constructor2：

public SimpleReadPlan(int[]ants)

Specified antenna only

Constructor3：

public SimpleReadPlan(int[]ants,TagProtocolE tp)

Specified antenna and protocol

Constructor4：

public SimpleReadPlan(int[]ants,TagProtocolE tp,int weight)

The constructor in the multi-counter mode, the weight specific gravity value, represents the single-dot mode ratio weight.

The specific weight can take any positive integer value, compared with other SimpleReadPlan weights to get the time ratio of the inventory method to the total inventory.

4.11 MultiReadPlan Class

It is used to specify the reader to count the labels according to the multi-dot mode, first allocate the time according to the protocol ratio, and then assign the time according to the antenna.

Constructor1：

public MultiReadPlan(ReadPlan[] plans)

Initialize multi-inventory mode with SimpleReadPlan array

Constructor2：

public MultiReadPlan(ReadPlan[] plans, int weight)

Same as above, and specify the specific gravity value, for example, the Gen2 protocol weight value is 30, and the 6B protocol weight value is 25. If the inventory Read is 1000ms, the Gen2 protocol time is 30/(30+25)=0.55,about 550ms.

example：

```
ReadPlan[] rp = new ReadPlan[2];
        SimpleReadPlan srp1 = new SimpleReadPlan(new int[] { 3 },
                TagProtocolE.Gen2, 30);
        SimpleReadPlan srp2 = new SimpleReadPlan(new int[] { 3 },
                TagProtocolE.ISO18000_6B, 25);
        rp[0] = srp1;
        rp[1] = srp2;
```

```
        MultiReadPlan testMultiReadPlan = new MultiReadPlan(rp);

        reader.paramSet(ParamNames.Reader_Read_Plan,
testMultiReadPlan);
```

4.12 TagData Class

Tag data, generally used for incoming parameters

Constructor1：

public TagData(byte[] pc,byte[] epc,byte[] crc)

Parameter Description

Name	Type	Description
Pc	Byte[]	Pc value
Epc	Byte[]	Epc data
Crc	Byte[]	Crc value

Constructor2：

public TagData(byte[] epcdata,byte[] crc)

epc Byte array data, crc check code

Constructor3：

public TagData(String hexstrdata) throws ReaderException

Hexadecimal string epc data

Constructor4：

public TagData(byte[] epcdata) throws ReaderException

epc byte array data

Constructor5：

public TagData(String hexstrdata,String hexstrcrc) throws ReaderException

Hexadecimal string epc data and hexadecimal crc checksum data

4.13 TagReadData Class

The inventory tag returns an array of tag information, generally without a constructor. Users only need to define the TagReadData array.

Constructor1：

public TagReadData(TagData tdata)

Constructor1：

public TagReadData(byte[] epc,byte[] crc,byte[] pc)

Constructor2：

public TagReadData(byte[] epc,byte[] crc,byte[] pc,int ant,int count,int rssi,int fre,Date time,int readoff)

Constructor3：

public TagReadData(byte[] epc,int ant,int count,int rssi,int fre,long time) throws ReaderException

Property function description

Name	Type	Description
TagData()	TagData	Tag TagData Structural data
EPCHexstr()	Byte[]	Tag EpcID Hexadecimal string data
EPCbytes()	Byte[]	Tag EpcID byte array
CRC()	Byte[]	Crc byte array
PC()	Byte[]	Pc byte array
Antenna()	Int	Return tag read antenan id
ReadCount()	Int	Return tag read count
RSSI()	Int	Return tag read RSSI value
Frequency()	Int	Return tag read frequency
Time()	Date	Return tag read time
ReadOffTime()	Int	Return tag cost time
toString()	String	Return to general print information

4.14 QuickModeOption Class

Constructs an option object for configuring the quick count mode.

Constructor:

```
public QuickModeOption()
```

Property function description

Name	Type	Description
tmf	TagMetaFlags	Returns the value of the tag attribute item, the setting to be returned is true, default all are false
isHeartbeat	boolean	Whether to return heart beat
stoppercent	int	Pause ratio, default 0
ishandlmodefor1200	boolean	For 1200 module, which is special mode in handset
getcmdotpion()	Int	Return a int value

```
example:
QuickModeOption qmioption=new QuickModeOption();
qmioption.tmf.IsReadCnt = true;
qmioption.tmf.IsAntennaID = false;
qmioption.stoppercent= 0;
qmioption.isHeartbeat=true;
int option= qmioption.getcmdotpion();
```

4.15 TagMetaFlags Class

Construct an object that returns a label attribute item flag

Property function description

Name	Type	Description
IsAntennaID	boolean	Whether to return antenna id of tag info
IsReadCnt	boolean	Whether to return read count of tag info

IsRSSI	**boolean**	Whether to return rssi of tag info
IsFrequency	**boolean**	Whether to return frequency of tag info
IsTimestamp	**boolean**	Whether to return time stamp of tag info
IsRFU	**boolean**	reserver field always as false
IsPro	**boolean**	Whether to return protocol of tag info
IsEmdData	**boolean**	Whether to return addition data of tag info read the bank data when inventorying,you must set parameter of ReadPlan with TagOP (only base on R2000 chip series reader supported)

5 Enumeration constant type

5.1 Gen2.MemBankE

Gen2 bank enumeration of protocol tag

Name	Description
RESERVED	Reserved bank, for storing passwords
EPC	Epc bank，thing code,the bank of the inventory is generally 96
TID	tag itself id, generally fixed, not modifiable
USER	User bank

5.2 Gen2.SessionE

Gen2 Session mode of the protocol tag

Name	Description
Session0	Gen2 Session0，Less applied to tags
Session1	Gen2 Session1，More applied to tags
Session2	Gen2 Session2，For a long period of time, when the tag is not powered, the tag is only counted once.
Session3	Gen2 Session3，The effect is the same as above, can be used together with Session2

5.3 Gen2.TargetE

Gen2 Protocol tag target

Name	Description
A	Target with A status tag
B	Target with B status tag
AB	Count the A status tag first, and then count the B status tag until it is not found.
BA	Count the B status tag first, and then count the A

	status tag until it is not found.

5.4 Gen2.WriteModeE

Gen2 Protocol label write mode

Name	Description
WORD_ONLY	Word write
BLOCK_ONLY	Block write

5.5 Region.RegionE

Regional frequency band

Name	Description
NA	US band 902MHz-927MHz
China,	Chinese band 920MHz-924MHz
Europe	European band 865 MHz-867MHz
Korea	Korean band 910MHz-913MHz

5.6 TagProtocol.TagProtocolE

Tag protocol

Name	Description
Gen2	Gen2 protocol
ISO18000_6B,	ISO180006B

5.7 Gen2.Gen2LockAction Enumeration constant

This type of constant is used during the tag lock operation to identify which bank and which lock operation is performed on the tag.

Lock type：

member	description

ACCESS_LOCK	Temporarily lock access password
ACCESS_PERMALOCK	Permanently lock access password
ACCESS_PERMAUNLOCK	Permanently unlock access password
ACCESS_UNLOCK	Unlock access password
EPC_LOCK	Temporarily lock the EPC bank
EPC_PERMALOCK	Permanently lock EPC bank
EPC_PERMAUNLOCK	Permanently unlock EPC bank
EPC_UNLOCK	Unlock EPC bank
KILL_LOCK	Temporarily lock kill password
KILL_PERMALOCK	Permanently lock kill password
KILL_PERMAUNLOCK	Permanently unlock kill password
KILL_UNLOCK	Unlock kill password
TID_LOCK	Temporarily lock TID bank （This area is generally permanently shipped from the factory. ）
TID_PERMALOCK	Permanently lock TID bank
TID_PERMAUNLOCK	Permanently unlock TID bank
TID_UNLOCK	Unlock TID bank
USER_LOCK	Temporarily lock USER bank
USER_PERMALOCK	Permanently lock USER bank
USER_PERMAUNLOCK	Permanently unlock USER bank
USER_UNLOCK	Unlock USER bank

5.8 ParamNames Enumeration constant

Parameters Name for reader configuration

Reader parameter table

Parameter Name	Parameter Type	Description	Readabl

			e and writable
Reader_CommandTimeout	int	Operation timeout	R&W
Reader_TransportTimeout	int	Communication timeout	R&W
Reader_Antenna_CheckPort	Boolean	Check the antenna connection before transmitting power	R&W
Reader_Antenna_ConnectedPortList	int[]	Connected antennas	Read only
Reader_Antenna_PortList	int[]	Number of device antenna ports	Read only
Reader_Gen2_AccessPassword	Gen2.Password	Access password	R&W
Reader_Gen2_WriteMode	Gen2.WriteModeE	Write mode	R&W
Reader_Gen2_Q	Gen2.Q	Q value	R&W
Reader_Gen2_TagEncoding	Gen2.TagEncodingE	Gen2 code	R&W
Reader_Gen2_Session	Gen2.SessionE	Session mode	R&W
Reader_Gen2_Target	Gen2.TargetE	Target mode	R&W
Reader_Gen2_Tari	Gen2.TariE	Gen2 Tari	R&W
Reader_Gpio_InputList	GPioPin[]	GPI input list	Read only

Reader_Gpio_OutputList	GPioPin[]	GPO out list，	Write only
Reader_Radio_PortReadPowerList	int[][]	Port read power list	R&W
Reader_Radio_PortWritePowerList	int[][]	Port write power list	R&W
Reader_Radio_PortPowerList	AntPower[]	Port read and write power list	R&W
Reader_Radio_Temperature	Int	Module temperature	Read only
Reader_Radio_ReadPower	Int	Read power (not recommend ed)	R&W
Reader_Radio_WritePower	Int	Write power (not recommend ed)	R&W
Reader_Radio_PowerMax	Int	Max power	Read only
Reader_Radio_PowerMin	Int	Min power	Read only
Reader_Read_Plan	ReadPlan	Inventory tags method	R&W
Reader_Read_Filter	TagFilter	Filter condition	R&W
Reader_Region_HopTable	Int[]	Regional	R&W

		frequency table	
Reader_Region_Id	RegionE	Area	R&W
Reader_Tagop_Antenna	int	Operating a single antenna	R&W
Reader_Tagop_Protocol	TagProtocol	Operational single protocol	R&W
Reader_Version_SupportedProtocols	TagProtocol[]	Supported protocols	Read only
Reader_Version_Hardware	String	hardware information	Read only
Reader_Version_Model	String	Module version number	Read only
Reader_Version_Serial	String	serial number	Read only
Reader_Version_Software	String	Software information	Read only
InitMode_APP_SIGN		Unused	
InitMode_WORK_MODE	Set String Get Integer	Specify active inventory mode and inventory return label information options	R&W

InitMode_TIMEOUT	SetString Get Integer	inventory timeout, in milliseconds	R&W
InitMode_SELECT_OPTION	SetString Get Integer	Filtering options Calculated value method: Match|bank ,match is 0x08,not match is 0x00,bank is 1,2,3, as epc,tid,user bank	R&W
InitMode_SELECT_ADDRESS	SetString Get Integer	Filter address	R&W
InitMode_SELECT_LENGTH	SetString Get Integer	Filter data length	R&W
InitMode_SELECT_FILTER_DATA	String	Filter data	R&W
InitMode_RFU	String	Unused	
InitMode_POWER	SetString Get Integer	power	R&W
InitMode_REGION	SetString Get Integer	region: Legal value 0x01,0x06,0x08,0x0A	R&W
InitMode_RFU2	String	Unused	

InitMode_FREQUENCY	SetString Get Integer[]	Set with string Each frequency point is separated by a comma.	R&W
InitMode_HIGH_RSSI	SetString Get Integer	Legal value:0,1	R&W
InitMode_SESSION	SetString Get Integer	Legal value:0,1,2,3	R&W
InitMode_TARGET	SetString Get Integer	Legal value:0,1,2,3	R&W
InitMode_SENDTIME	SetString Get Integer	Send data time	R&W
InitMode_Q_VALUE	SetString Get Integer	Qvalue:-1~15	R&W
InitMode_READ_BANK	SetString Get Integer	Embedded read password bank,1 is epc,2 is tid,3 is user bank	R&W
InitMode_READ_ADDRESS	SetString Get Integer	Embedded read address	R&W
InitMode_READ_LENGTH	SetString	Embedded	R&W

	Get Integer	read length	
InitMode_READ_PASSWORD	SetString Get Integer	Embedded read password	R&W
InitMode_READ_MODE		Unused	
InitMode_BLUE_NAME	String	Bluetooth Name is no more than 20 characters	R&W
Reader_Perma_Radio_PortPowerList	AntPower[]	Port read and write power list	R&W
Reader_Perma_Region_HopTable	`int[]`	Regional frequency table	R&W
Reader_Perma_Region_HopTime	`Integer`	a value not less than 0，default 400	R&W
Reader_Perma_Region_Id	RegionE	Regional frequency band	R&W
Reader_Perma_Antenna_CheckPort	`Boolean`	Check the antenna connection before transmitting power	R&W
Reader_Perma_Gen2_Session	`SessionE`	Session	R&W

		mode	
Reader_Perma_Gen2_Target	TargetE	Target mode	R&W
Reader_Perma_Gen2_Q	Gen2.Q	Q value	R&W
Reader_Perma_Gen2_Millerm	Integer	Unused	R&W
Reader_Perma_reset	Boolean	True to restore the default configuration	Write only

5.9 ReaderTypeE Enumeration constant

Used to initialize the reader, the parameter passed to the reader type, indicating the type of the reader

Reader type

Name	Description
M5E_NA7_ONEANTS	M5e series Single antenna, desktop UHF reader
M5E_A7_TWOANTS	M5e series Dual antenna, fixed
M5E_A7_FOURANTS	M5e series Four antennas, fixed
M6E_A7_FOURANTS	M6e series Four antennas, fixed
MT100_ONEANTS	MT100 series Single antenna, desktop UHF reader

5.10 AntTypeE Enumeration constant

The parameter used to input the reader type when initializing the reader, indicating the number of antennas that initialize the reader

Reader type

Name	Description
ONE_ANT	Single antenna, desktop UHF reader

TWO_ANTS	Dual antenna, fixed
THREE_ANTS	Three antennas, fixed
FOUR_ANTS	Four antennas, fixed

6 abnormal

You can see that most of the above functions are declared with a ReaderException exception. ReaderException is a parent class that represents a reader exception error. Several subclasses are derived from the parent class, as follows: **com.silionmodule.ReaderException**

- └ com.silionmodule.OpFailedException
- └ com.silionmodule.SeriousException
- └ com.silionmodule.HardwareAlertException
- └ com.silionmodule.CommunicationException

OpFailedException　It means that there is no problem communicating with the reader device, and the error code returned by the module is returned. Need to be processed according to the specific error code. Generally speaking, the operation fails, you can ignore it and continue to operate the reader.

CommunicationException Refers to port communication error, may check the connection communication status, and need to reconnect the reader.

SeriousException HardwareAlertException A fatal error can be thrown due to a hardware problem or a problem that can damage the reader. If you are catching this anomaly, please contact the Core Module Technical Support Service.

Note: Some behaviors may cause damage to the reader hardware.

1 Transmitting power at a port without an antenna

2 The temperature of the module is too high, or the temperature of the environment is too high

3 When the tag is read, the antenna faces the metal surface at a close distance. If the transmission power is large, it is easy to be a high-power signal feedback antenna and enter the inside of the reader. Causes damage to the reader amplifier. When this abnormality is detected, all operations should be stopped immediately. And check if the above behavior exists.

7 Thread safe

All methods in the current version of the Reader class are not thread-safe, and use in a multi-threaded environment requires the user to handle the race condition itself. You can use locks, or repulsions, etc.

8 Tool Class

Some type conversions are often encountered in reader applications. Here are some utility functions for users to use, reducing development time. Conversion tool class Functional. In the com.silionmodule package.

Function definition：

public static final String bytes_HexString(byte[] bArray)

Function Description: Used to convert byte array to hex string

Function definition：

public static final String shorts_HexString(short[]bArray)

Function Description: Used to convert a hex string to a short array

Function definition：

public static byte[] hexstr_Byte(String hexStr) throws ReaderException

Function Description:Used to convert a hex string to bytes array

Function definition：

public static short[] bytes_short16s(byte[] bytes) throws ReaderException

Function Description: Used to convert bytes array to short array

Function definition：

public static short[] hexstr_short16s(String hexStr) throws ReaderException

Function Description: Used to convert hex string to short array

Function definition：

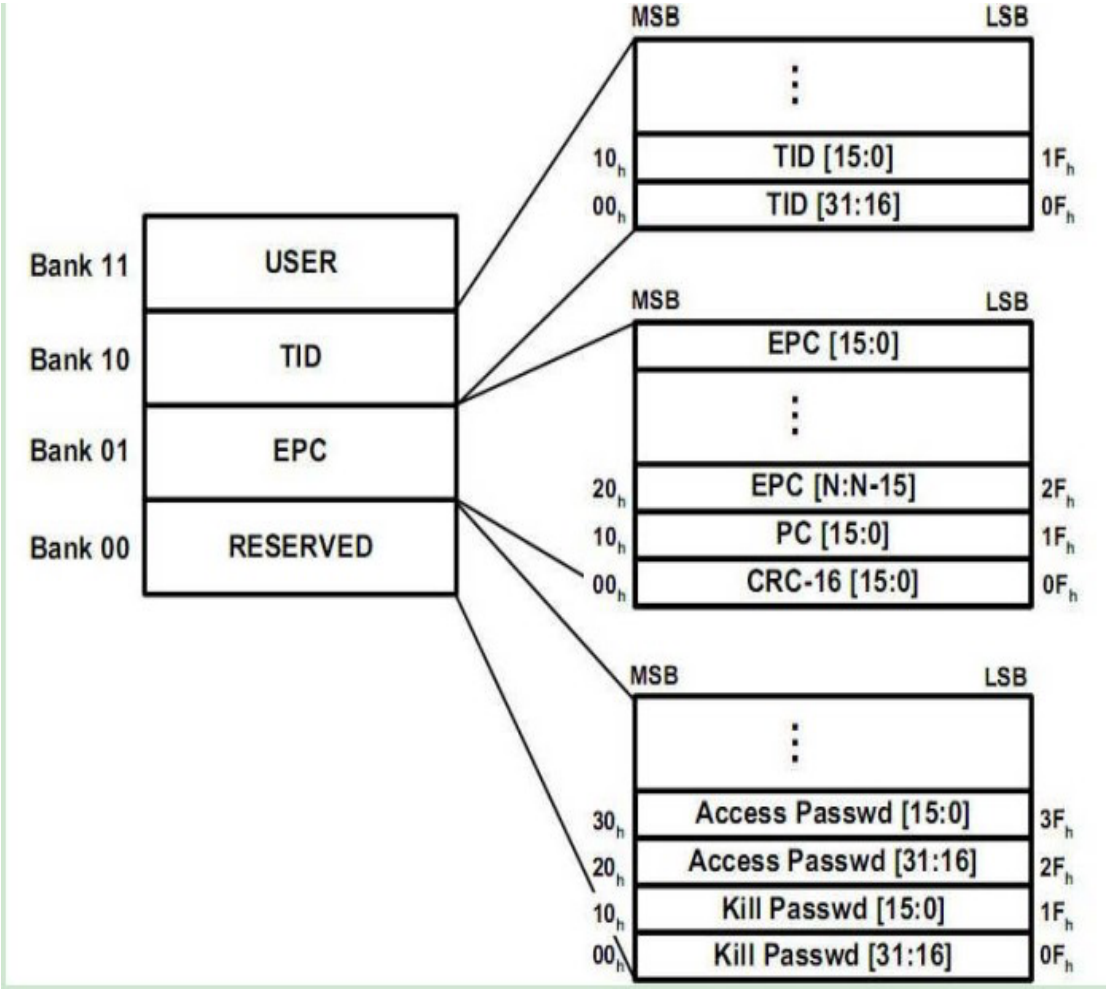
public static short bytes_Short16(byte[] bytes,int startindex) throws ReaderException

Function Description: Used to convert bytes array to short array specify the start index

9 appendix

Gen2 protocol: IS018000-6C, which is a commonly used tags;

Gen2 tag memory map



The memory of the Gen2 tag is divided into four banks, bank0.

Bank1, bank2, bank3, bank0 are also called reserved bank, which store access passwords and destroy passwords. Each password has

32bit. Bank1 is also known as the EPC bank, which also contains the CRC field (16bit), PC field (16bit), EPC code.

(up to 496bit, usually 96bit), bank2 is also known as the TID bank, which contains the world's only serial number.

64bit. Bank3 is also known as the USER bank. The capacity is longer. Different label

brands have different capacities. There are also many labels without bank3.

Addressing starts from block 0. For example, for destroying a password, the memory area it occupies is the 0th block and the 1st block of bank0. All tag operations functions can specify a timeout period. If the operation is completed in the supermarket time, the function will return before the timeout period, otherwise the function will block until the timeout. Multiple antennas can be specified for the Read operation. Only one antenna can be specified for other tags to read, write, lock, and destroy.

For tags read, write, lock, and destroy operations other than Read, if there are multiple tags in the antenna field, the tag of the first responder will be the tag being operated. Therefore, if you want to accurately apply an operation to a specified tag, you should ensure that there is only one target in the field of the antenna, or by setting the filter condition.

The gen2 tag generally has four banks, and the reserved zone has a total of 0~4 and 4 blocks. A 16-bit; EPC area has a total of 0 to 7, a total of 8 blocks, the actual operating memory is 2 to 7 blocks, a total of six. The TID area has a total of 0~4 blocks. And optional user area 0~32 blocks.

ISO18000-6B: In bytes (blocks), 0-7 total 8 blocks are fixed values. That is, the tag id, starting from the 8th block, to 223, a total of 216 blocks can be read and written.