



**I-PICK**  
**868 MHz RFID Handheld**  
**Software Development Manual**

iDTRONIC GmbH  
Donnersbergweg 1  
67059 Ludwigshafen  
Germany/Deutschland

Phone: +49 621 6690094-0  
Fax: +49 621 6690094-9  
E-Mail: [info@idtronic.de](mailto:info@idtronic.de)  
Web: [idtronic-rfid.com](http://idtronic-rfid.com)

Issue 0.1  
– 18. February 2016 –

Subject to alteration without prior notice.  
© Copyright iDTRONIC GmbH 2016  
Printed in Germany

## Contents

<b>1</b>	<b>Serial port bidirectional communication protocol .....</b>	<b>5</b>
1.1	Summary .....	5
1.1.1	Command packet format .....	5
1.1.2	Return packet format .....	5
1.1.3	Error code .....	6
1.1.4	Example .....	6
1.2	Serial port control command format .....	6
1.2.1	Read version number .....	6
1.2.2	Add name list .....	7
1.2.3	Delete name.....	7
1.2.4	Get List .....	8
1.2.5	Getting work parameter .....	8
1.2.6	set up Label filter .....	10
1.2.7	Getting Tag filter parameter .....	10
1.2.8	Set work parameter .....	11
1.2.9	Factory Reset .....	12
1.2.10	Set current time .....	12
1.2.11	Get current time .....	13
1.2.12	Get record in specific time from SD card .....	13
1.2.13	Delete all records in SD card .....	14
1.2.14	Set handheld Bluetooth name .....	14
1.2.15	Get handheld Bluetooth name .....	15
1.3	Checksum computing method(C language).....	15
<b>2</b>	<b>SDK Software Development .....</b>	<b>16</b>
2.1	SDK Component .....	16
2.2	Design Description .....	16
2.2.1	Basic Constant and Structure.....	16
2.2.1.1	Definition of the Constant .....	16
2.2.1.2	API function return code.....	16
2.2.1.3	Data Mode Definition .....	17
2.2.1.4	Parameter structure of handheld reader .....	17
2.2.2	Control Command Function.....	18
2.2.2.1	connect to handheld reader .....	18
2.2.2.2	Disconnect .....	18
2.2.2.3	Test if the USB is on .....	18
2.2.2.4	Initialize USB .....	18
2.2.2.5	Close USB .....	18
2.2.2.6	Transmission Mode Settings .....	19
2.2.2.7	Character and time switch to metric system .....	19
2.2.2.8	Compare if the two data are equal .....	19
2.2.2.9	Get Version .....	20
2.2.2.10	Add label ID.....	20
2.2.2.11	Delete Label ID.....	20
2.2.2.12	Get Label ID.....	20
2.2.2.13	Submit the list.....	21
2.2.2.14	Get the basic working parameter of the handheld reader .....	21

2.2.2.15	Setting the basic working parameter of the handheld reader.....	21
2.2.2.16	Reset to the factory parameter of the handheld reader .....	22
2.2.2.17	Setting Time.....	22
2.2.2.18	Get the time.....	22
2.2.2.19	Setting the tag filter .....	22
2.2.2.20	Get the tag filter .....	23
2.2.2.21	Get the record .....	23
2.2.2.22	Delete all record .....	24
2.2.2.23	Get handheld reader ID .....	24
2.2.3	Read & Write ISO18000-6C Function .....	24
2.2.3.1	Read ISO18000-6C tag's EPC number .....	24
2.2.3.2	Read a Block of data .....	25
2.2.3.3	Write a Block of data .....	26
2.2.3.4	Setting protection mode for read & write .....	26
<b>3</b>	<b>Appendix .....</b>	<b>28</b>
3.1	Document Revisions .....	28

## 1 Serial port bidirectional communication protocol

Application development methods including:

Direct operation on reader using control code of serial port communication protocol.(Bluetooth and USB are both use Serial port communication)

### 1.1 Summary

In application system, handheld is connected with control board( PC) via RS232, It receive the command from control board , and return result of executing command to control board. So , the command packet is defined as when packet from control board to handheld; Return packet is defined as when packet from handheld to control board

#### 1.1.1 Command packet format

BootCode	Length	Command	Command Param	CheckSum
1 Byte	1 Byte	1 Byte	N Byte	1 Byte

As shown above, command packet includes 5 parts:

**Boot Code:** 1byte, fixed as 40H

**Length:** 1byte, This length is the total bytes of last 3 parts.

**Command:** 1byte

**Command parameter** its length will change follow command.

**Check Sum:** 1byte, byte complement at end

#### 1.1.2 Return packet format

BootCode	Length	Command	Return Data	CheckSum
1 Byte	1 Byte	1 Byte	N Byte	1 Byte

As shown above , return packet includes 5 parts:

**Boot Code:** 1byte. when command execute correct, bootcode of return packet will be F0H; when command execute fail, bootcode of return packet will be F4H.

**Length:** 1byte. This length is the total bytes of Last 3 parts.

**Command:** 1byte, if same as received command code, means execute correct

**Return Data:** return the result of executing command, its length will change follow command.

**Check Sum:** 1byte, byte complement at end

### 1.1.3 Error code

When command execute fail, the Bootcode is F4H, Return Data is 1byte error code. Common error code including:

00H is command success, 10H~1FH is invalid command, More than 20H means command exec error.

00(00H)	Command correct or test correctly
16(10H)	Invalid command
17(11H)	Error parameter
18(12H)	Error checkSum
32(20H)	Command exec error

### 1.1.4 Example

**For example:** Getting reader's current time on command packet is: [40H 02H 12H ACH]

40H	bootcode
02H	Effective length is 2bytes
12H	Command code of[get reader's current time]
null	No command parameter
ACH	Check sum

That's complement code of 40H+02H+12H=54H(against increasing1)

If current time is year 2012 month 01 date 01 hour 9 minute 01 second 01

When execute correct, return packet is : [F0H 08H 12H 0CH 01H 01H 09H 01H 01H DDH]

When execute error, return packet may is: [F4H 03H 12H 10H E7H]

## 1.2 Serial port control command format

### 1.2.1 Read version number

**Function:** To get VH-75handheld hardware, software version number **Command code:** 02H

**Command parameter** null

**Command packet:** [40H 02H 12H AC]

**Return Data:** if successful, return packet will be as below

[F0H 06H 02H Byte0-Byte3 Check]

Parameter specification :

Byte0	MAIN HARDWARE VERSION
Byte1	Second Hardware VERSION
Byte2	Main software version
Byte3	Second Software VERSION

### 1.2.2 Add name list

**Function:** Add a new list based on the original list of VH-75 handset.

**Command code:** 03H

**Command parameter** M DATA

**Command packet:**[40H LEN 03H M [Len EPC]...[Len EPC] Check]

**Return data:** if successful, return packet will be as below

[F0H 02H 03H Check]

**Command parameter specification:**

parameter	LENGTH (Byte)	Introductions
M	1	indicate increase total number of list ,M<=8
Len	1	The length of EPC code
EPC	Len*2	EPC code

### 1.2.3 Delete name

**Function:** Delete list based on the original list of VH-75 handset

**Command:** 04H

**Command parameter** M [Len EPC]

**Command packet:** if successful, return packet define as following:

[40H LEN 04H M [Len EPC]...[Len EPC] Check]

**Return packet parameter:**[F0H 02H 04H Check]

**Command parameter specification:**

parameter	length (Byte)	Introductions
M	1	indicate delete total number of list ,M<=8
Len	1	The length of EPC code
EPC	Len*2	EPC code

#### 1.2.4 Get List

**Function:**To get list from VH-75 handheld

**Command:** 05H

**Command parameter** SADDR M

**Command packet:**[40H 05H 05H SADDR M Check]

**Command parameter specification :**

parameter	LENGTH (Byte)	INTRODUCTIONS
SADDR	1	Start address of getting list
M	1	indicate total number of list in this action ,M<=8

**Return data:** if successful, return packet will be as below

[F0H LEN 05H N [Len EPC]...[Len EPC] Check]

**Return data parameter specification :**

parameter	LENGTH (Byte)	INTRODUCTIONS
N	1	indicate get total number of list ,N<=8
Len	1	EPC code length
EPC	Len*2	EPC code

#### 1.2.5 Getting work parameter

**Function:** To get VH-75 Handheld work parameter

**Command:** 06H

**Command parameter** null

**Command packet:**[40H 02H 06H B8H]

**Return Data:** if successful, return packet will be as below

[F0H 22H 06H PAM Check]



**Command parameter specification:**

BYTES	DEFINE										
0	tag types :01H-02H 18000 -18000,04H-ISO										
1	Alarm based on list : 0 - No Alarm ,1 - Alarm										
2	tag output way: 0-Save and output directly,1- save but not outout,2-not save but output directly .										
3	<div> <div>USB port baud rate, default :07—57600bps</div> <table border="1"> <tbody> <tr> <td>04H</td><td>9600bps</td></tr> <tr> <td>05H</td><td>19200bps</td></tr> <tr> <td>06H</td><td>38400bps</td></tr> <tr> <td>07H</td><td>57600bps</td></tr> <tr> <td>08H</td><td>115200bps</td></tr> </tbody> </table> </div>	04H	9600bps	05H	19200bps	06H	38400bps	07H	57600bps	08H	115200bps
04H	9600bps										
05H	19200bps										
06H	38400bps										
07H	57600bps										
08H	115200bps										
4	reserve										
5	Min carrier frequency,1-63,Refer to frequency list										
6	Max carrier frequency ,1-63, Refer to frequency list										
7	RF Output power 0-63										
8	Byte0 Main hardware version number (only read parameters Low Level software settings)										
9	Byte1 Second hardware version number (only read parameters Low Level settings)										
10	Byte2 Main software major version number (only read parameters Low Level software settings)										
11	Byte3 Second software version number (only read parameters Low Level software settings)										
12	Whether reading TID area										
13	Start ADD on Read TID										
14	Reading TID length										
15	Whether reading USER area										
16	Start ADD Read USER area										
17	read USER area length										
18	vibration,0-NO, 1-Vibrate										
19	UHF module type,0---R2000, 1---VM5E										
20	reserve										
21	reserve										
22	reserve										
23	reserve										
24	reserve										
25	reserve										
26	reserve										
27	reserve										
28	reserve										
29	reserve										
30	reserve										
31	reserve										

1.2.6 set up Label filter

**Function:** output the list only when the tag ID accord with filter setting

**Command:** 07H

**Command parameter** ADDR Len M

**Command packet:**[40H LEN 07H ADDR Len M Check]

**Return Data:** if successful, of return packet will be as below

[F0H 02H 07H Check]

**Command parameter specification:**

parameter	LENGTH	INTRODUCTIONS
ADDR	2	MAC address , high back, Bit address
Len	2	MAC length , Bit length .
M	M	high in the front , low integer , M=Len/8 ; if Le integer , Last byte choose the Max bit, ,low padded with zeros.

When LEN =0, filters no use , without command parameter M

**Parameter packet:**[40H 04H 07H 00 Check]filter object:

ISO18000	— 6B	64bits UID
ISO18000	— 6C	EPC code

1.2.7 Getting Tag filter parameter

**Function:** Read filter parameter

**Command:** 08H

**Command parameter** null

**Command packet:** [40H 02H 08H Check]

**Return Data:** if successful, return packet will be as below

[F0H LEN 08H ADDR Len M Check]

**Return parameter specification :**

parameter	LENGTH	INTRODUCTIONS
ADDR	2	MAC address , high back, Bit address
Len	2	MAC length , Bit length .
M	M	high in the front , low integer , M=Len/8 ; if Le integer Last byte 8H choose the Max bit ,low padded with zeros.

### 1.2.8 Set work parameter

**Function:** Set VH-75 Handset work parameter

**Command:** 09H

**Command parameter** null

**Command packet:**[40H 22H 09H PAM Check]

**Return Data:** if successful, return packet will be as below

[F0H 02H 09H Check]

**Command parameter specification:**

BYTES	DEFINE										
0	Tag type : 04H -ISO18000 , 05H -EPC1, 04H -ISO										
1	Whether Alarm based on alarm: 0 - No Alarm, 1 – Alarm										
2	tag output way: 0-Save and output directly,1- save but not outout,2-not save but output directly .										
3	USB port baud rate, default 07—57600bps <table border="1"> <tr> <td>04H</td><td>9600bps</td></tr> <tr> <td>05H</td><td>19200bps</td></tr> <tr> <td>06H</td><td>38400bps</td></tr> <tr> <td>07H</td><td>57600bps</td></tr> <tr> <td>08H</td><td>115200bps</td></tr> </table>	04H	9600bps	05H	19200bps	06H	38400bps	07H	57600bps	08H	115200bps
04H	9600bps										
05H	19200bps										
06H	38400bps										
07H	57600bps										
08H	115200bps										
4	reserve										
5	Min carrier frequency,1-63,Refer to frequency list										
6	Max carrier frequency,1-63,Refer to frequency list										
7	RF Output power 0-160W										
8	Byte0 Main hardware version number (only read parameters Low Level software settings)										
9	Byte1 Second hardware version number (only read parameters Low Level settings)										
10	Byte2 Main software major version number (only read parameters Low Level software settings)										
11	Byte3 Second software version number (only read parameters Low Level software settings)										

12	Whether reading TID area
13	Start ADD on Read TID
14	Reading TID length
15	Whether reading USER area
16	Start ADD Read USER area
17	read USER area length
18	Vibration,0-NO Vibrate
19	UHF module type,0---R2000, 1---VM5E
20	reserve
21	reserve
22	reserve
23	reserve
24	reserve
25	reserve
26	reserve
27	reserve
28	reserve
29	reserve
30	reserve
31	reserve

### 1.2.9 Factory Reset

**Function:** VH-75 handset restart to factory-reset parameters.

**Command:** 0DH

**Command parameter** null

**Command packet:** [40H 02H 0DH Check]

**Return Data:** if successful, return packet will be as below

[F0H 02H 0DH Check]

### 1.2.10 Set current time

[F0H 02H 11H FDH]

**Function:** set reader's current time.

**Command code:** 11H

**Command parameter** 6bytes: YY/MM/DD/hh/mm/ss

**Command packet:** [40H 08H 11H YY MM DD hh mm ss CheckSum]

**Return Data:** if successful, return packet will be null.

[F0H 02H 11H FDH]

### 1.2.11 Get current time

**Function:** getting current time of reader

**Command code:** 12H

**Command parameter** null

**Command packet:**[40H 02H 12H AC]

**Return Data:** if successful, return packet will be YY/MM/DD/hh/mm/ss:

YY	Year, select last 2 digits of current year. Such as year 2011 will be 11H.
MM	month January will be 01
DD	Date
hh	Hour
mm	Minute
ss	second

[F0H 08H 12H YY MM DD hh mm ss CheckSum]

### 1.2.12 Get record in specific time from SD card

**Function:** getting tag's record in specific time from SD card

**Command code:**16H

**Command parameter**

6bytes, indicates starting time: yy/mm/dd/hh/mm/ss:

6bytes, indicates stop time: yy/mm/dd/hh/mm/ss

**Command packet:**[40H 0EH 16H StartTime EndTime CheckSum]

**Return Data:** if successful, return packet define as following:

[F0H L 16H Sum [Data time ID] CheckSum]

**Return packet parameter description:**

<b>L</b>	1BYTE, DATA LENGTH OF COMMAND PACKET
<b>Sum</b>	1byte, total record
<b>Data</b>	Return packet parameter description as below

Data :

BYTE	NAME	CONTENT
0-5	DateTime	6byte, time of reading tag,YY MM DD hh mm ss
6	TagType	1byte,tag type :1-ISO180006B 4-ISO180006C
7	EPCLen	1byte,EPC code or UID code length
8-23	EPC/UID	16 byte,EPC code or UID code, balance byte fill 0
24	TIDLen	1byte,if 6C tag is TID code length, If 6B tag is 0
25-32	TID	8 byte,if 6C tag isTID, If 6B tag the 8 byte is 0
33	USERLen	1byte,is USER district data length
34-250	USER	216 byte, tag USER district data
Total length of 250 bytes, if no using ,byte 0		

```
typedef struct stTagData
{
    BYTE DateTime[6];
    BYTE TagType;
    BYTE EPCLen;
    BYTE EPC[16];
    BYTE TIDLen;           //if 6B tag is 0
    BYTE TID[8];           //if 6B tag is 0
    BYTE USERLen;
    BYTE USER[216];
} TagData;
```

Notes

when SD Card time and end time both are 0 ,indicate down

if SUM > 0, means there are still some data need to be transmitted; if SUM = 0, means data transmission finished.

### 1.2.13 Delete all records in SD card

**Function:** delete all tags' records in SD card.

**Command code:** 17H

**Command packet:** [40H 02H 17H B7H]

**Return Data:** if command execute correct, return packet will be example null.

[F0H 02H 17H F7H]

### 1.2.14 Set handheld Bluetooth name

**Function:** set Bluetooth handset name

**Command code:** 8BH

**Command parameter** Data

**Command packet:** [40H 02H 8BH Data Check]

**Return Data:** if command execute success, return packet will be null.

[F0H 02H 8BH Check]

**Command parameter specification:**

PARAMETER NAME	LENGTH(BYTE)	INSTRUCTION
Data	20	Handheld Bluetooth name

### 1.2.15 Get handheld Bluetooth name

**Function:** get handheld Bluetooth name

**Command code:** 8CH

**Command parameter:** null

**Command packet:** [40H 02H 8CH Check]

**Return Data:** if successful, return packet will be null

[F0H 02H 8CH Data Check]

**Command parameter specification:**

PARAMETER NAME	LENGTH (BYTE)	INSTRUCTION
Data	20	Handheld Bluetooth name

### 1.3 Checksum computing method(C language)

```

unsigned char CheckSum(unsigned char *uBuff, unsigned char uBuffLen)
{
    unsigned char i,uSum=0;
    for(i=0;i<uBuffLen;i++)
    {
        uSum = uSum + uBuff[i];
    }
    uSum = (~uSum) + 1;
    return uSum;
}

```

## 2 SDK Software Development

### 2.1 SDK Component

The I-PICK product package provides the Software Development Kit (SDK), the SDK mainly consist of documents as below:

- A. VH75DIIlib.h — Dynamic Link Library
- B. VH75Lib.lib — Static libraries
- C. VH75HandsetAPI.h — API function declaration file
- D. SDK catalog — Include the demo software to study the API function

### 2.2 Design Description

#### 2.2.1 Basic Constant and Structure

##### 2.2.1.1 Definition of the Constant

Description		Explanation
#define ID_MAX_SIZE_64BIT	8	// The ID number of the label is 64bit
#define ID_MAX_SIZE_96BIT	13	// The ID number of the label is 128bit
#define MAX_LABELS	100	// Maximum tags to be read in one time operation is 100pcs

##### 2.2.1.2 API function return code

#define _OK	0x00	//operation succeeds.
//wrong information in communication		
#define _init_rs232_err	0x81	//fail to initialize the communication port
#define _no_scanner	0x82	//fail to find the handheld reader
#define _comm_error	0x83	//error occur when receiving and sending the communication data
#define _baudrate_error	0x84	// error occur when setting the baud rate
// return wrong information of the handheld reader		
#define _no_antenna	0x01	// fail to connect the antenna
#define _no_label	0x02	//cannot detect label
#define _invalid_label	0x03	//label is invalid
#define _less_power	0x04	//the power of reader is not enough
#define _write_prot_error	0x05	//write protection error
#define _check_sum_error	0x06	//checksum is wrong
#define _parameter_error	0x07	//parameter is wrong
#define _memory_error	0x08	//the memory area is not exist
#define _password_error	0x09	//the password is wrong
#define _killpassword_error	0x0a	//kill code of G2 tag is all zero
#define _nonlicet_command	0x0b	//nonlicet command
#define _nonlicet_user	0x0c	//nonlicet user for mismatch code
#define _invalid_command	0x1e	//means invalid command, like command of wrong parameter
#define _other_error	0x1f	//unknown order



// function input error		
#define _no_cardID_input	0x20	//other errors

### 2.2.1.3 Data Mode Definition

```
typedef USHORT apiReturn;    // function return mode
```

All API function will return to an apiReturn value after executing. By this value (function return code), we can figure out if the execute function is successful or the failure reason and other information.

### 2.2.1.4 Parameter structure of handheld reader

```
typedef struct tagReaderDate //date on handheld reader
{
    BYTE Year;           //year
    BYTE Month;          //month
    BYTE Day;            //day
    BYTE Hour;           //hour
    BYTE Minute;         //minute
    BYTE Second;         //second
}ReaderDate;

typedef struct tagHandsetParam
{
    BYTE TagType;        //(1)Tag Type : 01H—ISO18000-6B, 02H—EPCC1, 04H—EPCC1G2, 08H—EM4442
    BYTE Alarm;           //(2)bit0-bit7 bit0:0-not alarm,
                        ...// 1-alarm bit1:0-do not use white list 1-use white list.
    BYTE OutputMode;     //(3)data output mode : 0- reserve and output directly,
                        //1-reserve but do not output directly,
                        //2-do not reserve but output directly
    BYTE USBBaudRate;    //(4)USB port baud rate 04H--08H
    BYTE Reserve5;        //(5)reserve
    BYTE Min_Frequency;  //(6)initial point to send microwave frequency signal, value:1~63.
    BYTE Max_Frequency; //(7) initial point to send microwave frequency signal, value:1~63.
    BYTE Power;          //(8)transmitted power, value:0~160.
    BYTE RFhrdVer1;      //(9)RF module hardware major version
    BYTE RFhrdVer2;      //(10)RF module hardware minor version
    BYTE RFSftVer1;      //(11)RF module software major version
    BYTE RFSftVer2;      //(12)RF module software minor version
    BYTE ISTID;          //(13)if read TID zone
    BYTE TIDAddr;        //(14)TID read initial address
    BYTE TIDLen;         //(15)TID read length
    BYTE ISUSER;         //(16)if read USER
    BYTE USERAddr;       //(17)USER read initial address
    BYTE USERLen;        //(18)USER read length
    BYTE Reserve19;       //(19) motor vibration,0-no, 1-vibrate
    BYTE Reserve20;       //(20) module mode,0---R2000, 1---VM5F
    BYTE Reserve21;       //(21)reserve
    BYTE Reserve22;       //(22) reserve
    BYTE Reserve23;       //(23) reserve
    BYTE Reserve24;       //(24) reserve
    BYTE Reserve25;       //(25) reserve
    BYTE Reserve26;       //(26) reserve
    BYTE Reserve27;       //(27) reserve
    BYTE Reserve28;       //(28) reserve
    BYTE Reserve29;       //(29) reserve
    BYTE Reserve30;       //(30) reserve
    BYTE Reserve31;       //(31) reserve
    BYTE Reserve32;       //(32) reserve
} HandsetParam;
```

## 2.2.2 Control Command Function

### 2.2.2.1 *connect to handheld reader*

```
apiReturn _stdcall ConnectScanner(HANDLE *hScanner, char *szPort, int nBaudRate);
```

```
apiReturn _stdcall VH75_ConnectScannerUsb(HANDLE *hScanner);
```

Feature: connect to the handheld reader which is connected with the transmission port; and set transmission rate.

Inlet parameter:

hScanner : handle of the handheld reader

szPort : character pointer for transmission port, eg. serial port[COM1], [COM2]

nBaudRate: Baud rate when serial port is transmitting data, effective transmission rate: 9600, 19200, 38400, 57600, 115200.

Inlet parameter:

Figure out if connect successfully or failure reason by the return value apiReturn of the function.

### 2.2.2.2 *Disconnect*

RS232 disconnect the handheld reader:

```
apiReturn _stdcall DisconnectScanner(HANDLE hScanner);
```

```
apiReturn _stdcall VH75_DisconnectScannerUsb(HANDLE hScanner);
```

Feature: close the connection with handheld reader, release serial port resources

Inlet parameter:

**hScanner**: handle of the handheld reader

### 2.2.2.3 *Test if the USB is on*

```
typedef int (_stdcall *VH75_fDechUsb)(BOOL bDech);
```

bDech---1 is on or is off

### 2.2.2.4 *Initialize USB*

Can be tested, address of input function

```
apiReturn _stdcall VH75_InitUsb(VH75_fDechUsb fFunc);
```

### 2.2.2.5 *Close USB*

```
apiReturn _stdcall VH75_CloseUsb();
```

#### 2.2.2.6 *Transmission Mode Settings*

```
apiReturn _stdcall VH75_SetCommunicationOpt(long lOpt, void *lpRWHand);
```

Feature: set the transmission mode of the handheld reader

Input parameter: lOpt is transmission mode(1--RS232, 3--USB)

lpRWHand is the handle of the reader

Output parameter:

Return value: return 0 is success, others are fail

Remark: external interface

#### 2.2.2.7 *Character and time switch to metric system*

##### **Time switch to metric system**

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

Feature: time switch to metric system

Input parameter:

**hSacnner**: handle of the handheld reader communication port

##### **Character switch to metric system**

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

Feature: character switch to metric system

Input parameter:

**hSacnner**: handle of the handheld reader communication port

#### 2.2.2.8 *Compare if the two data are equal*

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

Feature: compare if the two record data are equal

Input parameter:

**hSacnner**: handle of the handheld reader communication port

olddata: old data.

data: new data.

Len: The length to compare.

Return: if the function return value is \_OK, then get successfully, or is unsuccessful.

#### 2.2.2.9 *Get Version*

```
apiReturn _stdcall VH75_GetVersion(HANDLE hScanner, WORD *wHardVer, WORD *wSoftVer);
```

Feature: get the hardware and software version of the handheld reader.

Input parameter:

**hScanner**: handle of the handheld reader communication port

Output parameter:

wHardVer: Hardware version of the handheld reader.

WSoftVer: Software version of the handheld reader.

Return: if the function return value is \_OK, then get successfully, or is unsuccessful.

#### 2.2.2.10 *Add label ID*

```
apiReturn _stdcall AddLabelID(HANDLE hScanner, int Count, int Len, BYTE * data);
```

Feature: add label ID list of the handheld reader

Input parameter:

**hScanner**: handle of the handheld reader communication port

Count: Number of label ID to add.

len: Length of single label ID.

data: Label ID to add.

Return: if the function return value is \_OK, then get successfully, or is unsuccessful.

#### 2.2.2.11 *Delete Label ID*

```
apiReturn _stdcall DelLabelID(HANDLE hScanner);
```

Feature: delete Label ID of the handheld reader

Input parameter:

**hScanner**: handle of the handheld reader communication port

#### 2.2.2.12 *Get Label ID*

```
apiReturn _stdcall GetLabelID(HANDLE hScanner, int startaddr, int listlen, int *nTotal, int *DataLen, BYTE * data);
```

Feature: read out the identified Label ID of the handheld reader

Input parameter:

**hSacnner**: handle of the handheld reader communication port

startaddr: Start from which list, 1word length.

listlen: How many Label ID get

Output parameter:

nTotal: actual reading data number

taglen: actual reading data length

data: Reading data

Return: if the function return value is \_OK, then get successfully, or is unsuccessful.

#### **2.2.2.13 Submit the list**

```
apiReturn _stdcall SaveLabelID(HANDLE hScanner);
```

Feature: save the manager computer's label ID to the handheld reader.

Input parameter:

**hSacnner**: handle of the handheld reader communication port

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

#### **2.2.2.14 Get the basic working parameter of the handheld reader**

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

Feature: read the working parameter that written to the handheld reader by previous command.

Input parameter:

**hSacnner**: handle of the handheld reader communication port

Output parameter:

pParam: return to the working parameter of the handheld rader, 32 bytes.

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

#### **2.2.2.15 Setting the basic working parameter of the handheld reader**

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

Feature: set the basic working parameter of the handheld reader.

Input parameter:

**hSacnner**: handle of the handheld reader communication port

**pParam**: working parameter of the handheld reader, 32 bytes.

Return: if the function return value is `_OK`, then reading is successful, or is failing reason.

#### **2.2.2.16 Reset to the factory parameter of the handheld reader**

```
apiReturn _stdcall ReadFactoryParameter(HANDLE hScanner);
```

Feature: reset to the factory parameter of the handheld reader.

Input parameter:

**hSacnner**: handle of the handheld reader communication port

Return: if the function return value is `_OK`, then reading is successful, or is failing reason.

#### **2.2.2.17 Setting Time**

```
apiReturn _stdcall SetReaderTime(HANDLE hScanner, ReaderDate time);
```

Feature: set the time of handheld reader by the time of manager computer

Input parameter:

**hSacnner**: handle of the handheld reader communication port

**time**: time of the manager computer, 6 bytes.

Return: if the function return value is `_OK`, then reading is successful, or is failing reason.

#### **2.2.2.18 Get the time**

```
apiReturn _stdcall GetReaderTime(HANDLE hScanner, ReaderDate *time);
```

Feature: read the time of the handheld reader

Input parameter:

**hSacnner**: handle of the handheld reader communication port

Output parameter:

**time**: return to the time of the handheld reader, 6 bytes.

Return: if the function return value is `_OK`, then reading is successful, or is failing reason.

#### **2.2.2.19 Setting the tag filter**

```
apiReturn _stdcall SetReportFilter(HANDLE hScanner, int ptr, int len, BYTE *mask);
```

Feature: set the tag filter

Input parameter:

**hSacnner**: handle of the handheld reader communication port

ptr: Mask initial address, unit is bit.

Len: Mask length, unit is bit.

Mask: Mask, hexadecimal.

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

#### **2.2.2.20 Get the tag filter**

```
apiReturn _stdcall GetReportFilter(HANDLE hScanner, int *ptr, int *len, BYTE *mask);
```

Feature: get the tag filter.

Input parameter:

**hSacnner**: handle of the handheld reader communication port

Output parameter:

ptr: Mask initial address, unit is bit.

Len: Mask length, unit is bit.

Mask Mask, hexadecimal.

#### **2.2.2.21 Get the record**

```
apiReturn _stdcall GetRecord(HANDLE hScanner, ReaderDate *stime, ReaderDate *etime, int *nTotal, int *taglen, BYTE * data);
```

Feature: read the indentified tag record from the handheld reader.

Input parameter:

**hSacnner**: handle of the handheld reader communication port

stime: start time

etime: end time

Output parameter:

nTotal: actual read record

taglen: actual read record length

data: Reading record.

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

#### 2.2.2.22 Delete all record

```
apiReturn _stdcall DeleteAllRecord(HANDLE hScanner);
```

Feature: delete all record of the handheld reader

Input parameter:

**hScanner**: handle of the handheld reader communication port

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

#### 2.2.2.23 Get handheld reader ID

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

Feature: after receiving the order, immediately send handheld reader ID to manager computer.

Input parameter:

**hScanner**: handle of the handheld reader communication port

Output parameter:

HandsetID: handheld reader ID

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

### 2.2.3 Read & Write ISO18000-6C Function

#### 2.2.3.1 Read ISO18000-6C tag's EPC number

```
apiReturn _stdcall EPC1G2_ReadLabelID(HANDLE hScanner, BYTE mem, int ptr, BYTE len, BYTE *mask, BYTE *IDBuffer, int *nCounter,int Address);
```

Feature: read all EPC numbers which can be indentified within radiation range of antenna.

Input parameter:

**hScanner**: handle of the handheld reader communication port

mem: choose data zone;

0	Password zone
1	EPC number
2	ID number of TID tag
3	User zone

ptr: mask initial address(unit: Bit)



len: mask length (unit: Bit)

mask: mask(unit: Byte), if len/8 is an integer number, then mask length is len/8; if len/8 is not an integer number, then mask length is len/8+1;final byte of mask put high, add zero in low.

**Address:** network address of the reader RS485 port

**Address =0:** disconnect the network

Output parameter:

IDBuffer: EPC number of tags had been read

NCounter: how many tags had been read

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

**Remark: LEN=0 means to read all identified tag ID within range of antenna's radiation**

### 2.2.3.2 Read a Block of data

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

Feature: read the data of a block continuous address memory in tag

Input parameter:

**hScanner:** handle of the handheld reader communication port

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

IDBuffer: selected EPC number of tags

mem: select data zone; 0-password zone, 1-EPC number, 2-TID tag ID number, 3- User zone。

ptr: read initial address(unit: WORD)

len: read length(unit: WORD)

AccessPassword:4 bytes AccessPassword

**Address:** network address of the reader RS485 port **Address =0:** disconnect the network

Output parameter:

Data: data had been read

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

**Remark: AccessPassword only work when the password zone is lock.**

### 2.2.3.3 Write a Block of data

```
apiReturn_stdcall EPC1G2_WriteWordBlock(HANDLE hScanner, BYTE EPC_WORD, BYTE *IDBuffer, BYTE mem, BYTE ptr, BYTE len, BYTE *Data, BYTE *AccessPassword,int Address);
```

Feature: write data to appointed address block of tag

Input parameter:

**hScanner**:handle of the handheld reader communication port

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

**IDBuffer**: selected EPC number of tags

**mem**: select data zone;

0	Password zone
1	EPC number
2	TID tag ID number
3	User zone

**ptr**: write initial address (unit: WORD)

**len**: write length (unit: WORD)

**Data**: data to write

**Access Password**:4 bytes Access Password password

**Address**: network address of the reader RS485 port

**Address** =0 disconnect the network

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

**Remark**: Access Password only work when the password zone is lock. When the password is unlock, then no need password to write; when the data is permanently lock, the password is useless.

### 2.2.3.4 Setting protection mode for read & write

```
apiReturn_stdcall EPC1G2_SetLock(HANDLE hScanner, BYTE EPC_WORD, BYTE *IDBuffer, BYTE mem, BYTE Lock, BYTE *AccessPassword,int Address);
```

Feature: set the assigned data zone of assigned tag as write protection.

Input parameter:

**hScanner**: handle of the handheld reader communication port

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

**IDBuffer**: selected EPC number of tags

Mem: select data zone;

0	Kill Password
1	Access Password
2	EPC number
3	TID tag ID number
4	User zone

Lock: control word Lock 。

0	Writable
1	Permanently writable
2	Writable by password
3	Never writable
4	Readable & writable
5	Permanently readable & writable
6	Readable & writable by password
7	Never readable & writable

Remark: 0 ~~are~~ only suitable for EPC, TID and User; 4 ~~are~~ only suitable for Kill Password and Access Password.

Access Password: 4 bytes Access Password password

**Address:** network address of the reader RS485 port

**Address** = 0 disconnect the network

Return: if the function return value is \_OK, then reading is successful, or is failing reason.

### 3 Appendix

#### 3.1 Document Revisions

Date	Version	Description
18. Feb 2016	0.1	First draft