[1 Introduction 3](#_Toc413835192)

[2 Enumeration 3](#_Toc413835193)

[READER\_ERR 3](#_Toc413835194)

[Region\_Conf 4](#_Toc413835195)

[Mtr\_Param 4](#_Toc413835196)

[Lock\_Obj 5](#_Toc413835197)

[Lock\_Type 5](#_Toc413835198)

[Custom Cmd Type 6](#_Toc413835199)

[3 Stuctures 6](#_Toc413835200)

[TAGINFO 6](#_Toc413835201)

[NXPChangeEASPara 7](#_Toc413835202)

[NXPEASAlarmPara 8](#_Toc413835203)

[NXPEASAlarmResult 8](#_Toc413835204)

[ALIENHiggs3BlockReadLockPara 8](#_Toc413835205)

[IMPINJM4QtPara 9](#_Toc413835206)

[IMPINJM4QtResult 9](#_Toc413835207)

[4 Functions： 9](#_Toc413835208)

[4.1 Initialization and close functions 10](#_Toc413835209)

[InitReader\_Notype 10](#_Toc413835210)

[CloseReader 11](#_Toc413835211)

[4.2 functions of setting reader’s parameter 11](#_Toc413835212)

[ParamGet 11](#_Toc413835213)

[ParamSet 11](#_Toc413835214)

[4.3 GPIO operating function 18](#_Toc413835215)

[SetGPO 18](#_Toc413835216)

[GetGPI 19](#_Toc413835217)

[4.4 Tag operating functions 20](#_Toc413835218)

[TagInventory 21](#_Toc413835219)

[TagInventory\_BaseType 22](#_Toc413835220)

[TagInventory\_Raw 24](#_Toc413835221)

[GetNextTag 25](#_Toc413835222)

[GetTagData 26](#_Toc413835223)

[WriteTagData 27](#_Toc413835224)

[WriteTagEpc 29](#_Toc413835225)

[WriteTagEpcEx 30](#_Toc413835226)

[LockTag 32](#_Toc413835227)

[KillTag 35](#_Toc413835228)

[CustomCmd 36](#_Toc413835229)

[4.6 Tag Filter 37](#_Toc413835230)

[4.7 Additional Data Setting 37](#_Toc413835231)

[5 Life Cycle of Reader 38](#_Toc413835232)

[6 Error Handling 38](#_Toc413835233)

[7 Thread Safety 39](#_Toc413835234)

1. Introduction

ModuleAPI\_C.dll internally call ACE.dll and PCOMM.DLL，please put them in the same catalog.

1. Enumeration

READER\_ERR

typedef enum

{

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,

}READER\_ERR; //Return value of Reader API

SL\_Tag Protocol

typedef enum

{

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

} SL\_TagProtocol;

Region\_Conf

typedef enum

{

RG\_NA = 0x01, // FCC 47 CFG Ch. 1 Part 15 Industrie Canada RSS-210

RG\_EU = 0x02, //ETSI EN 302 208

RG\_EU2 = 0X07, // ETSI EN 300 220

RG\_EU3 = 0x08, // Revised ETSI EN 302 208

RG\_KR = 0x03, // MIC

RG\_PRC = 0x06, // SRRC, MII

RG\_OPEN = 0xFF, // No regulatory compliance enforced

} Region\_Conf; // Frequency regulatory of regions

Mtr\_Param

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

Lock\_Obj

typedef enum

{

LOCK\_OBJECT\_KILL\_PASSWORD = 0x01, //Locked object is kill password

LOCK\_OBJECT\_ACCESS\_PASSWD = 0x02, // Locked object is access password

LOCK\_OBJECT\_BANK1 = 0x04, // Locked object is bank1

LOCK\_OBJECT\_BANK2 = 0x08, // Locked object is bank2

LOCK\_OBJECT\_BANK3 = 0x10, // Locked object is bank3

} Lock\_Obj;

Lock\_Type

typedef enum

{

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

} Lock\_Type;

Custom Cmd Type

typedef enum

{

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

} CustomCmdType;

1. Stuctures

TAGINFO

typedef struct

{

unsigned char ReadCnt; //Number of times of the tags have been read

unsigned char RSSI; //Received signal strength of tag

unsigned char AntennaID; //The antenna ID that read tag

unsigned int Frequency; //The frequency point that read tag

unsigned int TimeStamp; // The time the tag was read, relative to the time the command to read was issued

read was issued, in milliseconds

unsigned short EmbededDatalen; //The length of embedded data, in bytes

unsigned char EmbededData[MAXEMBDATALEN]; //Embedded data

unsigned char Res[2]; //Reserve

unsigned short Epclen; //length of EPC, in bytes

unsigned char PC[2]; //PC segment

unsigned char CRC[2]; //CRC segment

unsigned char EpcId[MAXEPCBYTESCNT]; //EPC code

int Phase; //

SL\_TagProtocol protocol; //Tag protocol

} TAGINFO; //For inventory operation，this structure is applied to show all the information of every tag.

NXPChangeEASPara

typedef struct

{

unsigned char AccessPwd[4]; //Access password;if want successfully execute ChangeEAS instructions，tags must set nonzero password

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

unsigned short TimeOut; // the timeout of executing instructions

} NXPChangeEASPara;

NXPEASAlarmPara

typedef struct

{

unsigned char DR; // Divide Ratio as Per Gen2，only support 0x01 currently

unsigned char MC; // Miller Cycles，only support 0x02 currently

unsigned char TrExt; // TrExt as Per Gen2，only support 0x01 currently

unsigned short TimeOut; // the timeout of executing instructions

} NXPEASAlarmPara;

NXPEASAlarmResult

typedef struct

{

unsigned char EASdata[8];//the return data when successfully executing EASAlarm instructions.

} NXPEASAlarmResult;

## ALIENHiggs3BlockReadLockPara

typedef struct

{

unsigned char AccessPwd[4]; //the access password

unsigned char 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)

unsigned short TimeOut; // the timeout of executing instructions

} ALIENHiggs3BlockReadLockPara;

## IMPINJM4QtPara

typedef struct

{

unsigned char AccessPwd[4]; //the access password.

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.

int MemType; //0 means it will use private data profile;1 means it will use

public data profile.

int PersistType; //0 means change QT control bits temporarily;1 means change

QT control bits permanently.

int RangeType; //0 means the read range is distant field;1 means the read

range type is near field.

unsigned short TimeOut; // the timeout of executing instructions

} IMPINJM4QtPara;

## IMPINJM4QtResult

typedef struct

{

int MemType; //0 means it will use private data profile;1 means it will use

public data profile.

int RangeType; //0 means the read range is distant field;1 means the

read range type is near field.

} IMPINJM4QtResult;

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.

* 1. Initialization and close functions

InitReader\_Notype

READER\_ERR stdcall InitReader\_Notype(int \*hReader, char \* src, int antscnt)

Functional description

Initialize the reader

Parameters

|  |  |
| --- | --- |
| Parameters | Description |
| hReader | output parameter, reader handle return to user. |
| src | The address of reader，serial number or IP address. |
| antscnt | 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：

int hReader;

char \*ip = “192.168.0.250”;

if (InitReader\_Notype(&hReader, ip, 4) != MT\_OK\_ERR)

{

printf(“error in InitReader\_Notype\n”);

}

CloseReader

void \_\_stdcall CloseReader(int hReader);

Functional description

Close readers

Parameters

|  |  |
| --- | --- |
| Parameters | Description |
| hReader | Handle of reader |

Example

CloseReader(hReader);

4.2 functions of setting reader’s parameter

ParamGet

ParamSet

READER\_ERR \_\_stdcall ParamGet(int hReader, Mtr\_Param key, void \*val);

READER\_ERR \_\_stdcall ParamSet(int hReader, Mtr\_Param key, void \*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) | Write & Read |
| MTR\_PARAM\_POTL\_GEN2\_MAXEPCLEN | the maximum supported EPC length in bits | int | Legal value: 96,496 | Write & 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.  Note:the default power when reader power on is the 2/3 of the maximum transmitting power of reader | Write & Read |
| MTR\_PARAM\_RF\_MAXPOWER | The maximum transmitting power of reader | unsigned short | Unit is centi-dbm | Write & Read |
| MTR\_PARAM\_RF\_MINPOWER | The minimum transmitting power of readers | unsigned short | Unit is centi-dbm | Write & Read |
| 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 the filter criteria，1 means not matching the filter criteria.  When you don’t use filter criteria you could set TagFilter\_ST as NULL. | Write & Read |
| 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 as NULL.  If reading extra data when inventory is not Required you should set EmbededData\_ST as NULL. | Write & Read |
| 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 inventory of multiply protocols）  potlcnt:the number of elements in potls array. | Write & Read |
| MTR\_PARAM\_READER\_CONN\_ANTS | the antennas detected （not all the antennas could be detected） | ConnAnts\_ST | connectedants: the antennas found.  antcnt: the number of the id of antenna within connectedants. | 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 recommended) | Write & Read |
| 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 FMO 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 protocol | int | Legal value:0,1(0:write in words;1:wirte in blocks) | Write & Read |
| MTR\_PARAM\_POTL\_GEN2\_TARGET | Target of Gen2 protocol | int | Legal value:0,1,2,3(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 epc data but with different other bank data whether consider these tags as different tag data records | int | Legal value:0,1(0:regard as one tag record;1:regard as multiply tag records) | Write & Read |
| 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\_BLF | 180006b backscatter link frequency, in KHz | int | Legal value:40,160 | Write & 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\_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 | Power saving mode | int | Legal value:0,1,2,3(0: no power saving;3: highest power saving) | Write & Read |
| MTR\_PARAM\_TAG\_SEARCH\_MODE | Tag search mode | int | Legal value:0,1 (0:normal mode;1:high speed mode. Only supported by m6e series reader, appropriate to small amount of tags with high speed) | Write & 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\_RF\_ANTPORTS\_VSWR | Antenna ports standing-wave ratio | AntPortsVSWR | reserved for get standing-wave ratio of antenna ports, readers is not supported currently. | Read only |
| MTR\_PARAM\_SAVECONFIGURATION | Permanently save the configuration of reader | int | Legal value: 0,1(1:permanently save configuration;0:delete the configuration，after deletion the reader would apply the default parameters when it power on. This function is not supported by slr1000 and slr1100 modules and the USB port reader made of these modules.  Not all parameters can be saved by all kinds of readers, for details please see the table below) | Write Only |

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter configuration | M5e series | M6e series | slr1100 Ethernet port readers |
| Gen2Session | Yes | Yes | Yes |
| Gen2Target | Yes | Yes | Yes |
| Gen2Qvalue | Yes | Yes | Yes |
| Working area | Yes | Yes | Yes |
| gen2encoding | Yes（no FMO） | Yes | No |
| Antenna detection | Yes | Yes | No |
| transmitting power | Yes | Yes | Yes |
| Uniqueness of data antenna | Yes | Yes | Yes |
| Uniqueness of the attachment data | Yes | Yes | Yes |
| Record the highest Rssi | Yes | Yes | Yes |
| Frequency hopping table | Yes | Yes | Yes |
| power saving mode | Yes | Yes | No |
| Maximum EPC length | Yes | No | No |
| Gen2 writing mode | Yes | Yes | Yes |
| Gen2Trai | No | Yes | No |
| Gen2 backward Scattering rate | No | Yes | No |
| Counting mode | No | Yes | No |
| Iso180006b backward Scattering rate | No | Yes | No |

4.3 GPIO operating function

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

SetGPO

READER\_ERR \_\_stdcall SetGPO(int hReader, int gpoid, int state);

Functional description

Set GPO pin’s status

Parameters

|  |  |
| --- | --- |
| parameters | description |
| hReader | Handle of reader |
| gpoid | GPO id，one-based numbering |
| state | Status value，1 is high，0 is low |

Example

if (SetGPO(hReader, 1, 1) != MT\_OK\_ERR)

{

printf(“SetGPO failed\n”);

}

GetGPI

READER\_ERR \_\_stdcall GetGPI(int hReader, int gpiid, int \*state);

Functional description

Get GPI pin’s status

Parameters

|  |  |
| --- | --- |
| parameters | Description |
| hReader | Handle of reader |
| gpiid | GPI ID，one-based numbering |
| state | output parameter，return to 1 when GPI is high，return to 0 when GPI is low |

Example

int state;

if (GetGPI(hReader, 1, &state) != MT\_OK\_ERR)

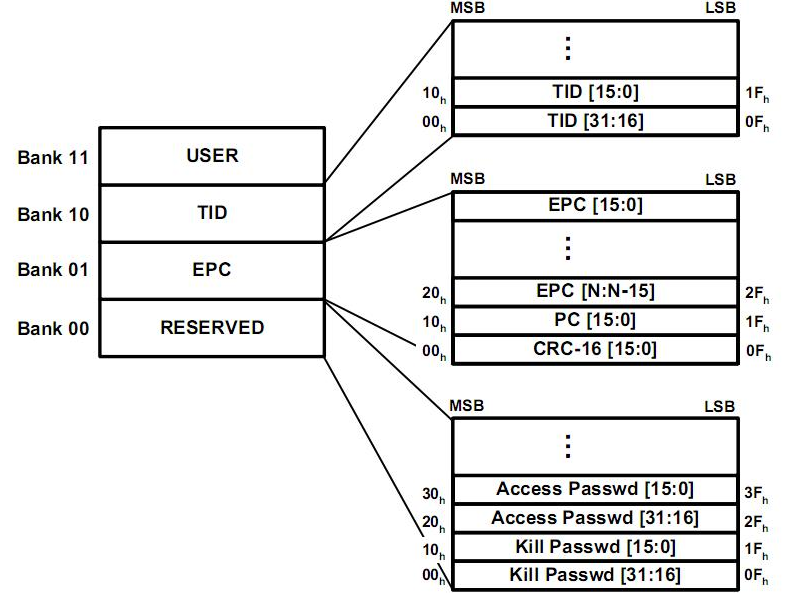
{

printf(“GetGPI failed \n”);

}

4.4 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 \_\_stdcall TagInventory(int hReader, int \*ants, int antcnt, unsigned short timedur, 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 |
| hReader | Handle of reader |
| ants | Store operating antennas in this array |
| antcnt | The number of antennas in ants array |
| timedur | The time period of operation， in milliseconds, for the inventory operation the function would block until timedur is expired. |
| 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[] = {1, 2, 3, 4}

int antcnt = 4;

TAGINFO tags[200];

int tagnum;

if (TagInventory(hReader, ants, antcnt, 1000, tags, & tagnum) != MT\_OK\_ERR)

{

printf(“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.

TagInventory\_BaseType

READER\_ERR \_\_stdcall TagInventory\_BaseType(int hReader, int \*ants, int antcnt, unsigned short timedur, unsigned char \*outbuf, int \*tagcnt);

Functional description

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 like VB, Delphi. Please set MTR\_PARAM\_TAG\_INVPOTL parameters before call this function.

Parameters

|  |  |
| --- | --- |
| Parameters | Description |
| hReader | Handle of reader |
| ants | Store operating antennas in this array |
| antcnt | The number of antennas in ants array |
| timedur | 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[] = {1, 2, 3, 4}

int antcnt = 4;

unsigned char tags[30\*200];

int tagnum;

if (TagInventory\_BaseType (hReader, ants, antcnt, 1000, tags, &tagnum) != MT\_OK\_ERR)

{

printf(“TagInventory\_BaseType failed”);

}

### TagInventory\_Raw

READER\_ERR stdcall TagInventory\_Raw(int hReader, int \*ants, int antcnt, unsigned short timedur, 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 |
| hReader | Handle of reader |
| ants | Store operating antennas in this array |
| antcnt | The number of antennas in ants array |
| timedur | 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. |

例子：

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

int antcnt = 4;

int tagnum;

if (TagInventory\_Raw(hReader, ants, antcnt, 1000, &tagnum) != MT\_OK\_ERR)

{

printf(“TagInventory\_Raw failed”);

}

### GetNextTag

READER\_ERR \_\_stdcall GetNextTag(int hReader, 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 |
| hReader | Handle of reader |
| pTInfo | Output parameter, store the tag data. |

example：

TAGINFO tag;

if (GetNextTag(hReader, &tag) != MT\_OK\_ERR)

{

printf(“GetNextTag failed”);

}

GetTagData

READER\_ERR \_\_stdcall GetTagData(int hReader, int ant, unsigned char bank, unsigned int address, unsigned char blkcnt, unsigned char \*data, unsigned char \*accesspasswd, unsigned 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 |
| hReader | Handle of reader |
| 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;

unsigned char bank = 0;

unsigned int addr = 2;

unsigned int blks = 2;

unsigned char data[4];

unsigned char pwd[4];

pwd[0] = 0x12;

pwd[1] = 0x34;

pwd[2] = 0x56;

pwd[3] = 0x78;

if (GetTagData(hReader, ant, bank, addr, blks, data, pwd, 1000) != MT\_OK\_ERR)

{

printf(“GetTagData failed”);

}

WriteTagData

READER\_ERR \_\_stdcall WriteTagData(int hReader, int ant, unsigned char bank, unsigned int address, unsigned char \*data, int datalen, unsigned char \*accesspasswd, unsigned 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 |
| hReader | Handle of reader |
| 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 ant = 1;

int addr = 2;

unsigned char data[6];

data[0] = 0x11;

data[1] = 0x11;

data[2] = 0x22;

data[3] = 0x22;

data[4] = 0x33;

data[5] = 0x33;

if (WriteTagData(hReader, ant, bank, addr, data, 6, NULL, 1000) != MT\_OK\_ERR)

{

printf(“WriteTagData failed\n”);

}

WriteTagEpc

READER\_ERR \_\_stdcall WriteTagEpc(int hReader, int ant, unsigned char \*Epc, int epclen, unsigned 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 |
| hReader | Handle of reader |
| ant | The operating antenna |
| Epc | EPC data to write |
| epclen | Length of EPC data, in bytes. Note: epclen must be multiple of 2. |
| timeout | The timeout period of operation |

Example

Write EPC code 0x111122223333111122223333 to EPC bank.

char \*epcstr = ”111122223333111122223333”;

unsigned char epcdata[12];

epcdata[0] = 0x11;

epcdata[1] = 0x11;

epcdata[2] = 0x22;

epcdata[3] = 0x22;

epcdata[4] = 0x33;

epcdata[5] = 0x33;

epcdata[6] = 0x11;

epcdata[7] = 0x11;

epcdata[8] = 0x22;

epcdata[9] = 0x22;

epcdata[10] = 0x33;

epcdata[11] = 0x33;

if (WriteTagEpc(hReader, 1, epcdata, 12, 1000) != MT\_OK\_ERR)

{

printf(“WriteTagEpc failed\n”);

}

### WriteTagEpcEx

READER\_ERR \_\_stdcall WriteTagEpcEx(int hReader, int ant, unsigned char \*Epc,

int epclen, unsigned char \*accesspwd, unsigned short timeout);

Functional description

Do the same thing as WriteTagEpc except for supporting setting tag filter and adding access password parameter.

Parameters

|  |  |
| --- | --- |
| Parameters | Description |
| hReader | Handle of reader |
| 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 with access password 0x12345678.

char \*epcstr = ”111122223333111122223333”;

unsigned char epcdata[12];

unsigned char pwd[4];

pwd[0] = 0x12;

pwd[1] = 0x34;

pwd[2] = 0x56;

pwd[3] = 0x78;

epcdata[0] = 0x11;

epcdata[1] = 0x11;

epcdata[2] = 0x22;

epcdata[3] = 0x22;

epcdata[4] = 0x33;

epcdata[5] = 0x33;

epcdata[6] = 0x11;

epcdata[7] = 0x11;

epcdata[8] = 0x22;

epcdata[9] = 0x22;

epcdata[10] = 0x33;

epcdata[11] = 0x33;

if (WriteTagEpcEx(hReader, 1, epcdata, 12, pwd, 1000) != MT\_OK\_ERR)

{

printf(“WriteTagEpcEx failed\n”);

}

LockTag

READER\_ERR \_\_stdcall LockTag(int hReader, int ant, unsigned char lockobjects, unsigned short locktypes, unsigned char \*accesspasswd, unsigned 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 |
| hReader | Handle of reader |
| 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”

unsigned char pwd[4];

pwd[0] = 0x12;

pwd[0] = 0x34;

pwd[0] = 0x56;

pwd[0] = 0x78;

if (LockTag(hReader, 1，

LOCK\_OBJECT\_ACCESS\_PASSWD | LOCK\_OBJECT\_BANK1 | LOCK\_OBJECT\_BANK3, ACCESS\_PASSWD\_UNLOCK | BANK1\_LOCK | BANK3\_LOCK, pwd, 1000) != MT\_OK\_ERR)

{

printf(“LockTag failed\n”);

}

Attention： There must be a Lock\_Type enumeration value in locktypes corresponding to each Lock\_Obj enumeration value in lockobjects.

Lock180006BTag

READER\_ERR \_\_stdcall Lock180006BTag(int hReader, int ant, int startblk,

int blkcnt, unsigned 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 |
| hReader | Handle of reader |
| 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.

unsigned char uid[] = {1,2,3,4,5,6,7,8};//tag uid

TagFilter\_ST filter;

filter.bank = 4;

filter.fdata = uid;

filter.flen = 64;

filter.startaddr = 0;

filter.isInvert = 0;

ParamSet(hReader, MTR\_PARAM\_TAG\_FILTER, &filter);

if (Lock180006BTag(hReader, 1, 9, 8，1000) != MT\_OK\_ERR)

{

printf(“Lock180006BTag failed\n”);

}

KillTag

READER\_ERR \_\_stdcall KillTag(int hReader, int ant, unsigned char \*killpasswd , unsigned 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 |
| hReader | Handle of reader |
| ant | The operating antenna |
| killpasswd | Kill password |
| timeout | The timeout period of operation |

Example

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

unsigned char kpwd[4];

kpwd[0] = 0x43;

kpwd[0] = 0x21;

kpwd[0] = 0x56;

kpwd[0] = 0x78;

if (KillTag(hReader, 1, kpwd, 1000) != MT\_OK\_ERR)

{

printf(“KillTag failed \n”);

}

CustomCmd

READER\_ERR \_\_stdcall CustomCmd(int hReader, int ant, CustomCmdType cmdtype, void \*CustomPara, void \*CustomRet);

Functional description

Tag operation of custom command of tag IC.

Parameters

|  |  |
| --- | --- |
| Parameters | Description |
| hReader | Handle of reader |
| 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．

NXPChangeEASPara para;

para.AccessPwd[0] = 0;

para.AccessPwd[1] = 0;

para.AccessPwd[2] = 0;

para.AccessPwd[3] = 1;

para.isSet = 1;

para.TimeOut = 500;

if (CustomCmd(m\_pdlg->m\_reader, 1, NXP\_ChangeEAS, &para, NULL) != MT\_OK\_ERR)

{

printf("CustomCmd failed\n");

}

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.

1. 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.

1. 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.

1. 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.