**VD-67E**

**UHF RFID Desktop Reader**

**Development handbook**

Contents

[1 Develop application of unilateral communication 4](#_Toc388560684)

[1.1 Develop application of unilateral communication 4](#_Toc388560685)

[1.2 USB Simulation keyboard output agreement 4](#_Toc388560686)

[2 USB bilateral communication agreement 4](#_Toc388560687)

[2.1 Summary 5](#_Toc388560688)

[2.1.1 Command packet format 5](#_Toc388560689)

[2.1.2 Return format 5](#_Toc388560690)

[2.1.3 Error code 6](#_Toc388560691)

[2.1.4 Example 7](#_Toc388560692)

[3 USB control command format 7](#_Toc388560693)

[3.1 Get Reader Version 7](#_Toc388560694)

[3.2 Set Output Power 8](#_Toc388560695)

[3.3 Set Frequency 9](#_Toc388560696)

[3.4 Read Param 9](#_Toc388560697)

[3.5 WriteParam 10](#_Toc388560698)

[3.6 Reboot 13](#_Toc388560699)

[4 Command format of reading /writing ISO18000-6B 14](#_Toc388560700)

[4.1 Read Label ID 14](#_Toc388560701)

[4.2 List ID Report 15](#_Toc388560702)

[4.3 List Selected ID 15](#_Toc388560703)

[4.4 Read Byte Block 16](#_Toc388560704)

[4.5 Write Byte Block 17](#_Toc388560705)

[4.6 Write Protect 17](#_Toc388560706)

[4.7 Read Write Protect 18](#_Toc388560707)

[4.8 Write A Byte 18](#_Toc388560708)

[5 Read/write EPC C1G2 command format 19](#_Toc388560709)

[5.1 EPC1G2\_List Tag ID 19](#_Toc388560710)

[5.2 EPC1G2\_Get ID List 21](#_Toc388560711)

[5.3 EPC1G2\_Read Word Block 21](#_Toc388560712)

[5.4 EPC1G2\_Write Word Block 22](#_Toc388560713)

[5.5 EPC1G2\_Set Lock 23](#_Toc388560714)

[5.6 EPC1G2\_Write EPC 24](#_Toc388560715)

[5.7 EPC1G2\_Read Protect 25](#_Toc388560716)

[5.8 EPC1G2\_RST Read Protect 26](#_Toc388560717)

[6 Command code summary 27](#_Toc388560718)

[6.1 EPC Class1Gen2 command 27](#_Toc388560719)

[6.2 ISO18000-6B Command 27](#_Toc388560720)

[6.3 Other command 27](#_Toc388560721)

[7 sdk sOFTWRE DEVELOPMENT 24](#_Toc388560722)

7.1 The constitution of sdk……………………………………………………………………………..24

7.2 The statement of design…………………………………………………………………………..24

7.2.1 Basic parts and structure……………………………………………………………….24

7.2.1.1 Definition of parts…………………………………………………………………………24

7.2.1.2 API function return code……………………………………………………………….25

7.2.1.3 Definition of data form……………………………………………………………………25

7.2.1.4 Desktop reader param structure…………………………………………………….25

7.2.2 Control command function………………………………………………………………29

7.2.2.1 Connect the USB desktop reader…………………………………………………..29

7.2.2.2 Break the connection……………………………………………………………………..29

7.2.2.3 Check USB online…………………………………………………………………………..29

7.2.2.4 Initialize the USB……………………………………………………………………………29

7.2.2.5 Close the USB…………………………………………………………………………………29

7.2.2.6 Character and time changed to be decimal format…………………..30

7.2.2.7 Compare the record previous&after to check if they're equal…..30

7.2.2.8 Read version………………………………………………………………………………….31

7.2.2.9 Take simulation keyboard module param………………………………………31

7.2.2.10 Set simulation keyboard module param…………………………………………31

7.2.1.11 Take USB reader basic working param……………………………………………32

7.2.1.12 Set USB reader basic working param…………………………………………….32

7.2.1.13 Restore the factory param of usb reader…………………………………………32

7.2.1.14 Get the usb reader ID……………………………………………………………………..33

7.2.3 Read and write ISO18000-6C function………………………………………………..33

7.2.3.1 Identify the EPC no. of ISO18000-6C tag………………………………………..33

7.2.3.2 Read a range of data…………………………………………………………………………34

7.2.3.3 Write a range of data…………………………………………………………………………35

7.2.3.4 Set the protection while reading and writing………………………………….36

7.2.3.5 Permanent dormancy tag………………………………………………………………….37

7.2.3.6 EAS mode operating command……………………………………………………….37

7.2.3.7 EAS Alarm command………………………………………………………………………..38

7.2.3.8 Set the reading protection………………………………………………………………38

7.2.3.9 Relieve the reading protection……………………………………………………….39

7.2.4 Read and Write ISO18000-6B function…………………………………………….39

7.2.4.1 Identify the ID no. of tag………………………………………………………………..39

7.2.4.2 Identify the ID no. of tag appointed……………………………………………….39

7.2.4.3 Write data module…………………………………………………………………………….40

7.2.4.4 Read data module…………………………………………………………………………….41

7.2.4.5 Write slowly data module………………………………………………………………….41

7.2.4.6 Set the protection mode……………………………………………………………………42

7.2.4.7 Read/Write the protection mode……………………………………………………..43

[8 Tag saver and the notification 43](#_Toc388560723)

9. SCHEDULE

9.1 Parameter statements…………………………………………………………………………45

# Develop application of unilateral communication

## Develop application of unilateral communication

When apply to continuous and triggered auto working mode, the device will force to change to simulation keyboard output mode, PC only needs to accept tag ID from reader, no need of sending command to reader.

## USB Simulation keyboard output agreement

USB simulation keyboard unilateral output, only can upload reader’s data to PC through USB, and readers will not accept any command from PC.

The fixed format of simulation keyboard output

|  |  |  |
| --- | --- | --- |
| Prefix code | Hex and decimal in ASCII code | Enter and Wrap |

**Comments: The prefix’ maximum is 20 bytes. Output contends and length are set according to working parameters of reader.**

# USB bilateral communication agreement

To develop application has two ways:

1.Use USB communication agreement’s control code, directly operate on the reader.

2. Use SDK software for this reader, call API functions to operate with the reader.

## Summary

In RFID application system, the reader is connected with communication controller(or PC) through USB, and receive the command sent by controller, and return the executed command ending to controller. So, we call the data communication packet as command packet, and the data communication packet sending to controller is called return packet.

### Command packet format

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BootCode | Length | Command | Command Param | Check Sum |
| 1Byte | 1Byte | 1Byte | N Byte | 1Byte |

As above picture, command packet is divided to 5 sections:

**BootCode**: guidance code, 1 byte, fixed as 40H.

**Length**: packet’s valid length, 1 byte. This length is total bytes of last three parts.

**Command**:command code, 1 byte.

**Command Param**:the length vary according to command.

**CheckSum**:1 byte, Byte complement after add up boot code to command param then discard carry.

### Return format

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BootCode | Length | Command | Return Data | Check Sum |
| 1Byte | 1Byte | 1Byte | N Byte | 1Byte |

As above picture, return format is also devided into of 5 parts.

**BootCode:** guidance code, 1 byte, when command executed properly, return packet’s guidance code is F0H, when command failed to execute, return packet’s guidance code is F4H.

**Length**: packet’s valid length, 1 byte. This length is total bytes of last three parts.

**Command**: command code, 1byte, same as command code received, it shows the response for return packet.

**Return Data**:return ending of executed command, the length vary from command.

**CheckSum**: checksum, 1 byte, Byte complement after add up boot code to ReturnData then discard carry.

### Error code

When command failed to execute, the guidance code of return packet is F4H,and the return Data is one byte of error code.Common error code:

|  |  |
| --- | --- |
| 00(00H) | Command succeeded or detect correctly |
| 01(01H) | Antenna fail to connect |
| 02(02H) | No tags detected |
| 03(03H) | Illegal tags |
| 04(04H) | insufficient power |
| 05(05H) | This area is write protected |
| 06(06H) | Checksum error |
| 07 (07H) | Parameter error |
| 08 (08H) | Data area not existed |
| 09 (09H) | Incorrect password |
| 10 (0AH) | Kill password can’t be null |
| 11(0BH) | When the reader in the active work, this command is illegal. |
| 12(0CH) | Illegal user with unmated password |
| 13(0dH) | Radio frequency interference is at external |
| 14 (0EH) | Tag reading protected |
| …… |  |
| 30(1EH) | Invalid command, such as when the command is with incorrect parameters |
| 31(1FH) | Unknown command |
| 32(20H) | Other error |

### Example

E.g. set the baud rate of reader as 9600bps, the command packet is:『40H 03H 01H 04H B8H』

Among:

|  |  |
| --- | --- |
| 40H | guidance code |
| 03H | packet’s valid length is 3 bytes |
| 01H | Command code for『Set baud rate of reader 』 |
| 04H | Stands for 9600bps |
| B8H | Checksum |

Is 40H+03H+01H+04H=48H’s complement code (change to opposite and add 1)

If execute correctly, return packet is : 『F0H 02H 01H 0DH』

If execute wrongly, return packet is『F4H 03H 01H 1FH E9H』

# USB control command format

## Get Reader Version

Function: Get hardware and software’s version of reader.

Command code: 02H

Command parameter: none

Command packet『40H 02H 02H BBH』

Return data: If command execute correctly, then the data part of return packet is a 4-byte edition number.

|  |  |
| --- | --- |
| Byte0 | Hardware’s principal version |
| Byte1 | Hardware’s minor version(Use hardware’s version number to show reader’s model |
| Byte2 | Software’s principal version |
| Byte3 | Software’s minor version |

E.g: If reader’s model is Reader1102, software’s version number is V1.5，then return packet is:

『F0H 06H 02H 0BH 02H 01H 05H DDH』

## Set Output Power

Function: Set reader’s transmitting power. When the reader is set with new output power, it will be with immediate effect, and keep until resetting, no matter whether has turned off the power.

Command code: 04H

Command parameter:1 byte P, means magnitude of power, the value is 0~63.

Command packet:『40H 03H 04H P CheckSum』

Return packet: If command execute correctly, return packet’s data part will be null.

『F0H 02H 04H 0AH』

## Set Frequency

Function: Set the frequency channel number of reader to launch the microwave signal. Working frequency is immediately effected once set, and will keep until reset no matter whether has turned off the power of reader.

Command code: 05H

Command parameter:2 bytes, the first byte means starting frequency fmin, value is 1~63；The second byte means stop frequency fmax, value is 1~63. If fmax>fmin, which means reader is working in hopping frequency way, and the hopping frequency range is fmin～fmax. If stop frequency equals to starting frequency, which means reader is working in fixed-frequency way, and the frequency is fmax.

Command packet: 『40H 04H 05H fmin fmax CheckSum』

Return data: If command execute correctly, then return packet’s data part is null.

『F0H 02H 05H 09H』

## Read Param

Function: The working frequency written into the reader which is the last command from the reader.

Command code: 06H

Command parameter: null

Command packet: 『40H 02H 06H B8H』

Return data: If succeeded, then the data part of return packet will be 32-byte parameter PAM When setting command and write into

『F0H 22H 06H PAM CheckSum』

## Write Param

Function: To set readers’ series rate, transmitting power, output power and other basic parameters.

Command code: 09H

Command parameter: 32-byte parameter PAM

Command packet: 『40H 22H 09H PAM CheckSum』

Return data: If execute correctly, then the data part of return packet will be null, if succeed, then the data part of return packet will be 32-byte parameter read according to below order.

『F0H 02H 09H 05H』

32 byte parameter(every byte for every parameter):（You can also refer to addendum for detailed explanation, among it the blue parameter is reserved parameter, subject to addendum）

1. Serial port’s communication rate, the value is 00H~08H, representative rate is the same as command 『set serial port’s baud rate』default value is 07H.
2. Transmission power value, the value is 30～160.
3. The starting point of launch microwave signal frequency, the value ( default value is 1) is: 1~63。
4. Select antenna(default value is 1):

|  |  |
| --- | --- |
| 1 | Select ANT1 port to connect antenna |
| 2 | Select ANT2 port to connect antenna |
| 4 | Select ANT3 port to connect antenna |
| 8 | Select ANT4 port to connect antenna |

1. Set work mode of reader(default value is 2):

|  |  |
| --- | --- |
| 0 | timing mode |
| 1 | trigger mode |
| 2 | command mode |
| 3 | timing mode2 |

1. Set reading interval( default value is 0): When working in timing mode, how long it will pause for until next time to read card.

|  |  |
| --- | --- |
| 0 | 10ms |
| 1 | 30ms |
| 2 | 50ms |

1. Set card number output way( default value is 0): when reading the same card number continuously in a long time, whether out put every time.

|  |  |
| --- | --- |
| 0 | Direct output:out put every time |
| 1 | Standard output:out put once every 2 minutes |

1. Set trigger mode(default value is 0):when working in『trigger mode』,select rising trigger or falling edge trigger.

|  |  |
| --- | --- |
| 0 | low level |
| 1 | high level |

1. Set the address to store the card number(default value is 0)

|  |  |
| --- | --- |
| 0 | The ID number of tag itself |
| 1 | User defined ID number |

1. Set whether needs to judge validity of the card(default value is 0):

|  |  |
| --- | --- |
| 0 | no |
| 1 | yes |

1. Set card number output port and format(default value is 0):

|  |  |
| --- | --- |
| 0 | Out put the card number in Wiegand26 format from Wiegand port |
| 1 | Out put the card number in Wiegand34 format from Wiegand port |
| 2 | Out put the card number from RS485 port |
| 3 | Out put the card number from RS232 port |

1. Set the maximum of card can be read (default value is 2):
2. modulation depth, value is 30~ 160.
3. Tag type: default value is 01H;

|  |  |
| --- | --- |
| 01H | ISO18000-6B |
| 02H | EPCC1 |
| 04H | ISO18000-6C |
| 08H | ISO18000-6D |

03H means that can read tag ISO18000-6B and EPCC1 at the same time, and so forth.

1. Width of pulse output from Wiegand port, default value is 40.
2. Interval of pulse output from Wiegandport , default value is 200.
3. Set the starting place of output card number, value is 0～4. default value is 0.
4. The end point of launching the microwave signal frequency, the value is 1~63(default value is 63):
5. Page number and lasting time of reading tag ISO18000-6D, high 4 bits is page number, 0 means no page number, 1~2 means having 1 or 2 pages of data; Low 4 bits is lasting time of reading tag:0-5ms，1-10ms，2-20ms，3-30ms，…，16-160ms. Default value is 14H.
6. Standard out put time interval, default value is 120 s, 1~255.
7. This byte’s function as follow：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit sequence | Bit 7 | | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Function | | - | - | - | - | 1:Networking 0:None Networking | 1:  high  gain  0:  low  gain | 1: dense  reader  0:  non-dense  reader | 1:  enable  buzzer  0:  don’t  enable  buzzer |

1. Address of reader:0 and 255 are broadcast address, all readers will

implement all received commands or active output ID; 1～254 are

private address of reader, reader only implement the received

1. ～30 reservation

31)Transmitting mode:

|  |  |
| --- | --- |
| 0 | Receiving and sending  mode |
| 1 | Sending mode |

Definition of transmitting mode: uninterruptedly transmit RF signal(with

modulation or without modulation).

32)Set modulation:

|  |  |
| --- | --- |
| 0 | Without modulating signal |
| 1 | With modulating signal |

## Reboot

Function: to reboot reader, power off, then power on.

Command code: 0EH

Command parameter: no

Command packet: 『40H 02H 0EH B0H』

ReturnData: if succeed, return data will be null.

『F0H 02H 0EH 00H』

Order format of reader’s address:

Command code:0EH

# Command format of reading /writing ISO18000-6B

For tags, storage capacity inlay is 2048bits, which divides into 256 bytes. There is an address to each byte, correspondent from 0-255.

There into:

* Address 0-7 eight words (64bits): tag ID numbers, fixing when products come out, can not be amended.
* Address 8-223user information storage area, can be self distributed.
* Address 224-255 write protection bytes

## Read Label ID

Function: List tag ID in RF Field.

Command code: FEH

Command parameter: none

command packet:『40H 02H FEH C0H』

Rebound data: If succeed, bytes of rebound data part=number of all list tag M (1 byte) + (tag number of sent out L (<=8)\*8(ID) data.

『F0H 3+L\*8 FEH M L\*8 CheckSum』

Instruction format with reader’s address

Command code: FEH

## List ID Report

Function: get tag ID (already passes rfs\_listID order) from reader’s memory.

Command code: FDH

Command parameter: 2bytes, first byte begins from ADDR, second byte is tag number L (<=8)

Command packet:『40H 04H FDH ADDR L CheckSum』

Rebound data: If succeed, part bytes from rebound data= (tag number\*8(ID))

『F0H 2+L\*8 FDH L\*8 CheckSum』

## List Selected ID

Function: list existing recognized tag ID under range of antenna radiation

Command code: FBH

Command parameter 1:1byte is optional tag condition SEL.

|  |  |
| --- | --- |
| 00 | equal |
| 01 | not equal |
| 02 | than |
| 03 | Less than |

Command parameter 2:1byte is origination adrress ADDR of tag data, values 0-223

Command parameter3:1byte is data mask; bit of it is corresponding to a comparable word.

|  |  |
| --- | --- |
| 0 | This byte is not for comparison |
| 1 | This byte is for comparison |

Command parameter 4:8byte is comparable data.

Command packet: 『40H 0DH FBH SEL ADDR MASK DATA CheckSum』

Rebound data: if succeed, rebound data part bytes=listed number of read tagM (1byte) + (transmitted tag numberL (<=8)\*8(ID))

『F0H 3+L\*8 FBH M L\*8 CheckSum』

## Read Byte Block

Function: read partial data in tag.ISO18000 tag is with memory capacity of 2048bits, i.e.256 bytes. Byte address user can read is 0-223.length of data module takes byte as unit, Max 32byte can be read in a time.

Command code:F6H

Command parameter: 8byte ID,1 byte expresses origination address; values 0-223,1byte expresses module length, values 1-32.

Command packet: 『40H 0CH F6H id aannCheckSum』

Rebound packet: If succeed, rebound data divides nn byte

『F0H nn+2 F6H xx …… xx CheckSum』

## Write Byte Block

Function: write into partial data to tag, take byte as unit for the data, Max 4byte at a time .byte address user can write is 8-223.

Command code: F5H

Command parameter: 8byte ID, 1byte express origination address, values 8-223,1byte expresses module length, values 1-4.write data for nn byte.

Command packet: 『40H 12+nn F5H id aann xx --- xx CheckSum』

Rebound packet: if succeed, rebound data is empty

『F0H 02H F5H 19H』

## Write Protect

Function: write protection on special bytes

Command code: F4H

Command parameter: 8byte ID, 1byteaddress(aa), from 8-223

command packet: 『40H 0BH F4H ID aaCheckSum』

Rebound data: If succeed, leading code of rebound packet is F0H, data part is empty

『F0H 02H F4H 1AH』

## Read Write Protect

Function: read appointed tag if write protection

Command code: F3H

Command parameter: 8byte ID, 1byte origination address(aa), values0-223

Command packet: 『40H 0BH F3H ID aaCheckSum』

Rebound data: if succeed, leading code of rebound packet is F0H,1byte in data part

|  |  |
| --- | --- |
| 0 | unprotected, 『F0H 03H F3H 00H 1AH』 |
| 1 | protected, 『F0H 03H F3H 01H 19H』 |

## Write A Byte

Function: write into special data to special tag, the data length take byte as unit, Max 4byte at a once. byte address can write by user is 8-223

Command code: F2H

Command parameter: 8byte ID, 1byte express origination address(aa), values 8-223,1byte expresses module length(nn), values 1-4.write data for nn byte.

Command packet: 『40H 12+nn F2H id aann xx --- xx CheckSum』

Rebound packet: if succeed, rebound data part is empty

『F0H 02H F2H 1CH』

**Note: this command adopts to write data to tag by one byte and one byte, slow in rate, only used in the case when tag does not support no.5 writing instruction.**

# Read/write EPC C1G2 command format

Memory bank of IS08000-6C tag divides into 4 areas.

1. EPC area: area of storing EPC code, Max memory is 96bits EPC, can read and write.
2. TID area: keep ID number which set by tag manufacturer, 32 and 64Bits option.
3. User area: this area is different for various manufacturer.G2 tag from Impinj company has no user area,company of NXP Is with 96 Bits,can write and read
4. Password area: 32bits visit password and 32Bits kill password,can read and write

## EPC1G2\_List Tag ID

Function: Identify tag ID under radiation range of antenna, according to mask code condition.

Command code: EEH

Command parameter 1:1byte men, choose for data area.

|  |  |
| --- | --- |
| 0 | password area |
| 1 | EPC code |
| 2 | IDnumber of TID tag |
| 3 | User |

Command parameter 2:2byte, introduces origination of mask code.

Command parameter 3:1byte, introduces mask code length

Command parameter4:m byte,mask；If LEN％8=0,then m=LEN/8。If LEN％8≠0,then m=⎣LEN/8⎦+1。

Command parameter4: m byte, mask

Command packet:『40H m+6 EEH memaddr LEN Mask CheckSum』

Rebound data: If succeed, byte of rebound data part=list number of read tagM (1byte) + (transmitted tag numberL (<=8)\*8(ID))\*L (EPC digits+EPC))

**Note: LEN=0expresses all tag ID can be identified under range of antenna radiation**

EPC digit:00H-0Word,01H-1Word,02H-2Word,……,FFH-256Word

『F0H 3+L\*N EEH M L\*N CheckSum』

## EPC1G2\_Get ID List

Function: get tag ID which identifed by rfs\_ListTagID from reader EMS memory

Command code: EDH

Command parameter: 2byte. First byte is serial number of commence, second byte is tag number m (<=8)

Command packet: 『40H 04H EDH no m CheckSum』 command packet

Rebound data: If succeed, digits of rebound data= (1byte tag number M\*L Bytes (EPC digit+EPC)

『F0H 2+L\*8 EDH L\*M CheckSum』

## EPC1G2\_Read Word Block

Function: read partial data from the selected tag, data block is 16 bits in length,

Command code: ECH

Command parameter1:1byte EPC digits L, introduces word number for EPC

Command parameter2: L\*2byte EPC number, introduces to read which tag data

Command parameter 3:1byte mem, choose data area

|  |  |
| --- | --- |
| 0 | Password area |
| 1 | EPC number |
| 2 | IDnumber in TID tag |
| 3 | User |

Command parameter4:1byte origination address (Unite: word)

Command parameter 5:1byte data length len

Command parameter6:4byte AccessPassword, password

command packet:『40H 10+L\*2 ECH L EPC memaddrlenAccessPasswordCheckSum』command packet

Rebound data: If succeed, rebound data part is len\*2byte data

『F0H len\*2+2 ECH xx …… xx CheckSum』

**Note: Access password just works when password area is in password lock.**

## EPC1G2\_Write Word Block

Function: Write partial data into the selected tag, data block take 1 bits in a length,

Command code: EBH

Command parameter1:1byte EPC digits L, introduces word number for EPC code.

Command parameter2: L\*2 byte EPC number

Command parameter3:1byte mem, choose data area MemBank

|  |  |
| --- | --- |
| 0 | password area |
| 1 | No use |
| 2 | IDnumber in TID tag |
| 3 | User |

Command parameter4:1byte origination address (unit word)

Command parameter5:1byte data legthlen

Command parameter6: Len\*2 byte data

Command parameter7:4byte accesspassword, password

Command packet: 『40H 10+L\*2+len\*2 EBH L EPC memaddrlen data AccessPasswordCheckSum』

Rebound data: if succeed, rebound data is empty

『F0H 02H EBH 23H』

**Note: Accesspassword is only efficient when data area is locked. when unlocked, it writes none password, when data is forever locked, password is useless.**

## EPC1G2\_Set Lock

Function: set write protection in designated area of tag

Command code: EAH

Command parameter1:1byte of EPC digits L, introduces word numbers of EPC code

Command parameter2: L\*2byte EPC code, introduces to set read and write protection for which tag.

Command parameter: 1bytemem, choose protection area Mem Bank

|  |  |
| --- | --- |
| 0 | Kill Password |
| 1 | Access Password |
| 2 | EPC number |
| 3 | ID number in TID tag |
| 4 | User |

Command parameter4:1byte controlled word lock

|  |  |
| --- | --- |
| 0 | Can write |
| 1 | Can write permanently |
| 2 | Write with code |
| 3 | Can not write permanently |
| 4 | can read write |
| 5 | Can read write permanently |
| 6 | Read write with code |
| 7 | Can not read write permanently |

Password。0-3is only for EPC, TID and User 3data areas, 4-7 is only for kill password and access password.

Command parameter5:4byte accesspassword, password.

command packet: 『40H 9+L\*2 EAH L EPC mem Lock AccessPasswordCheckSum』

Rebound data: If succeed, leading code in rebound packet is F0H, data part is empty.

『F0H 02H EAH 24H』

## EPC1G2\_Write EPC

Function: write EPC data into tag EPC unit, written data length is in unit of word

Command code: E7H

Command parameter1:1byte EPCdigitsL, introduces word number of EPC code

Command parameter 2: L\*2byte EPC code

Command parameter3:4byte accesspassword

Command packet:『40H 7+L\*2 E7H L EPC AccessPasswordCheckSum』

Rebound packet: ifsucceed, rebound data part is empty

『F0H 02H E7H 27H』

**Note: Accesspassword works only when data area is locked by password.when data not locked, can write none password, if data is forever locked, password is useless.**

## EPC1G2\_Read Protect

Function:operates read protection to specified tag,tag can not read EPC content after success.Only efficient to tag of Philips and UCODE G2XM

Command code: E3H

Command parameter1:4byte AccessPassword

Command parameter2:1byte EPCdigits L, L=1—6, introduces word number of EPC code

Command parameter3: L\*2byte EPC code, introduces to write data for which tag

Command packet:『40H 7+L\*2 E3H AccessPassword L EPC CheckSum』

Rebound packet: If succeed, rebound data part is empty

『F0H 02H E3H 2BH』

## EPC1G2\_RST Read Protect

Function: release from read protection to designated tag,tag can read EPC content after success.Note:there is only one tag in field,works for UCODE G2XM tag of Philips

Command code: E2H

Command parameter1:4 byte AccessPassword

command packet:『40H 06 E2H AccessPasswordCheckSum』

Rebound packet: If succeed, rebound data part is empty

『F0H 02H E2H 2CH』

# Command code summary

## EPC Class1Gen2 command

|  |  |  |
| --- | --- | --- |
| Serial number | Command | Function |
| 1 | EEH | To identify tag ID under radiation range of antenna by mask conditions |
| 2 | EDH | to get listed electronic tag ID from reader memory |
| 3 | ECH | To read block data in appointed data area of appointed tag |
| 4 | EBH | to write data in appointed data area of tag |
| 5 | EAH | Set appointed data area to be write protection in appointed tag |
| 6 | E7H | write EPC data into EPC unit of tag |
| 7 | E5H | reboot Eas state of tag |
| 8 | E4H | EAS SET tag response for alarm |
| 9 | E3H | to do read protection for appointed tag |
| 10 | E2H | To do release read protection for appointed tag |

## ISO18000-6B Command

|  |  |  |
| --- | --- | --- |
| Serial number | Command | Function |
| 1 | FEH | list readable tag ID under the range of antenna radiation |
| 2 | FDH | to read electronic tag ID from memory of reader |
| 3 | FBH | list readable tag ID in the range of antenna radiation according to parameters as follows |
| 4 | F6H | To read block data of appointed tag |
| 5 | F5H | Write data in address of appointed tag |
| 6 | F4H | Set appointed address unit in tag to be write protection |
| 7 | F3H | To read if appointed address of tag write protection |
| 8 | F2H | Write data to appointed address unit of tag |

## Other command

|  |  |  |
| --- | --- | --- |
| Serial number | Command | Function |
| 2 | 02H | get version numbers of hardware and software in reader |
| 4 | 04H | set launch power coefficient in reader |
| 5 | 05H | set frequency tunnel number of microwave signal launched by reader. |
| 6 | 06H | To read operation parameter from last command |
| 7 | 09H | set basic operation parameter of baud rate,launch frequency,output power in reader |
| 9 | 0EH | reader reboot |
| 10 | 13H | Set reader’s auto work mode parameter |
| 11 | 14H | Get reader’s auto work mode parameter |
| 15 | 86H | Set frequency range |
| 16 | 87H | Get frequency range |
| 17 | 88H | Updating program |
| 18 | 8BH | Set reader’s ID |
| 19 | 8CH | Get reader’s ID |
| 20 | 8DH | Set output prefix parameter |
| 21 | 8EH | Get output prefix parameter |

# SDK software development

## Constitution of SDK

VD-67E provide SDK in its package, the SDK constitutes of files as the following:

1. VHDllLib.h file —— DLL mode
2. VHLib.lib file —— SLL model
3. VHHandsetAPI.h file —— API statement of function
4. SDK catalogue —— including the sample program of learning API function.

## Statement of design

### Basic constant and structure

### Definition of constant

| description | statement |
| --- | --- |
| #define ID\_MAX\_SIZE\_64BIT 8 | //RFID Tag ID is 64bit |
| #define ID\_MAX\_SIZE\_96BIT 13 | // RFID Tag ID is 128bit |
| #define MAX\_LABELS 100 | // max of RFID tag will less than 100PCS to be read and write at one time. |

### API function return code

|  | | |
| --- | --- | --- |
| #define \_OK | 0x00 | // operating success |
| //communication error | | |
| #define \_init\_rs232\_err | 0x81 | //communication interface initialization failed |
| #define \_no\_scanner | 0x82 | //can not find the usb reader |
| #define \_comm\_error | 0x83 | //send/receive the data error |
| #define \_baudrate\_error | 0x84 | //set baud rate error |
| // usb reader return back wrong information | | |
| #define \_no\_antenna | 0x01 | //antenna connect failed |
| #define \_no\_label | 0x02 | //can not find the label |
| #define \_invalid\_label | 0x03 | //illegal label |
| #define \_less\_power | 0x04 | //Not enough power to read and write |
| #define \_write\_prot\_error | 0x05 | //write protection area |
| #define \_check\_sum\_error | 0x06 | //check and error |
| #define \_parameter\_error | 0x07 | //parameter error |
| #define \_memory\_error | 0x08 | //data area not existing |
| #define \_password\_error | 0x09 | //wrong password |
| #define \_killpassword\_error | 0x0a | //G2 tag kiss password is 0 |
| #define \_nonlicet\_command | 0x0b | //illegal command |
| #define \_nonlicet\_user | 0x0c | //illegal user that not match the password |
| #define \_invalid\_command | 0x1e | //invalid command, such as wrong parameter command |
| #define \_other\_error | 0x1f | //unknown error |
| //function input error | | |
| #define \_no\_cardID\_input | 0x20 | //other error |

### Definition of data type

Type def USHORT apiReturn; // function return data type

All API function will return a data after it operated. We can judge whether the function operating is success or not through the data(function return code), if it failed, it will show the reason.

### USB reader parameter structure.

**Type def struct tag Reader Date // USB reader time**

{

BYTE Year; //year

BYTE Month; //month

BYTE Day; //day

BYTE Hour; //hour

BYTE Minute; //minute

BYTE Second; //second

}Reader Date;

**Type def struct tag Handset Param**

{

BYTE TagType; //(1)Tag type: 01H－ISO18000-6B，02H－EPCC1，04H－EPCC1G2，08H－EM4442。

BYTE Alarm; //(2)bit0-bit7 bit0:0- no alarm，1-alarm bit1 0-no using white list 1-using whitelist

BYTE OutputMode; //(3)data output mode: 0- save and output directly ,1-save but no output directly ,2- no save but output directly BYTE USB Baud Rate; //(4)USB baud rate: 04H--08H

BYTE Reserve5; //(5)reatain

BYTE Min\_Frequence; //(6)the start and finish line of sending baud rate signal, set to be： 1~63。

BYTE Max\_Frequence; //(7)the start and finish line of sending baud rate signal, set to be ： 1~63。

BYTE Power; //(8)send baud rate data，set to be：0~160。

BYTE RFhrdVer1; //(9)RF module hardware main version

BYTE RFhrdVer2; //(10)RF module hardware second version

BYTE RFSoftVer1; //(11)RF module software main version

BYTE RFSoftVer2; //(12)RF module software second version

BYTE ISTID; //(13)read TID area or not.

BYTE TIDAddr; //(14)the start and finish line of reading the TID area

BYTE TIDLen; //(15)the length of TID area reading

BYTE ISUSER; //(16)read USER area or not

BYTE USERAddr; //(17)the start and finish line of reading the USER area.

BYTE USERLen; //(18)the length of reading USER area.

BYTE Reserve19; //(19) motor vibration,0-no，1-vibrate

BYTE Reserve20; //(20) module type,0---R2000, 1---VM5F

BYTE Reserve21; //(21)restain

BYTE Reserve22; //(22)restain

BYTE Reserve23; //(23)restain

BYTE Reserve24; //(24)restain

BYTE Reserve25; //(25)restain

BYTE Reserve26; //(26)restain

BYTE Reserve27; //(27)restain

BYTE Reserve28; //(28)restain

BYTE Reserve29; //(29)restain

BYTE Reserve30; //(30)restain

BYTE Reserve31; //(31)restain

BYTE Reserve32; //(32)restain

} HandsetParam;

/ reader simulation keyboard parameter

**typedef struct tag SimParam**

{

BYTE DataFormat; //(0)data 2 types：hexzdecimal--0, decimal--1 BYTE DataBank; //(1)EPC area --0, USER area --1，TID area--2

BYTE IsPerfix; //(2)prefix or not，1--yes,0--no BYTE PerfixCode[PERFIXLEN]; //(3-22)prefix data, use 0x00 if it’s not enough BYTE IsEnter; //(23)new line or not, 1--yes,0--no BYTE StartAddress; //(24)the start and finish address of data BYTE DataLen; //(25)length of data

BYTE OutputInterval; //(26)standard output interval (0-255)(0—output no interval between ID.)

}SimParam;

### Control command function

### Connect the USB reader

apiReturn \_stdcall VH\_ConnectScannerUsb(HANDLE \*hScanner);

function : build the connection with the USB reader with the interface.

Suction parameter:

hScanner : usb reader handle

Outlet parameter:

Judge the connection success or fail according to the API return code.

### Break the connection

apiReturn \_stdcall VH\_DisconnectScannerUsb(HANDLE hScanner);

Function : close the connection with the usb reader, and keep the serial port free.

Suction parameter:

hSacnner: USB reader handle

### Check the USB online

typedef int (\_stdcall \*VH\_fDechUsb)(BOOL bDech);

bDech---1 online，or offline

### Inilializate the USB

Could check then input the function code

apiReturn \_stdcall VH\_InitUsb(VH\_fDechUsb fFunc);

### Close the USB

apiReturn \_stdcall VH\_CloseUsb();

### Character time change to be decimal

### Time change to be decimal

int StrtimetoDecimal(CString Command,BYTE \*CmdBuffer);

function: time change to be decimal

Suction parameter:

hSacnner: USB reader handle

**Character change to be decimal**

int strtodecimal(CString Command,BYTE \*CmdBuffer);

function : character change to be decimal

suction parameter:

hSacnner: usb reader handle

### Compare the record pervious and after if they’re equal.

int compare(BYTE \*olddata, BYTE \*data, int len);

Function: compare the record pervious and after if they’re equal.

Suction parameter:

hSacnner: usb reader handle

old data: old data

data: new data

Len：the length for comparing

Return: it means read success if the function return code is \_OK, or will be failed.

**Read the version**

apiReturn \_stdcall VH\_GetVersion(HANDLE hScanner, WORD \*wHardVer, WORD \*wSoftVer);

function: read the usb hardware and software version no.

Suction parameter:

hSacnner: usb reader handle

outlet parameter:

wHardVer: usb reader hardware version no.

WSoftVer: usb reader software version no.

Return: it means read success if the function return code is \_OK, or will be failed.

### Get simulation keyboard module parameter

apiReturn \_stdcall ReadSimParam(HANDLE hScanner, SimParam \* pParam);

function name: ReadSimParam

function: read the parameter of module.

Input parameter: hScanner is the communication handle.

Output parameter: pParam is the guide for structure.

Return data: return 0 means success, or failed.

### Set simulation keyboard module parameter

apiReturn \_stdcall WriteSimParam(HANDLE hScanner, SimParam \* pParam);

function: input the small module parameter.

Input parameter: hScanner is communication handle ,pParam is the guide for structure

Output parameter:

1. Return data: return 0 means success, or failed.

### Get the usb reader basic working parameter

apiReturn \_stdcall ReadHanderParam(HANDLE hScanner, HandsetParam \* pParam);

function: read the previous command and write it to the working parameter.

Suction parameter:

hSacnner: usb reader handle

outlet parameter:

pParam: return to the usb reader working parameter, 32 bytes

1. Return data: return 0 means success, or failed.

### Set the USB reader basic working parameter

apiReturn \_stdcall WriteHanderParam(HANDLE hScanner, HandsetParam \* pParam);

function: set usb reader working parameter

Suction parameter:

hSacnner: usb reader handle

pParam: usb reader working parameter, 32bytes.

Return: it means read success if the function return code is \_OK, or will be failed。

### Restore the factory param of usb reader…

apiReturn \_stdcall ReadFactoryParameter(HANDLE hScanner);

function: restore the factory param of usb reader

Suction parameter:

hSacnner:

Restore the factory param of usb reader…

Return: it means read success if the function return code is \_OK, or will be failed

### Get the usb reader ID.

apiReturn \_stdcall GetHandsetID(HANDLE hScanner, BYTE \*HandsetID);

function: usb reader will send the ID. to the server when it got the command.

Suction parameter:

hSacnner: usb reader handle

outlet parameter:

HandsetID: usb reader ID.

1. Return: it means read success if the function return code is \_OK, or will be failed.

### Read and write ISO18000-6C function

### Indentify ISO18000-6C tag EPC NO.

apiReturn \_stdcall EPC1G2\_ReadLabelID(HANDLE hScanner, BYTE mem, int ptr, BYTE len, BYTE \*mask, BYTE \*IDBuffer, int \*nCounter,int Address);

function: Read all the identify tag EPC no. in the range of antenna.

Suction parameter:

hSacnner:

mem: choose the data area;

|  |  |
| --- | --- |
| 0 | Password area |
| 1 | EPC no. |
| 2 | TID no. |
| 3 | User area |

ptr: address of Mask code(unit: Bit)

len: length of Mask code(unit: Bit)

mask: (unit :Byte)，if len/8 is integer，the mask length is len/8；if len/8 is not integer, the mask length is len/8+1, the last bytes of the Mask code put upper bit and the lower bit with zero.

Address: reader RS485 net address，Address =0 stands no net.

outlet parameter:

IDBuffer: the tag EPC NO. read.

NCounter: the tag qty read.

Return: it means read success if the function return code is \_OK, or will be failed.

**Notice: LEN=0 stands all the identify tag ID. in the range of antenna.**

### Read data in an area.

apiReturn \_stdcall EPC1G2\_ReadWordBlock(HANDLE hScanner, BYTE EPC\_WORD, BYTE \*IDBuffer, BYTE mem, BYTE ptr, BYTE len, BYTE \*Data, BYTE \*AccessPassword,int Address);

function: read the data of tag in consecutive address memory.

Suction parameter:

hSacnner:

EPC\_WORD:EPC Length(unit:Word)；such as 96BitsEPC L=6(Words)；

IDBuffer: the EPC no. chosen.

mem: the data area chosen.; 0-password area，1-EPC no.，2-TID no.，3-user area.

ptr: Read the start and finish address(unit:WORD)

len: read length(unit:WORD)

AccessPassword: 4 bytes

Address: reader RS485 net address，Address =0 no net.

outlet parameter:

Data: data to be read

Return: it means read success if the function return code is \_OK, or will be failed.

**Notice: AccessPassword have effect on user area that it’s have password.**

### Write a range of area

apiReturn \_stdcall EPC1G2\_WriteWordBlock(HANDLE hScanner, BYTE EPC\_WORD, BYTE \*IDBuffer, BYTE mem, BYTE ptr, BYTE len, BYTE \*Data, BYTE \*AccessPassword,int Address);

function: write data into the appointed address of the tag.

Suction parameter:

hSacnner:

EPC\_WORD: EPC length L(unit:Word)；like 96BitsEPC Length =6(Words)；

IDBuffer: the EPC no. chosen

mem: choose the data area;

|  |  |
| --- | --- |
| 0 | Password area |
| 1 | EPC no. |
| 2 | TID tag ID no. |
| 3 | User |

ptr: the start address of writing(unit:WORD)

len: the length of writing(unit:WORD)

Data: the data of writing

Access Password: 4 bytes Access Password

Address: reader RS485 net address, Address =0 means no net.

Return: it means read success if the function return code is \_OK, or will be failed

**Notice: Access Password have effect on user area that’s in locked. If the data area is unlocked, it could be write without the password, and if the data is permanent locked, the password is useless.**

### Set the protection while reading and writing

apiReturn \_stdcall EPC1G2\_SetLock(HANDLE hScanner, BYTE EPC\_WORD, BYTE \*IDBuffer, BYTE mem, BYTE Lock, BYTE \*AccessPassword,int Address);

function: set writing protection for the appointed data from the appointed tag.

Suction parameter:

hSacnner:

EPC\_WORD: EPC length L(unit:Word)； like 96Bits EPC length L=6(Words)；

IDBuffer: the EPC no. of tag chosen

mem: choose the data area.

|  |  |
| --- | --- |
| 0 | Kill Password |
| 1 | Access Password |
| 2 | EPC no. |
| 3 | TID tag ID no. |
| 4 | User |

Lock: control button Lock。

|  |  |
| --- | --- |
| 0 | Write available |
| 1 | Write available permanent |
| 2 | Write with password |
| 3 | No-write permanent |
| 4 | Read and write available |
| 5 | Read and write permanent |
| 6 | Read and write with password |
| 7 | No-read and write permanent |

Notice: 0～3 only work for EPC、TID and User area; 4～7 only work for Kill Password and Access Password。

Access Password: 4 bytes Access Password.

Address: reader RS485 net address, Address =0 means no net.

Return: it means read success if the function return code is \_OK, or will be failed.

### Permanent dormancy tag

apiReturn \_stdcall EPC1G2\_KillTag(HANDLE hScanner, BYTE EPC\_WORD, BYTE \*IDBuffer, BYTE \*KillPassword,int Address);

function: permanent dormancy tag so the tag will not be read later.

Suction parameter:

hSacnner:

EPC\_WORD:EPC length L(unit:Word)；like 96Bits EPC length L=6(Words)；

IDBuffer: EPC no. chosen

KillPassword: 4 bytes KillPassword

Address: reader RS485 net address, Address =0 means no net.

Return: it means read success if the function return code is \_OK, or will be failed

**EAS mode operating command**

apiReturn \_stdcall EPC1G2\_ChangeEas(HANDLE hScanner, BYTE EPC\_WORD, BYTE \*IDBuffer, BYTE State, BYTE \*AccessPassword,int Address);

function: reset or set the EAS of tag. Only have effect on **NXP UCODE EPC G2 tag.**

Suction parameter:

hSacnner:

EPC\_WORD:EPC length L(unit :Word)；like 96BitsEPC length L=6(Words)；

IDBuffer: EPC no. chosen

State: Alarm mode

|  |  |
| --- | --- |
| 0 | No alarm |
| 1 | alarm |

Access Password: 4 bytes Access Password, must write no matter if the tag visit password is 0 or not.

Address: reader RS485 net address, Address =0 means no net.

Return: it means read success if the function return code is \_OK, or will be failed.

### EAS Alarm command

apiReturn \_stdcall EPC1G2\_EasAlarm(HANDLE hScanner,int Address);

function: EAS set tag should alarm to test the command. Only have effect on **NXP UCODE EPC G2 tag.**

Suction parameter:

hSacnner: reader communication handhle.

Address: reader RS485 net address, Address =0 means no net.

Return: it means have tag alarm if the function return code is \_OK, or will be no tag alarm.

### Set reading protection

apiReturn \_stdcall EPC1G2\_ReadProtect(HANDLE hScanner,BYTE \*AccessPassword, BYTE EPC\_WORD, BYTE \*IDBuffer,int Address);

function: set reading protection for appointed tag, then it will not be ok to read the EPC of the tag. Only have effect on **NXP UCODE G2XM.**

Suction parameter:

hSacnner: reader communication handle

Access Password: tag visit password

EPC\_WORD: state the word amount of EPC NO.

IDBuffer: EPC no. state write to which tag.

Address: reader RS485 net address, Address =0 means no net.

Return: it means read success if the function return code is \_OK, or will be failed.

### Relieve reading protection

apiReturn \_stdcall EPC1G2\_RStreadProtect(HANDLE hScanner, BYTE \*AccessPassword,int Address);

function: relieve reading protection for the appointedtag, it could be read the EPC if succeed. Notice: only could have one tag, and only have effect on the N**XP UCODE G2XM.**

Suction parameter:

hSacnner: reader communication handle

Access Password: tag visit password

Address: reader RS485 net address, Address =0 means no net.

Return: it means relieve success if the function return code is\_OK, or will be failed.

### Read and Write ISO18000-6B function

### Identify tag ID. no.

apiReturn \_stdcall ISO6B\_ReadLabelID(HANDLE hScanner, BYTE \*IDBuffer, int \*nCounter,int Address) ;

function: read all the tag id no. within the scope of radiation of antenna.

Suction parameter:

hSacnner: reader communication handle

outlet parameter:

nCounter: return the real id no. amount that be read.

IDBuffer: the tag id no. save capacity of reading.

Address: reader RS485 net address, Address =0 means no net.

Return: it means relieve success if the function return code is\_OK, or will be failed

**Identify appointed tag id no.**

apiReturn \_stdcall ISO6B\_ListSelectedID(HANDLE hScanner, int Cmd, int ptr, BYTE Mask, BYTE \*Data, BYTE \*IDBuffer, int \*nCounter,int Address) ;

function: read the appointed tag id no. within the scope of radiation of antenna.

Suction parameter:

hSacnner: reader communication handle

Cmd: conditions of tag chosen

|  |  |
| --- | --- |
| 00 | Equal to |
| 01 | Unequal to |
| 02 | Greater than |
| 03 | Less than |

ptr: tag data start address; data range 0～223

Mask: data mask; the bytes each bit equal to a compared byte.

0 stands that byte will not take part in comparison; 1 stands that byte will take part in comparison.

Data: comparison data

Address: reader RS485 net address, Address =0 means no net.

outlet parameter:

nCounter: return real tag id no. read.

IDBuffer: read the appointed tag id no. within the scope of radiation of antenna.

Return: it means relieve success if the function return code is\_OK, or will be failed

### Write data module

apiReturn \_stdcall ISO6B\_WriteByteBlock(HANDLE hScanner, BYTE \*IDBuffer, BYTE ptr, BYTE len, BYTE \*Data,int Address) ;

function: write data at appointed tag address.

Suction parameter:

hSacnner: reader communication handle

IDBuffer: ID no. want to write.

ptr: the start address of tag capacity(8～223)

len: data module length, means write the numbers of word at one time. (4Bytes/word)

Data: data want to write

**Notice: ptr must be the integral multiple of 4. (nAddress+nLen) ≤223。**

Address: reader RS485 net address, Address =0 means no net.

Return: it means relieve success if the function return code is\_OK, or will be failed.

### Read data moudle

apiReturn ISO6B\_ReadByteBlock(HANDLE hScanner, BYTE \*IDBuffer, BYTE ptr, BYTE len,BYTE \*Data,int Address)

function: read the data in a tag serial address.

Suction parameter:

hSacnner: reader communication handle

IDBuffer: tag id no. want to read.

ptr: read start address of tag capacity(0～223 Byte)

len: data length, means read bytes at one time(Byte)

Address: Reader RS485 net address, Address =0 stands no net.

outlet parameter:

Data, return the data read.

**Notice: nLen must ≤ 32。 (nAddress+nLen) ≤223。**

Return: it means relieve success if the function return code is\_OK, or will be failed.

### Write slowly data moudle

apiReturn \_stdcall ISO6B\_WriteAByte(HANDLE hScanner, BYTE \*IDBuffer, BYTE ptr, BYTE len, BYTE \*Data,int Address);

function: write data at tag appointed address byte by byte.

Suction parameter:

hSacnner: reader communication handle.

IDBuffer: ID. no. want to write.

ptr: start address of writing capacity.(8～223)

len: data module length, means the numbers of words write at one time.(4Bytes/word)

Data: data want to be write.

**Notice:(nAddress+nLen) ≤223。**

Address: Reader RS485 net address, Address =0 stands no net.

Return: it means relieve success if the function return code is\_OK, or will be failed.

Notice: the command adopt data writing with byte by byte, it’s slowly. Only used in the tag do not support a pervious writing command situation.

### Set the protection module

apiReturn \_stdcall ISO6B\_WriteProtect(HANDLE hScanner, BYTE \*IDBuffer, BYTE ptr,int Address);

function: set protection on the appointed address of appointed tag.

Suction parameter:

hSacnner: reader communication handle

IDBuffer: tag id no. want to write.

ptr: tag capacity address that will write protection.(8～223)

Address: Reader RS485 net address, Address =0 stands no net.

Return: it means relieve success if the function return code is\_OK, or will be failed.

### Read and write protection moudle

apiReturn \_stdcall ISO6B\_ReadWriteProtect(HANDLE hScanner, BYTE \*IDBuffer, BYTE ptr, BYTE \*Protected,int Address);

function: read the appointed address of appointed tag whether write protection or not.

Suction parameter:

hSacnner: reader communication handle

IDBuffer: tag id no. want to write.

ptr: the address of protection module tag capacity.(0～223)

Address: Reader RS485 net address, Address =0 stands no net

outlet parameter:

Protected: protection module, 0-no protection, 1-protected already

Return: it means relieve success if the function return code is\_OK, or will be failed.

[8 Tag saver and the notification 43](#_Toc388560723)

# Electronic tag storage area and notes

Memory Bank of EPC Class1 Gen2 tag divides to be 4 areas.

* + EPC area (EPC): Storage area of EPC code leaves 96 Bits EPC code at most for now, can read and write.
  + TID area: Keep sett ID numbers by tag manufacturer, there are 32 and 64Bites two kinds for now, can read, and can not write.
  + User: different area for different manufacturer, G2 tag of Impinj company has no user.NXP company has 96 Bits, canread, can write.
  + Password: has 32Bits access password, and 32 bits kill password. Can read, can write, can make different protection for these two areas.

EPC Class1 Gen2tag can set different protection mode for different storage area, protection mode of each storage area is of 4 types:

* + EPC, TID and user of G2 tag.

Read in EPC, TID and user area of G2tag is not protected, write protection function

Writeable from any state---can write noneaccesspassword,can set password lock or permanent write or permanent lock.

Permanently writable---can wite none accesspassword,andcan not be passwordlocked or permanent locked.

Writable from secured state---can write in the case of know accesspassword

Never writable---can not write even know password.

Read and write in password area of G2 tag can be protected, read-write protection state in password area does not affect usage of password,and can put protection function to these 2 areas.

Readable and Writeable from any state---can read and write none accesspassword, can be password secured or permanent read write or permanent secured.

Can read and write none accesspassword, andcan not be password secured later.

In the case of knowing password, can read and revise password, can set to be permanent secured or permanent read-write later

Can not read or revise password, even know it.

**Note: set tag read and write protection,must know tag accesspassword.**

1. Memory bank in the tag of ISO-18000-6B divides to two area, storage capacity inside is 2048bits, and divides to be 256 byte.there is one address for each byte, 0-255 in correspondence.
   * Address 0-7 eight byte (64bits): is tag ID numbers, solidfy before products come out, can not be revised.
   * Address 8-233 user information can be left in user aera,can self-distributed according to details,can be revised and locked,but can not revise once loced and unlocked.

Address224-255 writes protection byte

9. SCHEDULE

9.1 Parameter statements…

typedefstructtagReaderBasicParam

{

BYTE BaudRate; //(1)baud rate of serial, value：00H~08H.i.e. the rate with "set the baud rate" command

BYTE Power; //(2)RF power output, value：20~30dBm.(0-63)

BYTE Min\_Frequence; //(3)start point of transmitting microwave signal frequency, value： 1~63.

BYTE Max\_Frequence; //(4)end point of transmitting microwave signal frequency, value： 1~63.

BYTE Reserve5; //(5)reserve, changed into modulation depth later.

BYTE WorkMode; //(6)work mode of reader：0-Auto, 1-Command

BYTE ReaderAddress; //(7)RS485 address:0--255

BYTE NumofCard; //(8)max tags of once reading.

BYTE TagType; //(9)type of tag：01H－ISO18000-6B，02H－EPCC1，04H－EPCC1G2，08H－EM4442.

BYTE ReadDuration; //(10)tag reading duration time：RF emission duration time is only effective for EM tag; 0－10ms，1－20ms，2－30ms，3－40ms.

BYTE ReadTimes; //(11)read times M：reader will execute the command for M times when receiving reading command from host computer.

BYTE EnableBuzzer; //(12)1:enable buzzer 0:disable buzzer

BYTE IP1; //(13)IP address of reader

BYTE IP2; //(14)

BYTE IP3; //(15)

BYTE IP4; //(16)

BYTE Port1; //(17)high-order of reader port

BYTE Port2; //(18)

BYTE Mask1; //(19)reader mask1

BYTE Mask2; //(20)reader mask2

BYTE Mask3; //(21)reader mask3

BYTE Mask4; //(22)reader mask4

BYTE Gateway1; //(23)reader address gateway

BYTE Gateway2; //(24)

BYTE Gateway3; //(25)

BYTE Gateway4; //(26)

BYTE MAC1; //(27)MAC address of reader

BYTE MAC2; //(28)

BYTE MAC3; //(29)

BYTE MAC4; //(30)

BYTE MAC5; //(31)

BYTE MAC6; //(32)

}ReaderBasicParam;

//Auto parameter of reader

typedefstructtagReaderAutoParam

{

BYTE AutoMode; //(1)tag reading mode：0-timing, 1-trigger.

BYTE TimeH; //(2)tag storage time: unit: second. Default 1.

BYTE TimeL; //(3)

BYTE Interval; //(4)0-10ms，1-20ms，2-30ms，3-50ms，4-100ms. Default 2. Auto reading tag once at intervals.

BYTE NumH; //(5)tag storage quantity：default 1. The quantity of read tag ID stored in reader memory.

BYTE NumL; //(6)

BYTE OutputManner; //(7)data output format：0-terse，1-standard，2-XML. Default 0.

BYTE OutInterface; //(8)output interface：0－RS232，1－RS485，2－RJ45. Default 0. 3- Wiegand26 4- Wiegand34

BYTE WiegandWidth; //(9)value of Weigand pulse width.

BYTE WiegandInterval; //(10)value of Weigand pulse interval.

BYTE ID\_Start; //(11)start address of output ID, value 0～4.

BYTE IDPosition; //(12)storage address for tag ID in tag.

BYTE Report\_Interval; //(13) report interval：unit is second. Default 1. Automatically notify host pc once at intervals.

BYTE Report\_Condition; //(14)condition of report：default 1. 0-notify now，1-timing，2-add，3-remove，4-change

BYTE Report\_Output; //(15)report output port

BYTE Antenna; //(16)select antenna.1-ant1,2-ant2,4-ant4,8-ant8

BYTE TriggerMode; //(17)trigger mode(default0): 0-low level 1-high level

BYTE HostIP1; //(18)notified IP address

BYTE HostIP2; //(19)

BYTE HostIP3; //(20)

BYTE HostIP4; //(21)

BYTE Port1; //(22)notified port

BYTE Port2; //(23)

BYTE Reserve24; //(24)notified MAC,mofi by mqs 20121207 reserve

BYTE ArgentinaSim; //(25)//emulation mode(argentina),0—non-emulation，1--emulation

BYTE CardTime1; //(26)//reading time-out 1

BYTE CardTime2; //(27)//reading time-out 2

BYTE ArgentinaMode; //(28)//5 modes for argentina, 0---Only ATA ; 1---Only EPC; 2---Only EPC & TID; 3---ATA + EPC; 4---ATA + EPC & TID.

BYTE Alarm; //(30)0-no alarm，1-alarm. To detect if alarm in timing and trigger modes.

BYTE Reserve31; //(31)time interval for standard output，default value is 120s，1～255.

BYTE EnableRelay; //(32)to control relay or not in Auto mode 1:control 0: no control

}ReaderAutoParam;

#define PERFIXLEN 20 //The length of the prefix value// Reader’s simulation key parameters

typedefstructtagSimParam

{

BYTE DataFormat; //(0)two kind data format ：hexadecimal --0, decimalism—1

The data format has two kinds: hexadecimal - 0, decimal - 1

BYTE DataBank; //(1)EPCarea--0, USER area--1，TID area--2

BYTE IsPerfix; //(2)Whether with the prefix, 1--with,0--without

BYTE PerfixCode[PERFIXLEN]; //(3-22)Perfix value, if not enough,fill with 0x00.

BYTE IsEnter; //(23)whether with enther and wrap,1--with,0—without.

BYTE StartAddress; //(24) data’s initial address BYTE DataLen; //(25)data’s length

BYTE OutputInterval; //(26)Standard output interval(0-255)(0—no interval between output ID)

}SimParam;

typedefstruct \_tagReaderFreq

{

char \*chFreq; //frequency strings of different countries

int iGrade; //grade = 50;

intiSkip; //skip = 500KHz;

intdwFreq; //starting frequency = 902750;

// formula: 902750 + grade\*skip

}tagReaderFreq;

Verifyalgorithm

unsigned char SetChecksum(unsigned char \*buf,unsignedint length)

{

unsigned int i;

unsigned char sum = 0;

for(i=0; i < length; i++)

{

sum += buf[i];

}

sum = ~sum + 1;

return sum;