UHFReader18.DLL 动态连接库

使用手册 V2.5

上位机应用程序通过 UHFReader18.DLL 操作 EPCC1-G2 和 18000-6B 格式电子标签读写器。

1. 操作系统：

WINDOWS 2000/XP

2. 函数详单：

UHFReader18.DLL 包括了如下的操作函数：

2.1）通用函数：

1)long WINAPI AutoOpenComPort(long *port,unsigned char * ComAdr,unsigned char baud, long *FrmHandle);

2)long WINAPI OpenComPort(long Port, unsigned char *ComAdr, unsigned char Baud，long *FrmHandle);

3)long WINAPI CloseComPort(void);

4)long WINAPI CloseSpecComPort(long FrmHandle);

5)long WINAPI GetReaderInformation(unsigned char *ComAdr, unsigned char *VersionInfo, unsigned char *ReaderType, unsigned char *TrType,unsigned char * dmaxfre , unsigned char *dminfre, unsigned char *powerdBm,unsigned char *ScanTime, long FrmHandle);

6)long WINAPI WriteComAdr(unsigned char *ComAdr, unsigned char *ComAdrData, long FrmHandle);

7)long WINAPI WriteScanTime(unsigned char *ComAdr, unsigned char *ScanTime, long FrmHandle);

8）long WINAPI SetPowerDbm (unsigned char *ComAdr, unsigned char powerDbm, long FrmHandle);

9)long WINAPI Writedfre (unsigned char *ComAdr, unsigned char dmaxfre, unsigned char dminfre,long FrmHandle);

10)long WINAPI Writebaud (unsigned char *ComAdr, unsigned char * baud, long FrmHandle);

11)long WINAPI SetWGParameter(unsigned char *ComAdr, unsigned char Wg_mode,unsigned char Wg_Data_Inteval,unsigned char Wg_Pulse_Width, unsigned char Wg_Pulse_Inteval,long FrmHandle);

12)long WINAPI SetWorkMode(unsigned char *ComAdr, unsigned char * Parameter, long FrmHandle);

13)long WINAPI GetWorkModeParameter (unsigned char *ComAdr, unsigned char * Parameter, long FrmHandle);

14)long WINAPI ReadActiveModeData (unsigned char *ActiveModeData, unsigned char * Datalength, long FrmHandle);

15)long WINAPI SetAccuracy(unsigned char *ComAdr, unsigned char Accuracy , long FrmHandle);

16)long WINAPI SetOffsetTime (unsigned char *ComAdr, unsigned char OffsetTime, long FrmHandle);

17)long WINAPI SetFhssMode (unsigned char *ComAdr, unsigned char FhssMode, long FrmHandle);

18)long WINAPI GetFhssMode (unsigned char *ComAdr, unsigned char *FhssMode, long FrmHandle);

19)long WINAPI SetTriggerTime(unsigned char *ComAdr, unsigned char* TriggerTime, long FrmHandle);

20)long WINAPI BuzzerAndLEDControl(unsigned char * ComAdr, unsigned char AvtiveTime, unsigned char SilentTime, unsigned char Times, long FrmHandle);

21)long WINAPI SetRelay (unsigned char * ComAdr, unsigned char RelayStatue, long FrmHandle);

22)long WINAPI OpenNetPort (int Port, LPSTR IPaddr, ,unsigned char *ComAdr, long FrmHandle);

23)long WINAPI CloseNetPort (long FrmHandle);

2.2）EPCC1-G2 协议函数：

1)long WINAPI Inventory_G2(unsigned char *ComAdr, unsigned char AdrTID, unsigned char LenTID, unsigned char TIDFlag,unsigned char * EPClenandEPC, long * Totallen, long *CardNum,long FrmHandle);

2)long WINAPI ReadCard_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char * Password , unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, unsigned char * Data , unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

3)long WINAPI WriteCard_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Mem, unsigned char WordPtr, unsigned char Writedatalen, unsigned char *Writedata,unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, long WrittenDataNum, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

4)long WINAPI EraseCard_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

5)long WINAPI SetCardProtect_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char select, unsigned char setprotect, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag,unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

6)long WINAPI DestroyCard_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag,unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

7)long WINAPI WriteEPC_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * WriteEPC, unsigned char WriteEPClen, unsigned char * errorcode,long FrmHandle);

8)long WINAPI SetReadProtect_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag,unsigned char EPClength, unsigned char * errorcode,long

FrmHandle);

9)long WINAPI SetMultiReadProtect_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * errorcode,long FrmHandle);

10)long WINAPI RemoveReadProtect_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * errorcode,long FrmHandle);

11)long WINAPI CheckReadProtected_G2 (unsigned char *ComAdr, unsigned char *readpro，unsigned char * errorcode,long FrmHandle);

12)long WINAPI SetEASAlarm_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, unsigned char EAS, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

13)long WINAPI CheckEASAlarm_G2 (unsigned char *ComAdr, unsigned char * errorcode,long FrmHandle);
14)long WINAPI LockUserBlock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, unsigned char BlockNum,unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

15)long WINAPI WriteBlock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Mem, unsigned char WordPtr, unsigned char Writedatalen, unsigned char *Writedata,unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, long WrittenDataNum, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

2.3）18000-6B 协议函数：
1)long WINAPI Inventory_6B (unsigned char *ComAdr, unsigned char * ID_6B ,long FrmHandle);

2)long WINAPI Inventory2_6B (unsigned char *ComAdr, unsigned char Condition , unsigned char StartAddress, unsigned char mask , unsigned char * ConditionContent，unsigned char * ID_6B , long * Cardnum,long FrmHandle);

3)long WINAPI ReadCard_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char StartAddress, unsigned char Num, unsigned char * Data, unsigned

char * errorcode, long FrmHandle);

4)long WINAPI WriteCard_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char StartAddress, unsigned char * Writedata, unsigned char Writedatalen, unsigned char * writtenbyte, unsigned char * errorcode, long FrmHandle);

5)long WINAPI LockByte _6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char Address, unsigned char * errorcode, long FrmHandle);

6)long WINAPI CheckLock_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char Address, unsigned char * ReLockState,unsigned char * errorcode, long FrmHandle);

3. 函数的描述：

3.1)通用函数：

3.1.1) AutoOpenComPort(): 自动连接串口

功能描述：

该函数用于自动识别与读写器连接的串口并且执行初始化操作，然后通过连接串口和读写器以创建通信连接。数据传输协议是 19200 bps，8 位数据，1 停止位，没有奇偶校验位。

在调用其它函数之前，您必须先连接串口和读写器。

应用：

long WINAPI AutoOpenComPort(long* Port, unsigned char *ComAdr, unsigned char Baud,long *FrmHandle);

参数：

Port：输出变量，COM1—COM9 与读写器连接的串口号。

ComAdr：输入/输出变量，远距离读写器的地址。以广播地址（0xFF）调用

此函数，函数将检测各个端口，并将检测到的连接有读写器的端口以及该端口上读写器的实际地址回写到指针 Port 和 ComAdr 所指变量中；以其它地址调用此函数，将在各个端口检测是否连接了具有指定 ComAdr 地址的读写器，并将检测到的端口号回写到指针 Port 所指变量中。

Baud：输入变量,用该值设置或更改串口通讯控件的波特率。

baudrate	实际波特率
0	9600bps
1	19200 bps
2	38400 bps
4	56000 bps
5	57600 bps
6	115200 bps

FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1.

COM1-COM9 的含义如下：

```
#define COM1        1
#define COM2        2
#define COM3        3
#define COM4        4
#define COM5        5
#define COM6        6
#define COM7        7
#define COM8        8
#define COM9        9
```

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.2) OpenComPort()：连接到指定串口

功能描述：

该函数用于指定串口初始化，并通过连接串口和读写器以创建通信连接。数据传输协议是 19200bps，8 位数据，1 位停止位，没有奇偶校验位。

在调用其它函数之前，您必须先连接串口和读写器。

应用：

long WINAPI OpenComPort(long Port, unsigned char *ComAdr, unsigned char Baud,long *FrmHandle);

参数：

Port：输入变量，COM1—COM9 常数。

ComAdr:输入/输出变量，远距离读写器的地址。以广播地址（0xFF）调用此函数，函数将检测指定端口，并将检测到的连接在此端口上的读写器的实际地址回写到指针 ComAdr 所指变量中；以其它地址调用此函数，将检测指定端口上是否连接了具有指定 ComAdr 地址的读写器。

Baud：输入变量,用该值设置或更改串口通讯控件的波特率。

baudrate	实际波特率
0	9600bps
1	19200 bps
2	38400 bps
4	56000 bps
5	57600 bps
6	115200 bps

FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1.

COM1-COM9 的定义如下：

```
#define COM1        1
#define COM2        2
#define COM3        3
#define COM4        4
#define COM5        5
#define COM6        6
#define COM7        7
#define COM8        8
#define COM9        9
```

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.3) CloseComPort():关闭串口连接

功能描述：

该函数用于撤销串口和读写器的连接并释放相应资源。在一些开发环境里，串口资源必须在离开该程序前被释放，否则可能会造成系统不稳定。

应用：

long WINAPI CloseComPort(void);

参数：无

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.4) CloseSpecComPort()：关闭指定串口

功能描述：

该函数用于关闭指定串口。

应用：

long WINAPI CloseSpecComPort(long FrmHandle);

参数：

FrmHandle：输入变量，与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.5)GetReaderInformation():获得读写器的信息

功能描述：

执行该命令后，将获得读写器的信息，这其中包括读写器地址（ComAdr）和读写器软件版本（VersionInfo）的信息等多项信息。

应用：

long WINAPI GetReaderInformation(unsigned char *ComAdr, unsigned char *VersionInfo, unsigned char *ReaderType, unsigned char *TrType,unsigned char * dmaxfre , unsigned char *dminfre, unsigned char *powerdBm,unsigned char *ScanTime, long FrmHandle);

参数：

ComAdr:输入/输出变量，远距离读写器的地址。以广播地址（0xFF）调用此函数，ComAdr 将返回读写器的实际地址，以其它地址调用此函数，将由 ComAdr 地址指定的读写器执行此函数命令。

VersionInfo：指向输出数组变量（输出的是每字节都转化为字符的数据），远距离读写器版本信息，长度 2 个字节。第 1 个字节为版本号，第 2 个字节为子版本号。

ReaderType:输出变量，读写器类型代码,0x09 代表 UHFReader18。

TrType:输出变量读写器协议支持信息，具体定义请参见用户手册。(bit1 为 1

表示支持 18000-6c 协议，其它位保留。)
dmaxfre: 输出变量，Bit7-Bit6 用于频段设置用；Bit5-Bit0 表示当前读写器工作的最大频率,具体定义请参见用户手册。
dminfre:输出变量，输出变量，Bit7-Bit6用于频段设置用；Bit5-Bit0表示当前读写器工作的最小频率，具体定义请参见用户手册。
PowerdBm: 输出变量,读写器的输出功率。范围是 0 到 18.
ScanTime:输出变量，读写器询查命令最大响应时间。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1.
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.6）WriteComAdr()：写入读写器地址

功能描述：
执行该命令后，读写器将会把读写器地址改为用户给定的值，并把这个值写入 EEPROM 保存。出厂时默认值是 0x00。允许用户的修改范围是 0x00~0xfe。当用户写入的值是 0xff 时，读写器将会自动恢复成默认值 0x00。
应用：
long WINAPI WriteComAdr(unsigned char *ComAdr, unsigned char *ComAdrData, long FrmHandle);
参数：
ComAdr：输入变量，原先的读写器地址
ComAdrData：输入变量，一个字节,待写入的读写器地址
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1.
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.7）WriteScanTime()：设置询查命令最大响应时间

功能描述：
询查命令的最大响应时间范围是 3～255*100ms，默认值为 10*100ms。
应用：
long WINAPI WriteScanTime(unsigned char *ComAdr, unsigned char

*ScanTime, long FrmHandle);

参数：

ComAdr：输入变量，读写器地址

ScanTime：输入变量，一个字节, 询查命令响应时间

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1.

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.8）SetPowerDbm（）：设置读写器功率

功能描述：

本命令用来设置读写器功率。

应用：

long WINAPI SetPowerDbm (unsigned char *ComAdr, unsigned char powerDbm, long FrmHandle);

参数：

ComAdr：输入变量，读写器地址

Powerdbm：输入变量，一个字节。读写器的输出功率。取值范围是 20~30。30 约为 1 瓦的输出功率。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1.

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.9）Writedfre（）：设置读写器工作频率

功能描述：

此命令设置读写器工作地上限频率，下限频率。上限频率必须大于或等于下限频率。

应用：

long WINAPI Writedfre (unsigned char *ComAdr, unsigned char * dmaxfre, unsigned char * dminfre,long FrmHandle);

参数：

ComAdr ：输入变量，读写器地址
dmaxfre：输入变量，Bit7-Bit6用于频段设置用；Bit5-Bit0表示当前读写器工作的最大频率,具体定义请参见用户手册。
Dminfre：输入变量，Bit7-Bit6用于频段设置用；Bit5-Bit0表示当前读写器工作的最小频率,具体定义请参见用户手册。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.10）Writebaud ()：设置串口波特率

功能描述：
此命令用来更改读写器的串口波特率。
应用：
long WINAPI Writebaud (unsigned char *ComAdr, unsigned char * baud, long FrmHandle);
参数：
ComAdr ：输入变量，读写器地址
baud: 输入变量，一个字节。读写器上电后，波特率默认为 57600。Baud 的范围是 0～6。其它值保留。其对应的波特率为：

baudrate	实际波特率
0	9600bps
1	19200 bps
2	38400 bps
4	56000 bps
5	57600 bps
6	115200 bps

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.11）SetWGParameter()：设置韦根参数

功能描述：

此命令用来设置韦根参数。

应用：

long WINAPI SetWGParameter(unsigned char *ComAdr, unsigned char Wg_mode,unsigned char Wg_Data_Inteval,unsigned char Wg_Pulse_Width, unsigned char Wg_Pulse_Inteval,long FrmHandle);

参数：

ComAdr ：输入变量，读写器地址

Wg_mode: 输入变量，一个字节。

Bit0:韦根 26，34 选择位。Bit0=0 时，选择韦根 26，Bit0=1 时选择韦根 34.

Bit1：Bit1=0 时韦根输出高字节在前，Bit1=1 时，韦根输出低字节在前。其他位保留，默认为 0.

Wg_Data_Inteval：输出数据间隔时间(0~255)*100ms，默认为 30.

Wg_Pulse_Width：数据脉冲宽度(1~255)*100us,默认值为 10.

Wg_Pulse_Inteval：数据脉冲间隔(1~255)*100us，默认值为 15.

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.12）SetWorkMode()：设置工作模式

功能描述：

此命令用来设置工作模式参数。

应用：

long WINAPI SetWorkMode(unsigned char *ComAdr, unsigned char * Parameter, long FrmHandle);

参数：

ComAdr ：输入变量，读写器地址

Parameter: 指向输入数组变量，6 个字节。从第一个字节至第六个分别为：

Read_Mode: Bit0：协议选择位。Bit0=0 时读写器支持 18000-6C 协议；

Bit0=1 时读写器支持 18000-6B 协议。Bit1：输出方式选择位。

Bit1=0 时韦根输出，Bit1=1 时 RS232/RS485 输出。

Bit2：蜂鸣器提示选择位。Bit2=0 时开蜂鸣器提示，

Bit2=1 时关蜂鸣器提示，默认值为 0。

Bit3：韦根输出模式下 First_Adr 参数为字地址或字节地址选择位。

Bit3=0 时 First_Adr 为字地址；Bit3=1 时 First_Adr 为字节地址。
Bit4：玺瑞 485 选择位，Bit1=0 时该位无效。
Bit4=0 时是普通 485 输出方式，Bit4=1 时是玺瑞 485 模式。
玺瑞 485 模式下只支持单标签操作（18000-6C、18000-6B 均有效）（读保留区、EPC 区、TID 区、用户区，单张查询）。玺瑞 485 模式下 First_Adr 为字节地址。其它位保留，默认为 0
Mem_Inven：当读写器工作在 18000-6C 协议时才有效，选择要读取的存储区或询查标签。0x00：保留区；0x01：EPC 存储器；0x02：TID 存储器；0x03：用户存储器；0x04：多张查询；0x05：单张查询；0x06 ：EAS 检测。其他值保留，若命令中出现了其它值，将返回参数出错的消息。
First_Adr：指定要读取的起始地址。18000-6C 协议中：0x00 表示从第一个字(第一个 16 位存储区)开始读，0x01 表示从第 2 个字开始读，依次类推；18000-6B 中：0x00 表示从第一个字节开始读，0x01 表示从第 2 个字节开始读，依次类推。
Word_Num：要读取的字的个数，RS232 输出方式下才有效。不能设置为 0x00，否则将返回参数错误信息。Word_Num 不能超过 32，若 Word_Num 设置为 0 或者超过了 32，将返回参数出错的消息。
Tag_Time：主动模式下单张标签操作(读保留区、EPC 区、TID 区、用户区，单张查询)间隔时间(0~255)*1s，对同一张标签在间隔时间内只操作一次。默认值为零，即对标签操作不用等待时间。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.13）GetWorkModeParameter（）：读取工作模式参数

功能描述：
此命令用来读取工作模式参数。
应用：
long WINAPI GetWorkModeParameter (unsigned char *ComAdr, unsigned char * Parameter, long FrmHandle);
参数：
ComAdr ：输入变量，读写器地址.
Parameter: 指向输出数组变量，11 个字节。从第一个字节至第十一个分别为：Wg_mode ， Wg_Data_Inteval ， Wg_Pulse_Width ， Wg_Pulse_Inteval ，Read_mode，Mode_state，Mem_Inven，First_Adr，Word_Num，Tag_Time，

Accuracy。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

3. 1. 14）ReadActiveModeData（）：读取主动模式数据-

功能描述：
此命令用来读取主动模式下读写器发送数据。
应用：
long WINAPI ReadActiveModeData (unsigned char *ActiveModeData, unsigned char * Datalength, long FrmHandle);
参数：
ActiveModeData: 指向输出数组变量，读取主动模式下读写器发送数据，大小为 Datalength 个字节。
Datalength：输出变量，ActiveModeData 的字节大小。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

3.1.15) SetAccuracy():EAS 检测精度测试

功能描述：
该命令用于设置主动模式下 EAS 检测精度。
应用：
long WINAPI SetAccuracy(unsigned char * ComAdr, unsigned char Accuracy , long FrmHandle);
参数：
ComAdr ：输入变量，读写器地址.

Accuracy：输入变量，范围为 0~8，数值越大精度越高。默认值为 8。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

3.1.16) SetOffsetTime ():Syris 响应偏置时间设置

功能描述：
该命令用于设置 Syris485 模式下的 Syris 命令响应偏置时间。
应用：
long WINAPI SetOffsetTime(unsigned char * ComAdr, unsigned char OffsetTime, long FrmHandle);
参数：
ComAdr ：输入变量，读写器地址.
OffsetTime：输入变量，Syris485 命令响应偏置时间,范围(0~100)*1ms。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

3.1.17) SetFhssMode ():设置读写器跳频模式

功能描述：
该命令用于设置读写器的跳频模式。
应用：
long WINAPI SetOffsetTime(unsigned char * ComAdr, unsigned char FhssMode, long FrmHandle);
参数：
ComAdr ：输入变量，读写器地址.
FhssMode：输入变量，Bit0=0 时为随机跳频模式，Bit=1 时为自适应跳频模式，默认值为 0。Bit1~Bit7：保留，默认值为 0。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

3.1.18) GetFhssMode ():获取读写器跳频模式。

功能描述：

该命令用于获取读写器的跳频模式。

应用：

long WINAPI GetFhssMode(unsigned char * ComAdr, unsigned char *FhssMode, long FrmHandle);

参数：

ComAdr ：输入变量，读写器地址.

FhssMode：输出变量，Bit0=0 时为随机跳频模式，Bit0=1 时为自适应跳频模式。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义.

3.1.19) SetTriggerTime():触发延时设置。

功能描述：

该命令用于设置或获取读写器触发模式下的触发有效时间。

应用：

long WINAPI SetTriggerTime(unsigned char * ComAdr, unsigned char *TriggerTime, long FrmHandle);

参数：

ComAdr ：输入变量，读写器地址.

TriggerTime：输入/输出变量，触发有效时间(0~254)*1s，默认值为 0，当触发有效时间设置为 255 时，为获取当前的触发有效时间。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码

定义

3.1.20) BuzzerAndLEDControl ()：声光控制命令 -

功能描述：

该命令用来控制 LED 灯和蜂鸣器按一定规律闪烁和鸣叫。

应用：

Long BuzzerAndLEDControl(unsigned char * ComAdr, unsigned char AvtiveTime, unsigned char SilentTime, unsigned char Times, long FrmHandle);

参数：

ComAdr ：输入变量，读写器地址.

AvtiveTime：输入变量，LED 灯亮和蜂鸣器鸣叫时间(ActiveT*50ms)，默认值为零。0<=ActiveT<=255，当设置成 255 时，为获取当前触发有效时间。

SilentTime: 输入变量,LED 灯和蜂鸣器静默时间(SilentT *50ms)，默认值为零。0<= SilentT <=255。

Times: 输入变量,LED 灯亮和蜂鸣器鸣叫次数(0<=Times<=255) 默认值为零.

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

3.1.21) SetRelay ()：设置继电器动作 -

功能描述：

该命令用于设置继电器。

应用：

long WINAPI SetRelay(unsigned char *ComAdr,unsigned char RelayStatue,long FrmHandle);

参数：

ComAdr ：输入变量，读写器地址.

RelayStatue：输入变量，bit0 和 bit1 有效，每一位对应一个继电器，为 1 时继电器吸合，为 0 时继电器释放。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

3.1.22) OpenNetPort():连接网口

功能描述：

该函数用于打开与读写器连接的网口，然后通过连接网口和读写器以创建通信连接。

在调用其它函数之前，您必须先连接网口和读写器。

应用：

long WINAPI OpenNetPort(long Port , LPTSTR IPaddr, unsigned char *ComAdr,long * FrmHandle);

参数：

Port：输入变量，范围在 1024 到 65535 间，读写器的 TCP 端口号。

IPaddr：输入变量，读写器的 TCP 的 IP 地址。

ComAdr：输入/输出变量，远距离读写器的地址。以广播地址（0xFF）调用此函数，函数将检测端口，并将检测到的连接有读写器的读写器地址返回到 ComAdr 所指变量中；以其它地址调用此函数，将在 TCP 端口检测是否连接了具有指定 ComAdr 地址的读写器。

FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1.

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.1.23) CloseNetPort()：关闭网口

功能描述：

该函数用于撤销网口和读写器的连接并释放相应资源。在一些开发环境里，网口资源必须在离开该程序前被释放，否则可能会造成系统不稳定。

应用：

long WINAPI CloseNetPort(long FrmHandle);

参数：

FrmHandle：输入变量，与读写器连接网口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。

3.2) EPCC1-G2 协议函数：

3.2.1) Inventory_G2 ()：G2 询查命令

功能描述：

询查命令的作用是检查有效范围内是否有符合协议的电子标签存在。

应用：

long WINAPI Inventory_G2 (unsigned char *ComAdr, unsigned char AdrTID, unsigned char LenTID,unsigned char TIDFlag,unsigned char *EPClenandEPC, long * Totallen, long *CardNum,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

AdrTID：输入变量，询查 TID 区的起始字地址。

LenTID：输入变量，询查 TID 区的数据字数。LenTID 取值为 0~15。

TIDFlag：输入变量，

TIDFlag=1:表示询查 TID 区；

TIDFlag=1:表示询查 EPC；

EPClenandEPC：指向输出数组变量（输出的是每字节都转化为字符的数据）。是读到的电子标签的 EPC 数据，是一张标签的 EPC 长度+一张标签的 EPC 号，依此累加。每个电子标签 EPC 号高字在前，每一个字的最高位在前。

Totallen：输出变量，EPClenandEPC 的字节数。

CardNum：输出变量，电子标签的张数。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回值：

0x01 询查时间结束前返回

0x02 询查时间结束使得询查退出

0x03 如果读到的标签数量无法在一条消息内传送完，将分多次发送。如果 Status 为 0x0D，则表示这条数据结束后，还有数据。

0x04 还有电子标签未读取，电子标签数量太多，MCU 存储不了

返回其他值，请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.2) ReadCard_G2 ()：G2 读取数据命令

功能描述：

这个命令读取标签的整个或部分保留区、EPC 存储器、TID 存储器或用户存储器中的数据。从指定的地址开始读，以字为单位。

应用：

long WINAPI ReadCard_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char * Password , unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, unsigned char * Data , unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号

Mem：输入变量，一个字节。选择要读取的存储区。

0x00：保留区；

0x01：EPC 存储器；

0x02：TID 存储器；

0x03：用户存储器。

其他值保留。若命令中出现了其它值，将返回参数出错的消息。

WordPtr：输入变量，一个字节。指定要读取的字起始地址。0x00 表示从第一个字(第一个 16 位存储体)开始读，0x01 表示从第 2 个字开始读，依次类推。

Num：输入变量，一个字节。要读取的字的个数。不能设置为 0x00，将返回参数错误信息。Num 不能超过 120，即最多读取 120 个字。若 Num 设置为 0 或者超过了 120，将返回参数出错的消息。

Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

Data：指向输出数组变量（输出的是每字节都转化为字符的数据），是从标签中读取的数据。

EPClength：输入变量，一个字节。EPC 号的字节长度。

Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

maskadr：输入变量，EPC 掩模起始字节地址。

maskLen：输入变量，掩模字节数。

maskFlag：输入变量，掩模使能标记。

maskFlag =1：掩模使能；

maskFlag =0：掩模禁止；

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过

该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值，读到的数据在 Data 中。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.3) WriteCard_G2 ()：G2 写命令

功能描述：

这个命令可以一次性往保留内存、EPC 存储器、TID 存储器或用户存储器中写入若干个字。

应用：

long WINAPI WriteCard_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Mem, unsigned char WordPtr, unsigned char Writedatalen, unsigned char * Writedata, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, long *WrittenDataNum, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

Mem：输入变量，一个字节。选择要读取的存储区。

0x00: 保留区；

0x01：EPC 存储器；

0x02：TID 存储器；

0x03：用户存储器。

其他值保留。若命令中出现了其它值，将返回参数出错的消息。

WordPtr：输入变量，一个字节。指定要写入的字起始地址。指定要写入数据的起始地址。如果写的是 EPC 区，则会忽略这个起始地址。EPC 区总是规定从 EPC 区 0x02 地址(EPC 号的第一个字节)开始写。

Writedatalen：输入变量，一个字节。待写入的字节数（长度必须为偶数字节数）。Writedatalen 必须大于 0，这里字节数必须和实际待写入的数据个数相等。否则将会返回参数错误的消息。

Writedata：指向输入数组变量（输入的是每字节都转化为字符的数据）。待写入的字。这是要写入到存储区的数据。比如，WordPtr 等于 0x02，则输出变量 Data 中第一个字(从左边起)写在 Mem 指定的存储区的地址 0x02 中，第二个字写在 0x03 中，依次类推。

Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的

第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。
WrittenDataNum：输出变量，已经写入的字的个数。（以字为单位）
EPClength：输入变量，一个字节。EPC 号的字节长度。
Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。
maskadr：输入变量，EPC 掩模起始字节地址。
maskLen：输入变量，掩模字节数。
maskFlag：输入变量，掩模使能标记。
maskFlag =1：掩模使能；
maskFlag =0：掩模禁止；
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值，完全写入。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.4) EraseCard_G2 ()：G2 块擦除命令

功能描述：
此命令可以擦除标签的保留内存、EPC 存储器、TID 存储器或用户存储器的若干字。
应用：
long WINAPI EraseCard_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);
参数：
ComAdr：输入变量，读写器地址。
EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。
Mem：输入变量，一个字节。选择要读取的存储区。
0x00：保留区；
0x01：EPC 存储器；
0x02：TID 存储器；
0x03：用户存储器。
其他值保留。若命令中出现了其它值，将返回参数出错的消息。

WordPtr：输入变量，一个字节。指定要擦除的字起始地址 0x00 表示从第一个字(第一个 16 位存储体)开始擦除，0x01 表示从第 2 个字开始擦除，依次类推。当擦除 EPC 区时，WordPtr 必须大于等于 0x02，若小于 0x02，则返回参数错误消息。

Num：输入变量，一个字节。指定要擦除的字的个数。从 WordPtr 指定的地址开始擦除，擦除 Num 指定个数的字。若 Num 为 0x00，则返回参数错误信息。

Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

EPClength：输入变量，一个字节。EPC 号的字节长度。

Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

maskadr：输入变量，EPC 掩模起始字节地址。

maskLen：输入变量，掩模字节数。

maskFlag：输入变量，掩模使能标记。

maskFlag =1：掩模使能；

maskFlag =0：掩模禁止；

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值，擦除成功。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.5) SetCardProtect_G2 ()：G2 设定存储区读写保护状态命令

功能描述：

这个命令可以设定保留区为可读写、永远可读写、带密码可读写、永远可读写，可以分别设定 EPC 存储器、TID 存储器和用户存储器为可写、永远可写、带密码可写、永远不可写。EPC 存储器、TID 存储器或用户存储器是永远可读的。而且，TID 存储器是只读的，永远都不可写。

应用：

long WINAPI SetCardProtect_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char select, unsigned char setprotect, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。
EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。
Select: 输入变量，一个字节。
0x00 时，控制 Kill 密码读写保护设定。
0x01 时，控制访问密码读写保护设定。
0x02 时，控制 EPC 存储器读写保护设定。
0x03 时，控制 TID 存储器读写保护设定。
0x04 时，控制用户存储器读写保护设定。
其它值保留，若出读写器接收到了其他值，将返回参数出错的消息。
Setprotect：输入变量，一个字节。
当 Select 为 0x00 或 0x01，SetProtect 值代表的意义如下:
0x00：设置为可读写
0x01：设置为永远可读写
0x02：设置为带密码可读写
0x03：设置为永远不可读写
当 Select 为 0x02、0x03、0x04 时，SetProtect 值代表的意义如下:
0x00：设置为可写
0x01：设置为永远可写
0x02：设置为带密码可写
0x03：设置为永远不可写
Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。
EPClength：输入变量，一个字节。EPC 号的字节长度。
Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。
maskadr：输入变量，EPC 掩模起始字节地址。
maskLen：输入变量，掩模字节数。
maskFlag：输入变量，掩模使能标记。
maskFlag =1：掩模使能；
maskFlag =0：掩模禁止；
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值，设置成功。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码

定义。

3.2.6) DestroyCard_G2 ()：G2 销毁标签命令

功能描述：

这个命令用来销毁标签。标签销毁后，永远不会再处理读写器的命令。

应用：

long WINAPI DestroyCard_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

EPClength：输入变量，一个字节。EPC 号的字节长度。

Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

maskadr：输入变量，EPC 掩模起始字节地址。

maskLen：输入变量，掩模字节数。

maskFlag：输入变量，掩模使能标记。

maskFlag =1：掩模使能；

maskFlag =0：掩模禁止；

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值，销毁成功。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.7) WriteEPC_G2 ()：G2 写 EPC 号命令

功能描述：

这个命令向电子标签写入 EPC 号。写入的时候，天线有效范围内只能有一张电子标签。

应用：

long WINAPI WriteEPC_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * WriteEPC, unsigned char WriteEPClen, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

WriteEPC：指向输入数组变量（输入的是每字节都转化为字符的数据），对电子标签写入的 EPC 号（覆盖原 EPC 号）。

WriteEPClen：输入变量，一个字节。WriteEPC 的字节长度。范围 2~30 字节，必须为偶数字节长度，即 2、4、6…30。

Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值，写入成功。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.8) SetReadProtect_G2 ()：G2 单张读保护设置命令

功能描述：

这个命令根据电子标签的 EPC 号，对标签设置读保护，使得电子标签不能被任何命令读写，对标签进行询查操作，也无法得到电子标签的 EPC 号。仅对 NXP 的 UCODE EPC G2X 标签有效。

应用：

long WINAPI SetReadProtect_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag,unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。
EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。
Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。
maskadr：输入变量，EPC 掩模起始字节地址。
maskLen：输入变量，掩模字节数。
maskFlag：输入变量，掩模使能标记。
maskFlag =1：掩模使能；
maskFlag =0：掩模禁止；
EPClength：输入变量，一个字节。EPC 号的字节长度。
Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。
FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：
如果该函数调用成功，返回一个零值，读保护设置成功。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.9) SetMultiReadProtect_G2 ()：G2 多张读保护设置命令

功能描述：
这个命令可以为有效范围内的电子标签设定读保护。这个命令与前面一个任务的区别是，当有效范围内存在多张标签的时候，无法知道这个命令操作的是哪一张电子标签。如果要对多张标签进行操作，则标签的访问密码最好是相同的。仅对 NXP 的 UCODE EPC G2X 标签有效。
应用：
```
long WINAPI SetMultiReadProtect_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * errorcode,long FrmHandle);
```
参数：
ComAdr：输入变量，读写器地址。
Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。
FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值，多张读保护设置成功。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.10) RemoveReadProtect_G2 ()：G2 解锁读保护命令

功能描述：
这个命令用来给设置了读保护的标签解锁。用这个命令时，天线有效范围内只能放置一张要被解锁的电子标签。仅对 NXP 的 UCODE EPC G2X 标签有效。
应用：
long WINAPI RemoveReadProtect_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * errorcode,long FrmHandle);
参数：
ComAdr：输入变量，读写器地址。
Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。
Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值，解锁读保护成功。说明: 对于不支持读保护设定的标签，认为没有被解锁。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.11) CheckReadProtected_G2 ()：G2 测试标签是否被读保护命令

功能描述：
这个命令不能测试标签是否支持读保护锁定，只能测试标签是否被读保护锁定。对于不支持读保护锁定的电子标签，一致认为没有被锁定。

这个命令只能对单张电子标签进行操作，确保天线有效范围内只存在一张电子标签。仅对 NXP 的 UCODE EPC G2X 标签有效。

应用：

long WINAPI CheckReadProtected_G2 (unsigned char *ComAdr, unsigned char *Readpro ， unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

Readpro：输出变量，一个字节。Readpro 为 00，电子标签没有被设置为读保护。Readpro 为 01，电子标签被设置读保护。说明：对于不支持读保护设定的标签，认为没有被设置读保护。

Errorcode：输出变量，一个字节，保留。

FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.12) SetEASAlarm_G2 ()：G2 EAS 报警设置命令

功能描述：

对电子标签的 EAS 状态位进行设置或复位。仅对 NXP 的 UCODE EPC G2 标签有效。

应用：

long WINAPI SetEASAlarm_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag,unsigned char EAS, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

maskadr：输入变量，EPC 掩模起始字节地址。

maskLen：输入变量，掩模字节数。

maskFlag：输入变量，掩模使能标记。

maskFlag =1：掩模使能；
maskFlag =0：掩模禁止；
EAS: 输入变量，1 个字节。bit0 位为 0，表示设置为关闭 EAS 报警；为 1，表示设置为打开 EAS 报警。bit1 – bit7 位默认为 0，保留。
EPClength：输入变量，一个字节。EPC 号的字节长度。
Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。
FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值，EAS 报警设置成功。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.13) CheckEASAlarm_G2 ()：G2 EAS 报警探测命令

功能描述：
该命令检测电子标签的 EAS 报警。仅对 NXP 的 UCODE EPC G2 标签有效。
应用：
long WINAPI CheckEASAlarm_G2 (unsigned char *ComAdr, unsigned char * errorcode,long FrmHandle);
参数：
ComAdr：输入变量，读写器地址。
Errorcode：输出变量，一个字节，保留。
FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
EAS 报警，返回零值。
无 EAS 报警，返回值为 0xFB。
否则，返回其他值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.14) LockUserBlock_G2 ()：G2 user 区块锁命令（永久锁定）

功能描述：

这个命令每次永久锁定 user 区中的 32bits 数据，锁定后，这 32bits 数据只能读，不能被再次写，也不能被擦除。仅对 UCODE EPC G2 电子标签有效。

应用：

long WINAPI LockUserBlock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag,unsigned char BlockNum,unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

maskadr：输入变量，EPC 掩模起始字节地址。

maskLen：输入变量，掩模字节数。

maskFlag：输入变量，掩模使能标记。

maskFlag =1：掩模使能；

maskFlag =0：掩模禁止；

BlockNum：输入变量，一个字节，要锁定的字地址。一次会锁定 2 个字：

当 BlockNum 为 0 或 1 永久锁定 数据块(字地址) 0 和 1

当 BlockNum 为 2 或 3 永久锁定 数据块(字地址) 2 和 3

当 BlockNum 为 4 或 5 永久锁定 数据块(字地址) 4 和 5

当 BlockNum 为 6 或 7 永久锁定 数据块(字地址) 6 和 7

当 BlockNum 为 8 或 9 永久锁定 数据块(字地址) 8 和 9

当 BlockNum 为 10 或 11 永久锁定 数据块(字地址) 10 和 11

当 BlockNum 为 12 或 13 永久锁定 数据块(字地址) 12 和 13

EPClength：输入变量，一个字节。EPC 号的字节长度。

Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

FrmHandle：输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值，锁定成功。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.2.15) WriteBlock_G2 ()：G2 块写命令

功能描述：

该命令一次能将多个字写入标签的保留区、EPC 区、TID 区或用户区。

应用：

long WINAPI WriteBlock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Mem, unsigned char WordPtr, unsigned char Writedatalen, unsigned char * Writedata, unsigned char * Password, unsigned char maskadr, unsigned char maskLen, unsigned char maskFlag, long *WrittenDataNum, unsigned char EPClength, unsigned char * errorcode,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

EPC：指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

Mem：输入变量，一个字节。选择要读取的存储区。

0x00: 保留区；

0x01：EPC 存储器；

0x02：TID 存储器；

0x03：用户存储器。

其他值保留。若命令中出现了其它值，将返回参数出错的消息。

WordPtr：输入变量，一个字节。指定要写入的字起始地址。指定要写入数据的起始地址。如果写的是 EPC 区，则会忽略这个起始地址。EPC 区总是规定从 EPC 区 0x02 地址(EPC 号的第一个字节)开始写。

Writedatalen：输入变量，一个字节。待写入的字节数（长度必须为偶数字节数）。Writedatalen 必须大于 0，这里字节数必须和实际待写入的数据个数相等。否则将会返回参数错误的消息。

Writedata：指向输入数组变量（输入的是每字节都转化为字符的数据）。待写入的字。这是要写入到存储区的数据。比如，WordPtr 等于 0x02，则输出变量 Data 中第一个字(从左边起)写在 Mem 指定的存储区的地址 0x02 中，第二个字写在 0x03 中，依次类推。

Password：指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

WrittenDataNum：输出变量，已经写入的字的个数。（以字为单位）

EPClength：输入变量，一个字节。EPC 号的字节长度。

Errorcode：输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。
maskadr：输入变量，EPC 掩模起始字节地址。
maskLen：输入变量，掩模字节数。
maskFlag：输入变量，掩模使能标记。
maskFlag =1：掩模使能；
maskFlag =0：掩模禁止；
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值，完全写入。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.3) 18000-6B 协议函数：

3.3.1) Inventory_6B ()：6B 寻查命令(单张)

功能描述：
本命令只能询查单张电子标签。如果多张标签同时处于天线有效范围内，可能无法询查到电子标签。
应用：
long WINAPI Inventory_6B (unsigned char *ComAdr, unsigned char * ID_6B ,long FrmHandle);
参数：
ComAdr：输入变量，读写器地址。
ID_6B：指向输出数组变量（输出的是每字节都转化为字符的数据），8 个字节。是电子标签的 ID 号。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回零值，查找了一张标签。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.3.2) Inventory2_6B ()：6B 按条件寻查电子标签命令

功能描述：
本命令按照给定的条件询查电子标签。

应用：

long WINAPI Inventory2_6B (unsigned char *ComAdr, unsigned char * Condition , unsigned char StartAddress, unsigned char mask , unsigned char* ConditionContent ， unsigned char * ID_6B , long * Cardnum,long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

Condition：输入变量，选择标签的要求。 0x00: 等于条件； 0x01: 不等于条件； 0x02: 大于条件；0x03: 小于条件。

StartAddress：输入变量，用于比较的标签起始地址。

Mask：输入变量，掩码，一个字节。用来指定要比较的数据。Mask 的每一位对应 ConditionContent 的一个字节。Mask 的最高位对应(bit7)对应 ConditionContent 的最左边的一个字节。Mask 的最低位对应(bit0)对应 ConditionContent 的最右边一字节。Mask 相应位为 1，表示要把该位在 ConditionContent 中对应的字节与标签中对应的字节进行比较;为 0 则不比较。

ConditionContent：指向输入数组变量（输入的是每字节都转化为字符的数据），八个字节。用于比较的条件，即用来比较的数据。

ID_6B：指向输出数组变量（输出的是每字节都转化为字符的数据）。多张电子标签 ID，每个 ID 号八个字节。

Cardnum：输出变量，电子标签数量，范围为 0 – 31。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回值：

0x15 询查时间结束前返回

0x16 询查时间结束使得询查退出

0x17 如果读到的标签数量无法在一条消息内传送完，将分多次发送。如果 Status 为 0x0D，则表示这条数据结束后，还有数据。

0x18 还有电子标签未读取，电子标签数量太多，MCU 存储不了

返回其他值，请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.3.3) ReadCard_6B ()：6B 读数据命令

功能描述：

该命令用来从电子标签的某个指定地址开始读若干个字节。

应用：

long WINAPI ReadCard_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char StartAddress, unsigned char Num, unsigned char * Data, unsigned char * errorcode, long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

ID_6B：指向输入数组变量（输入的是每字节都转化为字符的数据）。要读数据的电子标签的 ID，八个字节。

StartAddress：输入变量，一个字节。读数据的起始地址。地址范围为 0～223。如果地址超出 223，将返回参数错误信息。

Num：输入变量，一个字节，要读的数据字节数。 Num 的范围是 1～32。如果 StartAddress +Num 大于 224、或是 Num 超出 32、或 Num 为 0，读写器将返回参数错误信息。

Data：指向输出数组变量（输出的是每字节都转化为字符的数据）。是读到的数据内容。

Errorcode：输出变量，一个字节，保留。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值，读数据成功。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.3.4) WriteCard_6B ()：6B 写数据命令

功能描述：

该命令向指定的电子标签写入若干个字节。一次可以写入 1～4 个字节。

应用：

long WINAPI WriteCard_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char StartAddress, unsigned char * Writedata, unsigned char Writedatalen, unsigned char * writtenbyte, unsigned char * errorcode, long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

ID_6B：指向输入数组变量（输入的是每字节都转化为字符的数据）。要写数据的电子标签的 ID，八个字节。

StartAddress：输入变量，一个字节。读数据的起始地址。地址范围为 8～223。如果地址超出范围，将返回参数错误信息。

Writedata：指向输入数组变量（输入的是每字节都转化为字符的数据）。要写入的数据。长度限定在 32 个字节以内，超过 32 或 Address 加 wdta 的长度大于 224，读写器将返回参数错误信息。Writedata 高字节写在电子标签

的低地址。
Writedatalen：输入变量，一个字节。写入数据的字节数。
Writtenbyte：输出变量，一个字节。写成功的字节数。写入的部分从命令中的 Writedata 的高字节算起。
Errorcode：输出变量，保留，
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值，写数据成功。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.3.5) CheckLock_6B ()：6B 锁定检测命令

功能描述：
该命令用来检测指定的字节是否锁定。
应用：
long WINAPI CheckLock_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char Address, unsigned char * ReLockState，unsigned char * errorcode, long FrmHandle);
参数：
ComAdr：输入变量，读写器地址。
ID_6B：指向输入数组变量（输入的是每字节都转化为字符的数据）。要被检测锁定的电子标签的 ID，八个字节。
Address：输入变量，一个字节。要检测是否被锁定的字节的地址。地址范围为 0~ 223。如果地址超出范围，将返回参数错误信息。
ReLockState：输出变量，一个字节。ReLockState 为 0x0，不能锁定。ReLockState 为 0x02，已经锁定，不能再次锁定。
Errorcode：输出变量，一个字节，保留。
FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。
返回：
如果该函数调用成功，返回一个零值。
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

3.3.6) LockByte_6B ()：6B 锁定命令

功能描述：

该命令锁定指定的字节。

应用：

long WINAPI LockByte_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char Address, unsigned char * errorcode, long FrmHandle);

参数：

ComAdr：输入变量，读写器地址。

ID_6B：指向输入数组变量（输入的是每字节都转化为字符的数据）。要被锁定的电子标签的 ID，八个字节。

Address：输入变量，一个字节。要锁定的字节地址。范围：8 ~ 223。超过这个范围将返回参数错误信息。

Errorcode：输出变量，一个字节，保留。

FrmHandle：输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回：

如果该函数调用成功，返回一个零值，该字节未被锁定。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

4. 其他返回值定义

#define InventoryReturnEarly_G2	0x01 询查时间结束前返回
#define InventoryTimeOut_G2	0x02 指定的询查时间溢出
#define InventoryMoreData_G2	0x03 本条消息之后，还有消息
#define ReadermoduleMCUFull_G2	0x04 读写模块存储空间已满

#define AccessPasswordError	0x05	访问密码错误
#define DestroyPasswordError	0x09	销毁密码错误
#define DestroyPasswordCannotZero	0x0a	销毁密码不能为全 0
#define TagNotSupportCMD	0x0b	电子标签不支持该命令
#define AccessPasswordCannotZero	0x0c	对该命令，访问密码不能为 0
#define TagProtectedCannotSetAgain	0x0d	电子标签已经被设置了读保护，不能再次设置
#define TagNoProtectedDonotNeedUnlock	0x0e	电子标签没有被设置读保护，不需要解锁
#define ByteLockedWriteFail	0x10	有字节空间被锁定，写入失败
#define CannotLock	0x11	不能锁定
#define LockedCannotLockAgain	0x12	已经锁定，不能再次锁定
#define ParameterSaveFailCanUseBeforeNoPower	0x13	参数保存失败,但设置的值在读写模块断电前有效
#define CannotAdjust	0x14	无法调整
#define InventoryReturnEarly_6B	0x15	询查时间结束前返回
#define InventoryTimeOut_6B	0x16	指定的询查时间溢出
#define InventoryMoreData_6B	0x17	本条消息之后，还有消息
#define ReadermoduleMCUFull_6B	0x18	读写模块存储空间已满
#define NotSupportCMDOrAccessPasswordCannotZero	0x19	电子不支持该命令或者访问密码不能为 0
#define CMDExecuteErr	0xF9	命令执行出错
#define GetTagPoorCommunicationCannotOperation	0xFA	有电子标签，但通信不畅，无法操作
#define NoTagOperation	0xFB	无电子标签可操作
#define TagReturnErrorCode	0xFC	电子标签返回错误代码
#define CMDLengthWrong	0xFD	命令长度错误
#define IllegalCMD	0xFE	不合法的命令
#define ParameterError	0xFF	参数错误
#define CommunicationErr	0x30	通讯错误
#define RetCRCErr	0x31	CRC 校验错误
#define RetDataErr	0x32	返回数据长度有错误
#define CommunicationBusy	0x33	通讯繁忙，设备正在执行其他指令
#define ExecuteCmdBusy	0x34	繁忙，指令正在执行

```
#define  ComPortOpened          0x35 端口已打开
#define  ComPortClose           0x36 端口已关闭
#define  InvalidHandle          0x37 无效句柄
#define  InvalidPort            0x38 无效端口
#define  RecmdErr               0XEE 返回指令错误
```

5. 错误代码定义

#define OtherError	0x00	其它错误，全部捕捉未被其它代码覆盖的错误
#define MemoryOutPcNotSupport	0x03	存储器超限或不被支持的 PC 值，规定存储位置不存在或标签不支持 PC 值
#define MemoryLocked	0x04	存储器锁定，存储位置锁定或永久锁定，且不可写入
#define NoPower	0x0b	电源不足，标签电源不足，无法执行存储写入操作
#define NotSpecialError	0x0f	非特定错误，标签不支持特定错误代码