IDENTIUM TECH SOLUTIONS

ModuleAPI_J
development manual
v1.5

Version

date	modify	principal	version
2016.9.30	Original version		1.4
2017-5-3	Change Sdk package name		1.4
2018-10-30	Asynchronous inventory way		1.4
2020-3-31	Params List , temperture ,led tag		1.5

CATALOG

1 Introduction 5
2 enum type introduction 6
Reader_Type enum 6
READER_ERR enum 6
Region_Conf enum 8
Mtr_Param enum 8
Lock_Obj enum 8
Lock_Type enum 9
CustomCmdType enum 10
3 CLASS introduction 10
TAGINFO class 10
BackReadOption class 11
ReadListener interface 12
ReadExceptionListener interface 12
GpiState_ST class 12
GpiInfo_ST class 12
GPITrigger class 13
TagMetaFlags class 14
ErrInfo class 14
NXPChangeEASPara class 14
NXPEASAlarmPara class 15
NXPEASAlarmResult class 15
ALIENHiggs3BlockReadLockPara class 15
IMPINJM4QtPara class 16
IMPINJM4QtResult class 16
EmbededSecureRead_ST class 17
Default_Param class 17
Tagtemperture_DATA class 17
TagLED_DATA class 18
CustomParam_ST class 18
TagSelector_ST class 18
MultiTagSelectors_ST class 19
4 Functions: 19
4.1 Initialization and close functions 19
InitReader_Notype 19
InitReader 20
CloseReader 21
4.2 setting reader’s parameter functions 21
ParamGet 21
ParamSet 21
GetReaderAddress 34
GetLastDetailError 35

4.3 GPIO operating functions 35
SetGPO 35
GetGPI 36
GetGPIEx 36
4.4 string handle function 37
Hex2Str 37
Str2Hex 38
4.5 Tag operating functions 38
TagInventory 40
(a)TagInventory_Raw 41
(b)GetNextTag 42
TagInventory_BaseType 43
Asynchronous inventory way 44
(a)Start to asynchronous inventory 44
(b)Listener tag data and exception 44
(c)Stop asynchronous inventory 45
GetTagData 45
WriteTagData 47
WriteTagEpcEx 48
LockTag 49
KillTag 52
CustomCmd 53
PsamTransceiver 54
4.6 Tag Filter 55
4.7 Additional Data Setting 55
4.8 fast inventory mode 56
4.9 Smart mode 59
4.10 Temperature tag 60
4.11 LED tag 62
6 Error Handling 63
7 Thread Safety 64
8 Samples 64
8.1 read TID bank data by filter epc id 64
8.2 read TID bank data by Additional Data 66

1 Introduction

Windows pc java SDK：

ModuleAPI_C.dll is use for M5e，M6e，slr series UHF modules reader 。It calls ACE.dll,PCOMM.DLL internal on Windows 0x86 operation system。（Note that it need't ACE.dll and PCOMM.dll on the 64 bits operation system. If source of demo loads the two library,please remove them.To call the library must install vc2008 vcredist_x86 or vc2008 vcredist_x64 supported）。ModuleAPIJni.dll is packed using JNI。Please put them in the right path as java pack ModuleAPI_J.jar or put libModuleAPIJni.so or ModuleAPIJni.dll in the path within the paths of System.getProperty("java.library.path")。

ModuleAPI_J.jar is the interface for java. Fit for develop in windows system with java.

Linux java SDK：

Import the ModuleAPI_J.jar for Linux,the ModuleAPI_J.jar is packed using libModuleAPIJni.so . libModuleAPIJni.so is JNI interface and is complied on Linux System

Android SDK：

Import the ModuleAPI_J.jar for Android,the ModuleAPI_J.jar is packed using libModuleAPIJni.so . libModuleAPIJni.so is JNI interface and is complied on Android System

Import

When you build a new project,add the extra jar named ModuleAPI_J.jar. and put need files as other ".dll" files or ".so" files in the project root path(put in the \libs\armeabi\ in Android Project) or put them in the path within the paths of System.getProperty("java.library.path").

Use the package as follows:

```
import com.uhf.api.cls.Reader;
import com.uhf.api.cls.Reader.*;
```

all the user interface are define in this package. User should read these class define in the package.

2 enum type introduction

Reader_Type enum

public enum Reader_Type

```
{
    MODULE_TWO_ANTS, //for two antennas reader
    MODULE_FOUR_ANTS, //for four antennas reader
    MODULE_THREE_ANTS, //for three antennas reader
    MODULE_ONE_ANT, //for one antennas reader
    PR9000, //PR9000 architecture  one antennas reader
    MODULE_ARM7_TWO_ANTS, //ARM7 architecture M5E two antennas reader
    MODULE_ARM7_FOUR_ANTS, //ARM7 architecture M5E four antennas reader
    M6E_ARM7_FOUR_ANTS, //ARM7 architecture  M6E four antennas reader
    M56_ARM7_FOUR_ANTS,//ARM7 architecture  M56 four antennas reader

    R902_M1S,//R902 one antennas reader

    R902_M2S,//R902 two antennas reader
    ARM7_16ANTS,//ARM7 architecture  16 antennas reader

    SL_COMMN_READER,//Sl general reader
}; //for reader type
```

READER_ERR enum

```
public  enum READER_ERR
{
            MT_OK_ERR, // Operation Succeeded
            MT_IO_ERR,  //  Errors  occur  in  network  connection  or  serial
```

```
                    connection.
            MT_INTERNAL_DEV_ERR, // Deprecated error code
            MT_CMD_FAILED_ERR, // Operation Failed
            MT_CMD_NO_TAG_ERR, // No tags found
            MT_M5E_FATAL_ERR, // Deprecated error code
            MT_OP_NOT_SUPPORTED, // Operation not supported
            MT_INVALID_PARA, // Invalid Parameter
            MT_INVALID_READER_HANDLE, // Invalid reader handle
            MT_HARDWARE_ALERT_ERR_BY_HIGN_RETURN_LOSS,// High return loss,
                                                      check the antenna and
                                                      environment
            MT_HARDWARE_ALERT_ERR_BY_TOO_MANY_RESET,// Reset too many times
            MT_HARDWARE_ALERT_ERR_BY_NO_ANTENNAS, // No antenna detected
            MT_HARDWARE_ALERT_ERR_BY_HIGH_TEMPERATURE, // High temperature
            MT_HARDWARE_ALERT_ERR_BY_READER_DOWN, // Reader crashed
            MT_HARDWARE_ALERT_ERR_BY_UNKNOWN_ERR, // Unknown error
            M6E_INIT_FAILED, // M6E Initialization Failed
            MT_OP_EXECING, // Reader busy
            MT_UNKNOWN_READER_TYPE, // Unknown reader type
            MT_OP_INVALID, // Invalid operation
    MT_HARDWARE_ALERT_BY_FAILED_RESET_MODLUE,// reset rfid module failed
            MT_MAX_ERR_NUM,
            MT_MAX_INT_NUM,// = 0xffffffff2,
}; //return value of reader opration function


SL_TagProtocol enum
public  enum SL_TagProtocol
{
    SL_TAG_PROTOCOL_NONE              = 0x00, //none
```

```
    SL_TAG_PROTOCOL_ISO180006B          = 0x03, //ISO18000-6b
    SL_TAG_PROTOCOL_GEN2                = 0x05, //GEN2
    SL_TAG_PROTOCOL_ISO180006B_UCODE    = 0x06,  //ISO18000-6B-UCODE
    SL_TAG_PROTOCOL_IPX64               = 0x07, //IPX64
    SL_TAG_PROTOCOL_IPX256              = 0x08, //IPX256
};


Region_Conf enum
public  enum Region_Conf
{
    RG_NA = 0x01, // FCC 47 CFG Ch. 1 Part 15 Industrie Canada RSS-210 NA
    RG_EU = 0x02, //ETSI EN 302 208  Europe
    RG_EU2 = 0X07, // ETSI EN 300 220  Europe 2
    RG_EU3 = 0x08, // Revised ETSI EN 302 208  Europe 3
    RG_KR = 0x03, // MIC   Korea
    RG_PRC = 0x06, // SRRC, MII   China
    RG_OPEN = 0xFF, // No regulatory compliance enforced
} ; // Frequency regulatory of regions
```

Mtr_Param enum

For setting parameters of reader. For detailed information please check the introduction of ParamGet and ParamSet.

```
Lock_Obj enum
public  enum Lock_Obj
{
    LOCK_OBJECT_KILL_PASSWORD = 0x01, //lock object is kill password
    LOCK_OBJECT_ACCESS_PASSWD = 0x02, //lock object is access password
```

```
    LOCK_OBJECT_BANK1 = 0x04, //lock object is bank1
    LOCK_OBJECT_BANK2 = 0x08,  //lock object is bank2
    LOCK_OBJECT_BANK3 = 0x10, //lock object is bank3
};

Lock_Type enum
public  enum Lock_Type
{
    KILL_PASSWORD__UNLOCK = 0x0000, //Unlock kill password
    KILL_PASSWORD__LOCK = 0x0200, //Temporarily lock kill password
    KILL_PASSWORD_PERM_LOCK = 0x0300, //Permanently lock kill password

    ACCESS_PASSWD_UNLOCK = 0x00, //Unlock access password
    ACCESS_PASSWD_LOCK = 0x80, //Temporarily lock access password
    ACCESS_PASSWD_PERM_LOCK = 0xC0, //Permanently lock access password

    BANK1_UNLOCK = 0x00, //Unlock bank1
    BANK1_LOCK = 0x20, //Temporarily lock bank1
    BANK1_PERM_LOCK = 0x30, //Permanently lock bank1

    BANK2_UNLOCK = 0x00, //Unlock bank2
    BANK2_LOCK = 0x08, //Temporarily lock bank2
    BANK2_PERM_LOCK = 0x0C, //Permanently lock bank2

    BANK3_UNLOCK = 0x00, // Unlock bank3
    BANK3_LOCK = 0x02, //Temporarily lock bank3
    BANK3_PERM_LOCK = 0x03, //Permanently lock bank3
} ;
```

CustomCmdType enum

```
public  enum CustomCmdType
{
    NXP_SetReadProtect, // SetReadProtect command of NXP IC
    NXP_ResetReadProtect, // ResetReadProtect command of NXP IC
    NXP_ChangeEAS, // ChangeEAS command of NXP IC
    NXP_EASAlarm, // EASAlarm command of NXP IC
    NXP_Calibrate, // Calibrate command of NXP IC

    ALIEN_Higgs2_PartialLoadImage, // PartialLoadImage command of ALIEN Higgs2 IC
    ALIEN_Higgs2_FullLoadImage, // FullLoadImage command of ALIEN Higgs2 IC

    ALIEN_Higgs3_FastLoadImage, // FastLoadImage command of ALIEN Higgs3 IC
    ALIEN_Higgs3_LoadImage, // LoadImage command of ALIEN Higgs3 IC
    ALIEN_Higgs3_BlockReadLock, // BlockReadLock command of ALIEN Higgs3 IC

    IMPINJ_M4_Qt, // QT command of IMPINJ Monza4 IC

};
```

3 CLASS introduction

TAGINFO class

For inventory operation, this class is applied to show all the information of every

```
public class TAGINFO
    {
       public byte AntennaID; // The antenna ID that read tag
       public int Frequency; // The frequency point that read tag
       public int TimeStamp; // The time the tag was read, relative to the time the
                             command to read was issued read was issued, in
                             milliseconds
```

```
        public short EmbededDatalen; // The length of embedded data, in bytes
        public byte[] EmbededData=new byte[MAXEMBDATALEN]; // Embedded data
        public byte[] Res=new byte[2]; // Reserve
        public short Epclen; // length of EPC, in bytes
        public byte[] PC=new byte[2]; // PC segment
        public byte[] CRC=new byte[2]; // CRC segment
        public byte[] EpcId=new byte[MAXEPCBYTESCNT]; ///EPC code
        public int Phase; //phase of received signal
        public SL_TagProtocol protocol; //Tag protocol
        public int ReadCnt; //Number of times of the tags have been read
        public int RSSI; //Received signal strength of tag
};
```

BackReadOption class

Option of Asynchronous inventory class, use for seting work detail to Asynchronous inventory

```
public class BackReadOption {
  public short ReadDuration;// inventory cycle,as  unit.according to antenna count and each antenna read time as 200ms to get the inventory cycle.is invalid in high speed mode.
  public int ReadInterval;// it is the time that the reader is not working during inventory cycle,as ms unit,generally is set 0 increase the sleep time is in favour of save electricity and reduce heating (specially using battery or the space is not good for heat dissipation)
  public boolean IsFastRead;// whether use high speed mode(only base on R2000 chip series readers supported), for not many of tags normal, is needn't to use
  public char FastReadDutyRation;//must set value to this property if in the fast inventory mode,reader inventory and sleep ratio. To set 0 for the best effect.
  public TagMetaFlags TMFlags;// must set value to this property if in the fast
```

inventory mode,specified to the return tags with which infomations.

}

ReadListener interface

Asynchronous inventory tags listener interface

```
public interface ReadListener {
    void tagRead(Reader r, TAGINFO[] t);
}
```

Callback function tagRead

Parameter r is reader object,t is return tags array.

ReadExceptionListener interface

Asynchronous inventory exception listener interface

```
public interface ReadExceptionListener {
     void tagReadException(Reader r, READER_ERR re);
}
```

Call back function tagReadException

Parameter r is reader object, re is return value.

GpiState_ST class

Use for said a GPI state

```
public class GpiState_ST {
    public int GpiId; //gpi number
    public int State; //gpi state

}
```

GpiInfo_ST class

Use for said GPIs' s state

```
public class GpiInfo_ST {
    public int gpiCount; //gpiStats count
    public GpiState_ST[] gpiStats;

}
```

GPITrigger class

function: some applications request reader start or stop to inventory when satisfy certain conditions of GPI. This class is used for control reader work by GPI trigger.

Parameters	Description
GpiTrigger1States	The GPI's state to start to inventory, condition of gpi trigger must be set
GpiTrigger2States	Accord to TriggerType's value,it could be as GPI's state to stop inventory and as GPI's to start to iventory.
TriggerType	GPITRIGGER_TRI1START_TRI2STOP: GpiTrigger1States is GPI's state to start to inventory,GpiTrigger2States is GPI's state to stop inventorying; GPITRIGGER_TRI1START_TIMEOUTSTOP: GpiTrigger1States is GPI's state to start to inventory, ReadTimeout is inventory cycle; GPITRIGGER_TRI1ORTRI2START_TIMEOUTSTOP: Start to inventory when condition in accordance with GpiTrigger1States or GpiTrigger2States, after StopTriggerTimeout seconds would stop;
StopTriggerTimeout	When TriggerType is GPITRIGGER_TRI1START_TIMEOUTSTOP or GPITRIGGER_TRI1ORTRI2START_TIMEOUTSTOP, reader

	would inventory in StopTriggerTimeout seconds then stop.

TagMetaFlags class

Tag option data

function: specify to return tags with which informations when had inventoryed in fast inventory mode.

Parameters	Description
IsAntennaID	Whether to return antenna id of tag info
IsEmdData	Whether to return addition data of tag info read the bank data when inventorying,you must set parameter of MTR_PARAM_TAG_EMBEDEDDATA (only base on R2000 chip series reader supported)
IsFrequency	Whether to return frequency of tag info
IsReadCnt	Whether to return read count of tag info
IsRFU	reserver field always as false
IsRSSI	Whether to return rssi of tag info
IsTimestamp	Whether to return time stamp of tag info

ErrInfo class

function: error detail information

```
public class ErrInfo {
     public int derrcode;//error code
     public String errstr;//error decribe
}
```

NXPChangeEASPara class

```
public class NXPChangeEASPara
    {
        public byte[] AccessPwd;//[4] //Access password;if want successfully
```

execute ChangeEAS instructions, tags must set nonzero password

public int isSet; //EAS status, when isSet is set as 1 means it is set, tags would response to EASAlarm instructions; When isSet is set as 0 means it is reset, tags would not response to EASAlarm

public short TimeOut; // the timeout of executing instructions

}

NXPEASAlarmPara class

```
public class NXPEASAlarmPara
    {
        public byte DR; //Divide Ratio as Per Gen2, only support 0x01 currently
        public byte MC; //Miller Cycles, only support 0x02 currently

        public byte TrExt; //TrExt as Per Gen2, only support 0x01 currently
        public short TimeOut; // the timeout of executing instructions

    } ;
```

NXPEASAlarmResult class

```
public class NXPEASAlarmResult
    {
        public byte[]  EASdata; //the return data when successfully executing EASAlarm instructions.
    }
```

ALIENHiggs3BlockReadLockPara class

```
public class  ALIENHiggs3BlockReadLockPara
    {
        public byte[] AccessPwd; //the access password
        public byte BlkBits; //8 bits and every bit correspond to one user bank's piece(each piece include 8 bytes, if the corresponded bit is 1 which means enable read protection for this piece, when it is 0 means disable read protection)
        public short TimeOut; // the timeout of executing instructions
```

```
    }
```

IMPINJM4QtPara class

```
public class IMPINJM4QtPara
    {
        public byte[] AccessPwd; //the access password.
                    public int CmdType; //0 means read QT control bits; 1 means
                write QT control bits. When it is 0 the private instruction function
                would ignore parameters of MemType, PersistType, RangeType.
        public int MemType; //0 means it will use private data profile;1 means it
will use
public data profile.
        public int PersistType; //0 means change QT control bits temporarily;1 means
change QT control bits permanently.
        public int RangeType; //0 means the read range is distant field;1 means the
read range type is near field.
        public short TimeOut; // the timeout of executing instructions

}
```

IMPINJM4QtResult class

```
public class IMPINJM4QtResult
    {
    public int MemType; //0 means it will use private data profile;1 means it will
use public data profile.
        public int RangeType; //0 means the read range is distant field;1 means the
                    read range type is near field.

    } ;
```

EmbededSecureRead_ST class

```
public class EmbededSecureRead_ST
    {
        public int tagtype; //tag type, 1: Alien Higgs3;2:Impinj Monza 4
        public int pwdtype; //password type, 1: fix code; 2: calculate code
        public int ApIndexStartBitsInEpc; //the start address for epc bit
        public int ApIndexBitsNumInEpc; //for password index bit number
        public int bank; // the number of read bank
        public int address; //the start address in bank with block
        public int blkcnt; //read count
        public int accesspwd; //this is access passwrod if using fix code

    } ;
```

Default_Param class

```
public class Default_Param
    {
        public Mtr_Param key;//default parameter key
        public boolean isdefault;// is restore defatult?
        public Object val;// value object
    }
```

Tagtemperture_DATA class

```
public class Tagtemperture_DATA
 {
       public int ReadCount();//the temperature tag's readcount
       public int Lqi();//the temperature tag's Rssi
       public int Frequency();//the temperature tag's frequency
        public int Phase();//the temperature tag's phase
        public int Antenna();//the temperature tag's antenna id
        public int Timestamp();//the temperature tag's timestamp
        public int Protocol();//the temperature tag's protocol
```

```
        public byte[] Data();//the temperature tag's addition data（temperature data）
        public byte[] TagEpc();//the temperature tag's EPC id
        public Tagtemperture DATA();//The constructor
}
```

TagLED_DATA class

```
  public class TagLED_DATA
     {
         public int ReadCount()
         public int Lqi()
         public int Frequency()
         public int Phase()
         public int Antenna()
         public int Timestamp()
         public int Protocol()
         public byte[] Data()
         public byte[] TagEpc()
        public TagLED_DATA()
} the attributes Define the same
```

CustomParam_ST class

```
public class CustomParam_ST
    {
        public String ParamName;//Private parameter names
        public byte[] ParamVal; //Private array parameter value
    }; For key MTR_PARAM_CUSTOM configuration function, parameter values
```

TagSelector_ST class

```
public class TagSelector_ST
    {
        public int bank;//filter bank
        public int startaddr;//filter start bit address
        public int slen;//filter length
        public byte[] sdata;//filter data
        public TagSelector_ST()//The constructor
```

} ; element type of the MultiTagSelectors_ST class

MultiTagSelectors_ST class

```
//Maximum 16 tags filter at the same time
public class MultiTagSelectors_ST
{
    public TagSelector_ST[] tagselectors;//At the same time filter array of tag
    public int tagselectorcnt;//Filter tags number, no more than 16
    public MultiTagSelectors_ST()

};Used to filter the parameter value of the key value
```

MTR_PARAM_TAG_MULTISELECTORS at the same time

4 Functions:

All of the following examples of functions are based on the hypothesis that the hReader is the handle of reader. All the functions with return value will return to MT_OK_ERR after successfully executed.

All functions and enum are define in the Reader Class. As follow: we use the Jreader Object's functions.

Reader Jreader=new Reader();

4.1 Initialization and close functions

InitReader_Notype

READER_ERR InitReader_Notype(String src, int rtype)

Functional description

Initialize the reader

Parameters

Parameters	Description
src	The address of reader, serial number or IP address.
rtype	Amount of the antenna port of reader, it should be set to 1 for desktop card reader and Integrated reader, other types of reader would set to the corresponding value according to the amount of antenna port.

Example:

```
String ip = “192.168.0.250”;
if (Jreader.InitReader_Notype(ip, 4) != READER_ERR.MT_OK_ERR)
{
    System.out.println(“error in InitReader\n”);
}
```

InitReader

```
READER_ERR InitReader(String src, Reader_Type rtype);
```

Functional description

Initialize the reader（Is not recommended to use）

Parameters

Parameters	Description
src	The address of reader, serial number or IP address.
rtype	Reader type

Example

```
String ip = “192.168.0.250”;
if(Jreader.InitReader(ip,Reader_Type.MODULE_FOUR_ANTS)!=READER_ERR.MT_OK_ERR)
{
    System.out.println(“error in InitReader\n”);
}
```

CloseReader

```
public void  CloseReader();
```

Functional description

Close readers

Parameters

Parameters	Description

Example

```
Jreader.CloseReader();
```

4.2 setting reader’s parameter functions

ParamGet

ParamSet

```
READER_ERR ParamGet(Mtr_Param key, Object val);
READER_ERR ParamSet(Mtr_Param key, Object val);
```

Functional description

The two functions can get and set all the parameters of reader.The key parameter shows which parameter will be got and set, the type of val parameter depend on key parameter. See the table below. A parameter of reader will work consistently g until it is set to other value

key	Definition of key	val	Definition of val	Writable
MTR_PARAM_POTL_GEN2_SESSION	Gen2 Protocol Parameters Session	Int[]	Legal values:0,1,2,3	Write & Read
MTR_PARAM_POTL_GEN2_Q	Gen2 Protocol Parameters Q value	Int[]	Legal value:-1---15.(-1:automatically adjust Q value;0---15:static Q value)	Write & Read
MTR_PARAM_POTL_GEN2_TAGENCODING	Gen2 Protocol baseband encoding	Int[]	Legal value:0,1,2,3(0:FM0 only support the readers with m6e architecture;1:the M value of MILLER is 2;2:the M value of MILLER is 4;3:the M value of MILLER is 8), R2000 chip support profile1-profile5:value(0x10-0x15)	Write & Read
MTR_PARAM_POTL_GEN2_MAXEPC	the	Int[]	Legal value:	Write

LEN	maximum supported EPC length in bits		96,496(m6e unsupported)	& Read
MTR_PARAM_RF_ANTPOWER	transmitting power of reader	AntPowerConf	powers:the array of AntPower type and Each element represents the power of an antenna configurations(antid:the antenna ID（starting from 1）; The unit of read Power and write Power is centi-dbm) antcnt:the number of elements has been set up in Powers array.	Write & Read
MTR_PARAM_RF_MAXPOWER	The maximum transmitting power of reader	Int[]	Unit is centi-dbm	read only

MTR_PARAM_RF_MINPOWER	The minimum transmitting power of readers	Int[]	Unit is centi-dbm	read only
MTR_PARAM_TAG_FILTER	Tag filter is the selection criteria for tags when reading, writing, locking, killing and inventory operation	TagFilter_ST	bank:the memory bank to be matched , legal value is 0,1,2,3,4(0--3:gen2 tag's bank0-3; 4:iso180006b memory) Startaddr:the memory bank offset, in bits, at which to begin comparing the Fdata Fdata: the comparing data Flen: the length, in bits, of the Fdata isInvert: whether to invert the selection ;0 means matching	Write & Read

			the filter criteria, 1 means not matching the filter criteria. When you don't use filter criteria you could set TagFilter_ST as NULL.	
MTR_PARAM_TAG_EMBEDEDDATA	Read data of another bank while inventory operation running.	EmbededData_ST	bank: which bank to read when inventory, legal value is 0,1,2,3; startaddr: memory bank offset, in blocks, at which to begin reading bytecnt: how many bytes would be read from the starting address; accesspwd: access password, if the password is not required you could be set	Write & Read

			as NULL. If reading extra data when inventory is not Required you should set EmbededData_ST as NULL.	
MTR_PARAM_TAG_INVPOTL	Set protocol of inventory (only supported by the readers with M6e architecture)	Inv_Potls_ST	potls:the array of Inv_Potl type. Every element represent the inventory operation of some kind of protocol. (potl in Inv_Potl means protocol, weight in Inv_Potl is an integer, the relative weight of each of the Inv_Potl is used to determine what fraction of the total read time is allotted to that protocol when executing	Write & Read

			inventory of multiply protocols) potlcnt:the number of elements in potls array.	
MTR_PARAM_READER_CONN_ANTS	the antennas detected （not all the antennas could be detected)	ConnAnts_ST	connectedants: the antennas found.	read only
MTR_PARAM_READER_AVAILABLE_ANTPORTS	The number of antenna ports of reader	Int[]	the number of antenna ports of reader	read only
MTR_PARAM_READER_IS_CHK_ANT	Whether to detect the antennas on antenna ports before transmitting power	Int[]	Legal value:0,1(0:do not detect antenna before transmitting power;1:detect antenna before transmitting power which is strongly	Write & Read

			recommended)	
MTR_PARAM_READER_VERSION	Version number of reader	Reader_Ver	Currently unavailable	read only
MTR_PARAM_READER_IP	Set or get IP address of reader	Reader_Ip	ip:ip address mask:subnet mask	Write & Read
MTR_PARAM_FREQUENCY_REGION	Frequency regulatory of reader	Region_Conf		Write & Read
MTR_PARAM_FREQUENCY_HOPTABLE	frequency hopping table of reader	HoptableData_ST	htb: frequency points lenhtb: the number of frequency points.	Write & Read
MTR_PARAM_POTL_GEN2_BLF	Gen2 backscatter link frequency, in KHz	Int[]	Typical value are 250,640 Only using FM0 gen2 encoding it could be set as 640, when using Miller gen2 encoding it could only be set as 250	Write & Read
MTR_PARAM_POTL_GEN2_WRITEMODE	Writing mode of Gen2	Int[]	Legal value:0,1(0:write in	Write & Read

	protocol		words;1:wirte in blocks)	
MTR_PARAM_POTL_GEN2_TARGET	Target of Gen2 protocol	Int[]	0:A; 1:B; 2:A->B; 3:B->A	Write & Read
MTR_PARAM_TAGDATA_UNIQUEBYANT	For the same tag, whether it would be regarded as several different tag records when it was read by different antennas.	Int[]	Legal value:0,1(0:no matter how many antennas read the tag there would be only one tag record;1: different tag records when it was read by different antenna for the same tag)	Write & Read
MTR_PARAM_TAGDATA_UNIQUEBYEMDDATA	when inventory with embedded read another bank, some tags are the same	Int[]	Legal value:0,1(0:regard as one tag record;1:regard as multiply tag records)	Write & Read

	epc data but with different other bank data whether consider these tags as different tag data records			
MTR_PARAM_TAGDATA_RECORDHIGHESTRSSI	Whether or not only record the highest rssi value	Int[]	Legal value:0,1(0:record the highest rssi value;1：not to record the highest rssi value)	Write & Read
MTR_PARAM_RF_HOPTIME	Frequency hopping time in milisecond	Int[]		Write & Read
MTR_PARAM_RF_LBT_ENABLE	Enable or disable lbt	Int[]	Legal value:0,1(0: disable;1:enable)	Write & Read
MTR_PARAM_POTL_ISO180006B_	180006b	Int[]	Legal	Write

BLF	backscatter link frequency, in KHz		value:40,160	& Read
MTR_PARAM_POTL_GEN2_TARI	Gen2 protocol Tari	Int[]	Legal value:0,1,2(0:25 ms; 1:12.5 ms;2:6.25 ms)	Write & Read
MTR_PARAM_TAG_EMDSECUREREAD	Secure additional data, it is to point to in the process of inventory can use fixed passwords and password according to read the tag data from a bank. And for Impinj	EmbededSecureRead_ST	Tagtype value: 1:Alien Higgs3; 2:Impinj Monza 4, Other values are not legal pwdtype, 1 : fixed passwords ; 2:Password calculation , Other values are not legal; ApIndexStartBitsInEpcis is as password index start adress of epc bit ; ApIndexBitsNumInEpc is as bit count of password index ; bank point want to the bank ; address ponit start	Write & Read

	Monza 4 IC in the public view to enable QT peek function private view of a certain bank directly read data		address in bank(unit in block) ; blkcnt point read block count; accesspwd point access password(if using fix password)	
MTR_PARAM_TRANSMIT_MODE	Transmit mode of reader	int	Legal value: 0, 1(0:high performance;1:low power. Only supported by m5e reader)	Write & Read
MTR_PARAM_POWERSAVE_MODE	Transmit mode of reader	int	Legal value: 0, 1(0:high performance;1:low power. Only supported by m5e reader)	Write & Read
MTR_PARAM_TAG_SEARCH_MODE	Tag search mode	int	Legal value:0,1 (0:normal	Write &

			mode;1:high speed mode. Only supported by m6e series reader, appropriate to small amount of tags with high speed)	Read
MTR_PARAM_POTL_ISO180006B_MODULATION_DEPTH	Iso180006b modulation depth	int	Legal value: 0, 1(0:99% modulation depth;1:11% modulation depth. Only supported by m6e series reader)	Write & Read
MTR_PARAM_POTL_ISO180006B_DELIMITER	Iso180006b Delimiter	int	Legal value: 0, 4(1:Delimiter1; 4:Delimiter4. Only supported by m6e series reader)	Write & Read
MTR_PARAM_SAVEINMODULE	Default value when module is powered	Default_Param	Only support set to baudrate now	Write
MTR_PARAM_CUSTOM	Custom parameters	CustomParam_ST	Specific label function	Write

MTR_PARAM_TAG_MULTISELECTORS	Set multiple tags to filter at the same time	MultiTagSelectors_ST	The array of filter tags, the length does not exceed 16	Write

MTR_PARAM_CUSTOM keys used to set the tag private function parameters

Tag funciton tagfoucs

```
Reader.CustomParam_ST cpara = myapp.Mreader.new CustomParam_ST();
                    cpara.ParamName = "tagcustomcmd/tagfocus";
                    cpara.ParamVal = new byte[1];
                    cpara.ParamVal[0] = 1;//1 enable 0 disable


READER_ERR ret = myapp.Mreader.ParamSet(Reader.Mtr_Param.MTR_PARAM_CUSTOM,
cpara);
```

Tag funciton fastid

```
Reader.CustomParam_ST cpara = myapp.Mreader.new CustomParam_ST();
                        cpara.ParamName = "tagcustomcmd/fastid";
                        cpara.ParamVal = new byte[1];
                        cpara.ParamVal[0] = 1;//1 enable 0 disable

READER_ERR ret = myapp.Mreader.ParamSet(Reader.Mtr_Param.MTR_PARAM_CUSTOM,
cpara);
```

GetReaderAddress

Function:

To get reader address,usually is ip or serial address

public String GetReaderAddress()

example:

```
return reader.GetReaderAddress();
```

GetLastDetailError

public READER_ERR GetLastDetailError(ErrInfo ei)

function: to get the error detail from reader

example:

```
ErrInfo ei = new ErrInfo();
             reader.GetLastDetailError(ei);
```

4.3 GPIO operating functions

The GPIO function is optional, not all kinds of readers support GPIO

SetGPO

public READER_ERR SetGPO(int gpoid, int val)

Functional description

Set GPO pin’s status

Parameters

Parameters	Description
gpoid	GPO id, one-based numbering
Val	Status value, 1 is high, 0 is low

Example

```
if (Jreader.SetGPO(1, 1) != READER_ERR.MT_OK_ERR)
{
```

```
    printf(“SetGPO failed\n”);
}
```

GetGPI

public READER_ERR GetGPI(int gpoid, int[] val)

Functional description

Get GPI pin’s status

Parameters

Parameters	Description
gpoid	GPI ID, one-based numbering
val	output parameter, return to 1 when GPI is high, return to 0 when GPI is low

Example

```
Int[] val=new int[1];
if (Jreader.GetGPI(1,val) != READER_ERR.MT_OK_ERR)
{
    printf(“GetGPI  failed \n”);
}
```

GetGPIEx

public READER_ERR GetGPIEx(GpiInfo_ST gpist)

function

to get multiple GPI’s state at the same time

parameter

gpist out parameter, with multiple GPI’s state

example

GpiInfo_ST gpist=new GpiInfo_ST();

reader.GetGPIEx(gpist);

4.4 string handle function

Hex2Str

public void Hex2Str(byte[] buf, int len, char[] out);

Functional description

Transform bytes data to hex string

Parameters

Parameters	Description
buf	Bytes array
len	Buf array len
out	Out parameters, is use for saving a hex string value transformation from bytes array. the space of out must be more than 2 times addition 1

Example

```
byte[] hex=new byte[]{(byte) 0xA2,(byte) 0xC8,(byte) 0xD4,(byte) 0xE5};
    int len=4;
    char[] str=new char[4*2];
    Jreader.Hex2Str(hex, len, str);
```

str content as”A2C8D4E5”;

Str2Hex

public void Str2Hex(String buf, int len, byte[] hexbuf);

Functional description

Transform hex string to bytes data,it supports only a hex string that its length is less 600.

Parameters

Parameters	Description
Buf	Hex string
Len	Length of buf,must be a power of 2. Because two hexadecimal character converted to a binary bytes, each representing,each representing high 4 bits and low 4bits of a byte.
Hexbuf	Out parameters, is use for saving binary bytes data transformation form hex string,that length must be more than a half of len

Example:

Sting str = “12345678abcd”;

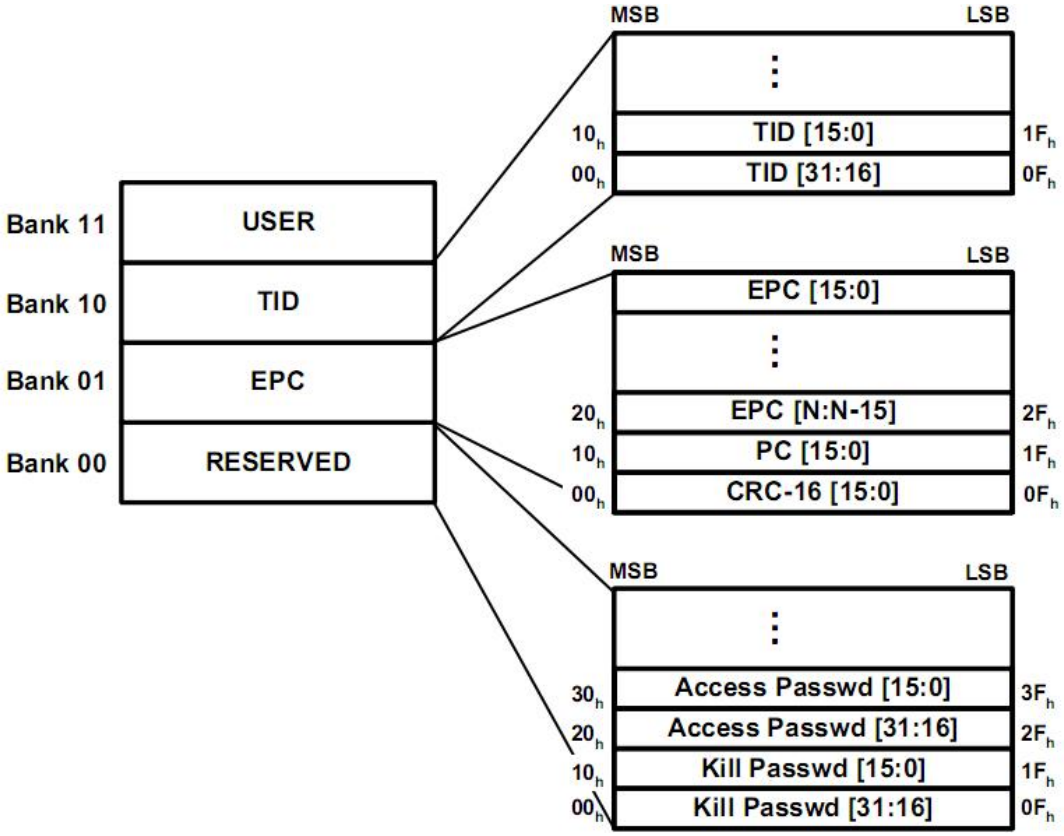
Byte[] out=new byte[100];

Jreader.Str2Hex(str, 12, out);

4.5 Tag operating functions

Gen2 tag is divided into four banks which are bank 0, bank 1, bank 2 and bank 3. Bank 0 is also called the reserve bank which contains the access password and

the kill password while each password has 32 bits. Bank 1 is called EPC bank, which contains CRC field (16 bits), PC field (16 bits) and EPC field (maximum length is 496 bits and the common length is 96 bits). Bank 2 also known as TID bank, which contains tag-and vendor-specific data (for example, a tag serial number). Bank 3 is called user bank, allows user-specific data storage, different tag IC may has different capacity and lots of tags do not have bank 3.



In different bank, the minimum unit of tag operating function is block, zero-based numbering, which is 16 bits. All the tag operations except inventory could be set a timeout period. If the operation is finished before the timeout expires, the function would return before timeout. Otherwise, the function would block until the timeout expires. Inventory operation can use several antennas to search tags while for other tag operations like read, write, lock or kill only one antenna must be specified used for these operations.

For the operations like read, write, lock and kill except inventory, if there are several tags in the antenna field, the first tag response to the reader would

be operated. Therefore, to ensure the operation aim at the specific tag, there must be only one tag in the antenna field or setting the tag filter.

Attention:Do not operate the tag on the antenna ports without antenna connection, this may cause hardware damage.

TagInventory

READER_ERR TagInventory(int[] ants, int antcnt, short timeout, TAGINFO[] pTInfo, int[] tagcnt);

Functional description

Read the EPC code of tags. Please set MTR_PARAM_TAG_INVPOTL parameters before call this function.

Parameters

Parameters	Description
ants	Store operating antennas in this array
antcnt	Ants array' len
timeout	The time period of operation
pTInfo	Output parameter, store the tag data. The memory of this parameter is allocated by users, but the users should estimate the number if the tags within the antenna fields then allocate enough memory space for pTInfo so that it could store all the tag data read.
tagcnt	Output parameter, the number of the tags read, all the tag data store at pTInfo.

Example

Int[] ants = new int[]{1, 2, 3, 4}

```
int antcnt = 4;
TAGINFO[] tags=new TAGINFO[200];
Int[] tagnum=new int[1];

if (Jreader.TagInventory(ants, antcnt, 1000, tags, tagnum) != READER_ERR.MT_OK_ERR)
{
    System.out.println("TagInventory failed");
}
```

Attention: the tags buffer in reader can store 200-1000 tags (depend on reader type).if there are many tags（more than 1000） in the antenna fields, the parameter timedur should not set to a too large value, this can avoid the reader reporting Buffer Full error.

(a)TagInventory_Raw

public READER_ERR TagInventory_Raw(int[] ants, int antcnt, short timeout, int[] tagcnt)

Functional description

Read EPC code of tags, but you cannot get any EPC code from this function except the number of tags read. After calling this function users must immediately call GetNextTag function to get the detailed data of tags.

Parameters

Parameters	Description
ants	Store operating antennas in this array
antcnt	The number of antennas in ants array
timeout	The time period of operation , in milliseconds, for the inventory operation the function would block until timedur is expired.

tagcnt	Output parameter, the number of the tags read.

Example:

```
int ants[] = {1, 2, 3, 4}
int antcnt = 4;
int tagnum;
if (Jreader.TagInventory_Raw(ants, antcnt, 1000, tagnum) != READER_ERR.MT_OK_ERR)
{
    System.out.println ("TagInventory failed");
}
```

(b)GetNextTag

```
public  READER_ERR  GetNextTag(TAGINFO pTInfo);
```

Functional description

Get the data of next tag, users could get the number of tags read by calling TagInventory_Raw function, and then execute GetNextTag function as many times as the number of tags to get all the data of tags read.

Parameters

Parameters	Description
pTInfo	Output parameter, store the tag data.

Example:

```
TAGINFO tag=new TAGINFO();
if (Jreader.GetNextTag (tag) != READER_ERR.MT_OK_ERR)
{
    System.out.println ("TagInventory failed");
}
```

TagInventory_BaseType

public READER_ERR TagInventory_BaseType(int[] ants, int antcnt,short timeout, byte[] outbuf, int[] tagcnt)

function

Read EPC code of tags, this function differs from TagInventory function in the format of tags information returned by reader, this function does not use TAGINFO structure in order to adapt to more development environments. Please set MTR_PARAM_TAG_INVPOTL parameters before call this function.

Parameters

Parameters	Description
ants	Store operating antennas in this array
antcnt	The number of antennas in ants array
timeout	The time period of operation, in milliseconds, for the inventory operation the function would block until timedur is expired.
outbuf	Output parameter, all the data of tags was stored in outbuf in a specific order, for the data of one tag the order are as follows, readcnt (1byte), rssi (1 byte), antennaid (1 byte), Frequency (3 个 bytes), TimeStamp (4 bytes), reserved (2 bytes), Epclen (2 bytes), PC (2 bytes), epc id (Epclen byte), crc (2 bytes), EmbededDatalen (2 bytes), EmbededData (if EmbededDatalen is 0 there is no this field)

tagcnt	Output parameter, the number of the tags read, all the tag data store in outbuf.

example:

```
int[] ants=new int[]{1,2,3};
byte[] outbuf=new byte[512];
int[] tagcnt=new int[1];
reader.TagInventory_BaseType(ants,3,(shor)1000,outbuf,tagcnt);
```

Asynchronous inventory way

(a)Start to asynchronous inventory

public READER_ERR StartReading(int[] ants, int antcnt, BackReadOption pBRO)

function:

after start inventory,it would trigger ReadListener event when read tags, and would trigger ReadExceptionListener event when happen wrong.

Parameters

Parameters	Description
ants	The antennas array in use for inventory
antcnt	The count of ants parameter
pBRO	The inventory option for backgroud work

(b)Listener tag data and exception

Add tag event listener and exception event listener

public void addReadListener(ReadListener listener)

example:

```
reader.addReadListener(listener);
```

public void addReadExceptionListener(ReadExceptionListener listener)

example:

reader.addReadExceptionListener (listener);

(c)Stop asynchronous inventory

public READER_ERR StopReading()

example:

int[] ants=new int[]{1,2,3};

int antcount=3;

BackReadOption m_BROption = new BackReadOption();

m_BROption.IsFastRead = false;

m_BROption.ReadDuration = 200*3;

m_BROption.ReadInterval = 50;

reader.StartReading(ants, antcount, myapp.m_BROption);

GetTagData

public READER_ERR GetTagData(int ant, char bank, int address, int blkcnt, byte[] data, byte[] accesspasswd, short timeout);

Functional description

Read data of tag bank. Before calling this function for ISO18000-6b tag operation the tag filter must be set and the detailed setting should be as follows: bank is 4, flen is 64, startaddr is 0, isInvert is 0. Fdata is the uid of 18000-6b.

Parameters

Parameters	Description
ant	The operating antenna

bank	The bank to read, Value range: from 0 to 4;0-3:use for Gen2 tag;4:use for ISO18000-6b tag.
address	Starting address in bank, in blocks.
blkcnt	The number of block to read.
data	Output parameter, to store the read data.
accesspasswd	If need access password, please fill in the password (4 bytes), if the access password is not necessary this parameter is NULL
timeout	The timeout period of operation

Example

Read two blocks of data from the second block of the bank 0(i.e. access password), and the access password is locked, the access password is 0x12345678.

```
int ant = 1;
char bank = 0;
int addr = 2;
int blks = 2;
byte[] data=new byte[4];

String pwd=”12345678”;
        byte[] pwdb=new byte[4];
        Jreader.Str2Hex(pwd, pwd.length(), pwdb);
if (Jreader.GetTagData(ant, bank, addr, blks, data, pwdb, 1000) !=READER_ERR.MT_OK_ERR)
{
    System.out.println(“GetTagData failed”);
}
```

WriteTagData

public READER_ERR WriteTagData(int ant, char bank, int address, byte[] data, int datalen, byte[] accesspasswd, short timeout);

Functional description

Write data into tag bank. Before calling this function for ISO18000-6b tag operation the tag filter must be set and the detailed setting should be as follows: bank is 4, flen is 64, startaddr is 0, isInvert is 0. Fdata is the uid of 18000-6b.

Parameters

Parameters	Description
ant	The operating antenna
bank	The bank to read, Value range: from 0 to 4;0-3:use for Gen2 tag;4:use for ISO18000-6b tag.
address	Starting address in bank, in blocks.
data	Data to write
datalen	Length of data, in bytes. Note: datalen must be multiple of 2.
accesspasswd	If need access password, please fill in the password (4 bytes), if the access password is not necessary this parameter is NULL
timeout	The timeout period of operation

Example

Write data ”0x111122223333” into bank 3 staring at the second block. No access

```
password.int bank = 3;

int bank = 3;

int ant = 1;

int addr = 2;

char data[6];

char[] datastr = new char[]{'1','1','1','1','2','2','2','2','3','3','3','3'};

Jreader.Str2Hex(datastr, 12, data);

if (Jreader.WriteTagData(ant, bank, addr, data, 6, NULL, 1000) !=READER_ERR.
MT_OK_ERR)

{

    System.out.println("WriteTagData failed\n");

}
```

WriteTagEpcEx

```
public  READER_ERR WriteTagEpcEx(int ant, byte[] Epc, int epclen, byte[] accesspwd,
short timeout);
```

Functional description

Write epc code to EPC bank. The epc code also can be rewritten using WriteTagData function with bank parameter setting to 1, address parameter setting to 2. However there are still differences with WriteTagEpc: WriteTagEpc would change the PC field of EPC bank while writing epc code. PC stores the epc length filed (this function may change the length) and this function does not support tag filter and access password. This function is used in initializing tag.

Parameters

Parameters	Description
ant	The operating antenna
Epc	EPC data to write

epclen	Length of EPC data, in bytes. Note: epclen must be multiple of 2.
accesspwd	If need access password, please fill in the password（4 bytes）, if the access password is not necessary this parameter is NULL
timeout	The timeout period of operation

Example

Write EPC code 0x111122223333111122223333 to EPC bank.

```
String pwd="00000000";
        byte[] data=new byte[]{0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,(byte)
0x88,(byte) 0x99,(byte) 0xaa,(byte) 0xbb};
        //write data
  byte[] pwdb=new byte[4];
          Jreader.Str2Hex(pwd, pwd.length(), pwdb);

    If(Jreader. WriteTagEpcEx (1,data ,2,data,12,pwdb, (short)1000) !=READER_ERR.
MT_OK_ERR)
{
    System.out.println(“WriteTagEpc failed\n”);
}
```

LockTag

public READER_ERR LockTag(int ant, byte lockobjects, short locktypes, byte[] accesspasswd, short timeout);

Functional description

Lock the tag. The following objects are able to be locked: kill password, access password, EPC bank, TID bank, USER bank. The accesspasswd parameter cannot be NULL.

This function could lock multiple objects of one tag at the same time. The lock type can be unlock, temporarily lock or permanently lock. This function only supports gen2 tag.

Parameters

Parameters	Description
ant	The operating antenna
lockobjects	Objects to lock, this parameter can be the value of OR operation on multiple Lock_Obj enumeration values when locking multiple objects.
locktypes	Lock type, this parameter can be the value of OR operation on multiple Lock_Type enumeration values when locking multiple objects.
accesspasswd	Access password
timeout	The timeout period of operation

Example

Temporarily locked bank1, bank3, unlock access password, the access password is ”0x12345678”

```
String pwd=”12345678”;
 byte[] pwdb=new byte[4];
          Jreader.Str2Hex(pwd, pwd.length(), pwdb);
if(Jreader.LockTag(1,(byte)(    JniModuleAPI.Lock_Obj.LOCK_OBJECT_ACCESS_PASSWD.
 .value()|JniModuleAPI.Lock_Obj.LOCK_OBJECT_BANK1.value()|
 JniModuleAPI.Lock_Obj.LOCK_OBJECT_BANK3.value()),(short)JniModuleAPI.Lock_Type
 .KILL_PASSWORD__UNLOCK.value()|(short)JniModuleAPI.Lock_Type.
```

```
 BANK1_LOCK.value()                                                    |
 (short)JniModuleAPI.Lock_Type.BANK3_LOCK.value(),pwdb,1000)      !=READER_ERR.
 MT_OK_ERR)
{
   System.out.println(“LockTag failed\n”);
}
```

Attention: There must be a Lock_Type enumeration value in locktypes corresponding to each Lock_Obj enumeration value in lockobjects.

Lock180006BTag function

```
public  READER_ERR Lock180006BTag(int ant, int startblk,
    int blkcnt, short timeout);
```

Functional description

Lock 18000-6b tag. Before calling this function for ISO18000-6b tag operation the tag filter must be set and the detailed setting should be as follows: bank is 4, flen is 64, startaddr is 0, isInvert is 0. Fdata is the uid of 18000-6b.

Parameters

Parameters	Description
ant	The operating antenna
startblk	starting block,
blkcnt	The number to lock
timeout	The timeout period of operation

Example

Lock 8 consecutive blocks from the ninth block.

```
byte[] uid =new byte[] {0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08};//taguid
TagFilter_ST filter=Jreader.new TagFilter_ST();;
```

```
filter.bank = 4;
filter.fdata = uid;
filter.flen = 64;
filter.startaddr = 0;
filter.isInvert = 0;
Jreader.ParamSet(JniModuleAPI.Mtr_Param.MTR_PARAM_TAG_FILTER, &filter);
Jreader.Lock180006BTag(1, 9, 8, 1000);
```

KillTag

```
public  READER_ERR  KillTag(int ant, byte[] killpasswd , short timeout);
```

Functional description

Destroy tags. Once the tags was destroyed, they can no longer in use. Before destroy a tag users must set the kill password instead of 0.

Parameters

Parameters	Description
ant	The operating antenna
killpasswd	Kill password
timeout	The timeout period of operation

Example

Destroy a tag, its kill password is ”0x43215678”

```
String kpwd=“43215678”;
 byte[] pwdb=new byte[4];
          Jreader.Str2Hex(pwd, pwd.length(), pwdb);
if (Jreader.KillTag(1, kpwdb, 1000) !=READER_ERR. MT_OK_ERR)
{
   System.out.println(“KillTag failed \n”);
```

}

CustomCmd

public READER_ERR CustomCmd(int ant, CustomCmdType cmdtype, Object CustomPara, Ojbect CustomRet);

Functional description

Tag operation of custom command of tag IC.

Parameters

Parameters	Description
ant	The operating antenna
cmdtype	The custom command type. For detailed information see the CustomCmdType enumeration.
CustomPara	Input parameter, different custom command would require different type of parameter.
CustomRet	Output parameter, the return data from custom command, different custom command would return different type of data.

Below is the type of CustomPara and CustomRet with different custom command

cmdtype	CustomPara	CustomRet
NXP_ChangeEAS	NXPChangeEASPara	NONE
NXP_EASAlarm	NXPEASAlarmPara	NXPEASAlarmResult
ALIEN_Higgs3_BlockReadLock	ALIENHiggs3BlockReadLockPara	NONE
IMPINJ_M4_Qt	IMPINJM4QtPara	IMPINJM4QtResult

Example

Execute the custom command of ChangeEAS of NXP tag IC, access password is 0x00000001, set the EAS.

```
JniModuleAPI.NXPChangeEASPara CustomPara3=Jreader.new NXPChangeEASPara();
        CustomPara3.AccessPwd=new byte[]{(byte) 0x99,(byte) 0xaa,(byte) 0xbb,(byte) 0xcc};
        CustomPara3.isSet=1;
        CustomPara3.TimeOut=900;
        If(Jreader.CustomCmd(1, JniModuleAPI.CustomCmdType.NXP_ChangeEAS, CustomPara3, null)!= READER_ERR. MT_OK_ERR)
MessageBox("ChangeEAS failed ");
```

PsamTransceiver

```
public READER_ERR PsamTransceiver(int soltid,int coslen,byte[] cos,
        int[] cosresplen,byte[] cosresp,byte[] errcode,short timeout);
```

Functional description

Send cos command to Psam sub system,and return cos response

Parameters

Parameters	Description
soltid	Psam slot number, legal value of 1 and 2
coslen	Cos command length, in word as unit
cos	Cos command
cosresplen	Out param, cos comand respone data length
cosresp	Out param, cos comand respone data
errcode	Out param, Psam sub system error code, sucessful when cos is 0,failed when not 0

timeout	Psam subsystem Psam card such as response time must be greater than 250 milliseconds

Example

Excute cos command,get random number

```
        int soltid=1;
        int coslen=2;
        byte[] cos=new byte[]{0x11,0x22,0x33,0x44,0x55,0x66,0x77,(byte)
0x88,(byte) 0x99,0x12,0x34};
        int[] cosresplen=new int[1];
        byte[] cosresp=new byte[cos.length+13];
        byte[] errcode=new byte[4];

        READER_ERR er=Jreader.PsamTransceiver(soltid, coslen, cos, cosresplen,
cosresp,errcode, (short)1000);
```

4.6 Tag Filter

Tag operations like inventory, read, write, lock and kill can operate on a specific tag by setting MTR_PARAM_TAG_FILTER parameter. You can specify filter criteria which allow specifying a bank (ban k1, bank 2, bank 3), a starting address in the bank and data to be compared in the bank. As for ISO18000-6b tag, it is meaningless to set tag filter for inventory. Once a tag filter has been set, it will work until another tag filter is set or the tag filter is canceled by setting as NULL.

4.7 Additional Data Setting

When executing the inventory operation, users could get another bank data (we call it additional data) by setting MTR_PARAM_TAG_EMBEDEDDATA parameter except epc code. The member EmbededDatalen of TAGINFO structure indicates the number of bytes of additional data. If the EmbededDatalen is 0 it means there is

no additional data read. The additional data stores in the member EmbededData of TAGINFO structure. This parameter only supports gen2 tag. Setting as NULL will cancel the parameter.

4.8 fast inventory mode

Base on R2000 chip uhf module supports fast mode, it will read more quickly than usually,but need keep the temperature of moudle.there are serveral interfaces functions about the fast inventory mode.

1 enable fast inventory mode

```
public READER_ERR AsyncStartReading(int[] ants,int antcnt,int option)
```

after call the function,the uhf module would be working all the time untill call the stop function.

Parameters

Parameters	Description
ants	Store operating antennas in this array
antcnt	The number of antennas in ants array
option	Set param option: Integer type.start high bit,the third byte is metaflag that is the sign every return items of data,please to get more detail from 1200 uhf module manual; the fourth byte is sleep time on work percent, 0 for 0%, 1 for 5%, 2 for 10%.....10 for 50%。

example:

```
int metaflag = 0;
metaflag |= 0X0001;//return read count
metaflag |= 0X0002; //return RSSI
 metaflag |= 0X0004; //return antenna id
 metaflag |= 0X0008;//return frequency
 metaflag |= 0X0010;// return timestamp
 metaflag |= 0X0080;// return additional data
```

Pause ratio: 50% (proportion of reading time and interval)

```
option = (metaflag << 8)
                | 10;
```

2 stop work in the fast inventory mode,once start read in fast inventory mode,must call the function to stop working,or it is working in all time except happen error.

```
public  READER_ERR AsyncStopReading()
```

Parameters none

3 get count of tags in the buffer,advised to request once about 50ms

```
public  READER_ERR AsyncGetTagCount(int[] tagcnt)
```

Parameters

Parameters	Description
tagcnt	Pass a integer array not empty or not zero length as a parameter,tag count value will save first element of the array

4 get tags data from buffer

```
public READER_ERR AsyncGetNextTag(TAGINFO pTInfo)
```

parameters

Parameters	Description
pTInfo	Pass a new TAGINFO object for get a tag object data from buffer. Should stop to get tag if the function retruns not MT_OK_ERR

4.9 Smart mode

Smart mode is used for reading work that requires a relatively stable reading rate and a temperature control function.

Steps for usage:

1 Add listener

Monitor tag data

Mreader.addReadListener(RL);

Monitor abnormal status

Mreader.addReadExceptionListener(REL);

2 Configure algorithm parameters

Generally do not need to be configured, the default is fine

```
public boolean Set_IT_Params(IT_MODE it_mode,Object[] objs) throws Exception
```

3 Start smart mode inventory

```
public  READER_ERR AsyncStartReading_IT_CT(int[] ants,int antcnt,int option)
```

Consistent with the quick mode interface parameters, but the function name is slightly different

4 Stop

```
public  READER_ERR AsyncStopReading_IT_CT()
```

5 Note that the listener is automatically removed after calling CloseReader()

4.10 Temperature tag

2000-based modules can support certain temperature tags.

1 Read tag temperature

```
public READER_ERR ReadTagTemperature(int ant,char bank, int address,
            int wordcnt, int timeout, int timeselwait, int timereadwait, short metaflag,
            byte[] accesspasswd,R2000_calibration.Tagtemperture_DATA tagtemp)
```

parameter

parameter	description
ant	Specify which antenna to perform the operation
Bank	The bank where the temperature tag stores temperature data
Address	The starting address of the temperature tag to store temperature data
wordcnt	The number of blocks in which the temperature tag stores temperature data
timeout	Total timeout in milliseconds
timeselwait	select timeout (timeout of temperature label manufacturer), 0-300 milliseconds
timereadwait	Read tag waiting timeout, 0-300ms, 100ms is recommended
metaflag	Return information item object
accesspasswd	Access password
R2000_calibration.Tagtemperture_DATA	Temperature data object

exmaple:

```
R2000_calibration.META_DATA rmeta=new R2000_calibration().new META_DATA();
rmeta.IsReadCnt=true;
rmeta.IsEmdData=true;
R2000_calibration.Tagtemperture_DATA tagtemp=new R2000_calibration().new
Tagtemperture_DATA();
                    er= reader.ReadTagTemperature(1,
                            (char) 3, 0x7F, 1,
                            1000,  0, 100, rmeta.getMetaflag(), null, tagtemp);
```

4.11 LED tag

R2000-based modules can light up certain tags.

1 Light up one tag at a time

```
public READER_ERR ReadTagLED(int ant,short timeout,short metaflag,
            R2000_calibration.TagLED_DATA tagled)
```

parameter

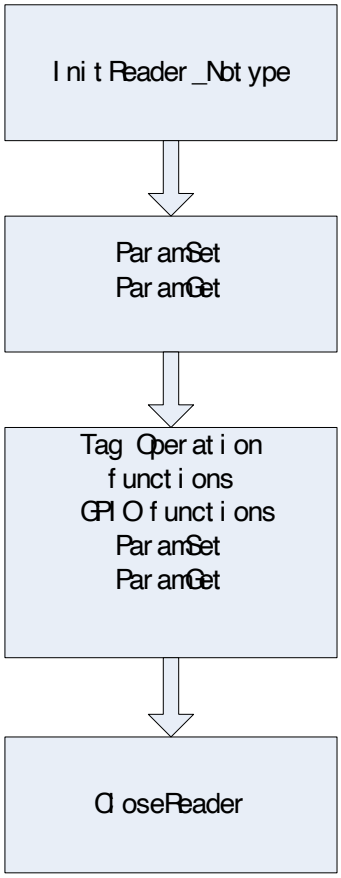
parameter	description
ant	Specify which antenna to perform the operation
timeout	Total timeout in milliseconds
metaflag	Return information item object
R2000_calibration.TagLED_DATA	Led tag object

exmaple:

```
R2000_calibration.TagLED_DATA tagled=new R2000_calibration().new
TagLED_DATA();
er=myapp.Mreader.ReadTagLED(1,(short)1000,rmeta.getMetaflag(),tagled);
```

Support to light up no more than 16 tags at the same time, you need to use MTR_PARAM_TAG_MULTISELECTORS key to set multi-tags filter setting first

5 Life Cycle of Reader



Firstly you should call InitReader_Notype function to initialize reader, after that you may call ParamSet or ParamGet function to configure reader. Now you can call tag operations and GPIO functions. Calling CloseReader means you no longer use reader.

6 Error Handling

All the API functions with returned value will return MT_OK_ERR on success. If the return value is MT_IO_ERR it means there are something wrong with network connection or serial port connection, you should call CloseReader and check these connections firstly, then try to call InitReader_Notype. As for MT_CMD_FAILED_ERR, it just means the failure of the function, it is not a fatal error, and you can do any operation next. As for MT_CMD_NO_TAG_ERR, strictly speaking, it cannot be called

an error, and means there is no tag was found. The errors starting with MT_HARDWARE_ALERT_ERR_BY are serious errors which cannot be ignored. Some improper operations can cause these errors and hardware damage. These operations include: transmit power through antenna ports without antenna connection, the use of unqualified antenna, high ambient temperature and the high return loss caused by metal plate in front of antennas. It would be best to turn off the reader and check the working condition and working environment for these errors.

7 Thread Safety

All the functions, except for InitReader_Notype function, of the current version of SDK are not thread-safe for the same handle of reader. Users should make certain that there is no race condition for the API functions calling or use some synchronization ways of multithread. There is no restriction above for different handles of readers.

8 Samples

8.1 read TID bank data by filter epc id

```
String[] tag = null;
int[] tagcnt=new int[1];
 tagcnt[0]=0;
 ///* clear filter condition
myapp.Mreader.ParamSet(Mtr_Param.MTR_PARAM_TAG_FILTER,null);
        //*/
    READER_ERR er=myapp.Mreader.TagInventory_Raw
    (myapp.Rparams.uants, myapp.Rparams.uants.length,
        (short) myapp.Rparams.readtime, tagcnt);
    if(er==READER_ERR.MT_OK_ERR)
    {
      if(tagcnt[0]>0)
      {
        tag=new String[tagcnt[0]];
```

```
                synchronized (this)
                {
               for(int i=0;i<tagcnt[0];i++)
              {
                TAGINFO tfs=myapp.Mreader.new TAGINFO();
                er=myapp.Mreader.GetNextTag(tfs);
    if(er==READER_ERR.MT_HARDWARE_ALERT_ERR_BY_TOO_MANY_RESET)
                {
                    //error handling,like as stop the read thread
                }
                    if (er == READER_ERR.MT_OK_ERR) {
                        tag[i] = Reader.bytes_Hexstr(tfs.EpcId);
                        // read tid by filter epc id ----------
                        try {
                            byte[] rdata = new byte[12];
                            byte[] rpaswd = new byte[4];
                TagFilter_ST g2tf = myapp.Mreader.new TagFilter_ST();
                            g2tf.fdata = tfs.EpcId;
                            g2tf.flen = tfs.EpcId.length * 8;
                            g2tf.isInvert = 0;
                            g2tf.bank = 1;
                            g2tf.startaddr = 32;
er = myapp.Mreader.ParamSet(Mtr_Param.MTR_PARAM_TAG_FILTER,g2tf);
//read tid bank（2），start block   0   block count 6
Int rbank=2,startblock=0,blockcount=6;
er = myapp.Mreader.GetTagData(tfs.AntennaID, (char) rbank, startblock, blockcount,rdata,
rpaswd, (short) 1000);
                            if(er==READER_ERR.MT_OK_ERR)
                            {String val = "";
                            char[] out = null;
                            out = new char[rdata.length*2];
                        myapp.Mreader.Hex2Str(rdata,rdata.length, out);
                            val = String.valueOf(out).trim();
                            // clear filter
                            g2tf = null;
    er = myapp.Mreader.ParamSet(Mtr_Param.MTR_PARAM_TAG_FILTER,g2tf);
                            }
                        } catch (Exception ex) {
}

                 if(!Tags.containsKey(tag[i]))
                    Tags.put(tag[i],tfs);
                else
                {
```

```
                                TAGINFO tf= Tags.get(tag[i]);
                                tf.ReadCnt+=tfs.ReadCnt;
                                tf.RSSI=tfs.RSSI;
                                tf.Frequency=tfs.Frequency;
                            }

                        }
                        else
                            break;
                    }
                }
                }}
```

8.2 read TID bank data by Additional Data

Note: base on R2000 chip UHF module supported

```
EmbededData_ST edst = myapp.Mreader.new EmbededData_ST();
                edst.accesspwd = null;
                    edst.bank =  2;
                    edst.startaddr = 0;
                    edst.bytecnt = 12;

if (!etapwd.getText().toString().equals("")) {
                    edst.accesspwd = new
   byte[etapwd.getText().length() / 2];
    myapp.Mreader.Str2Hex(etapwd.getText().toString(),
    etapwd.getText().length(), edst.accesspwd);
                    }
                    edst.accesspwd = null;
                } else
                    edst.bytecnt = 0;

READER_ERR er = myapp.Mreader.ParamSet(
        Mtr_Param.MTR_PARAM_TAG_EMBEDEDDATA, edst);
```

Then calls the taginventory function， EmbededDatalen of TAGINFO would be more then 0 when is successful，EmbededData is data of tid bank.