

SID_UF2861.DLL Dynamic Link Library

User's Guide V1.4

1.	OPERATION SYSTEM REQUIREMENT:	1
2.	FUNCTION LIST:	1
2.1)	GENERAL FUNCTION:	1
2.2)	EPCC1-G2 FUNCTION:	3
2.3)	18000-6B FUNCTION:	4
3.	FUNCTION EXPLANATION:	5
3.1)	GENERAL FUNCTION:	5
3.1.1)	<i>AutoOpenComPort()</i> :	5
3.1.2)	<i>OpenComPort()</i> :	6
3.1.3)	<i>CloseComPort()</i> :	7
3.1.4)	<i>CloseSpecComPort()</i> :	7
3.1.5)	<i>GetReaderInformation()</i> :	8
3.1.6)	<i>SetAddress ()</i> :	8
3.1.7)	<i>SetInventoryScanTime ()</i> :	9
3.1.8)	<i>SetRfPower ()</i> :	9
3.1.9)	<i>SetRegion ()</i> :	9
3.1.10)	<i>SetBaudRate ()</i> :	10
3.1.11)	<i>BuzzerAndLEDControl ()</i> :	10
3.1.12)	<i>SetWorkMode ()</i> :	11
3.1.13)	<i>GetSystemParameter ()</i> :	11
3.1.14)	<i>ReadActiveModeData ()</i> :	12
3.1.15)	<i>SetEASSensitivity ()</i> :	12
3.1.16)	<i>SetMask ()</i> :	13
3.1.17)	<i>SetResponsePamametersofAuto_runningMode ()</i> :	13
3.1.18)	<i>SetInventoryInterval ()</i> :	14
3.1.19)	<i>SelectTagType ()</i> :	14
3.1.20)	<i>SetAntennaMultiplexing ()</i> :	15
3.1.21)	<i>SetBeepNotification ()</i> :	15
3.1.22)	<i>SetReal_timeClock ()</i> :	15
3.1.23)	<i>GetTime ()</i> :	16
3.1.24)	<i>GetTagBufferInfo ()</i> :	16
3.1.25)	<i>ClearTagBuffer ()</i> :	17
3.1.26)	<i>SetRelay()</i> :	17
3.1.27)	<i>SetGPIO()</i> :	17
3.1.28)	<i>GetGPIOStatus()</i> :	17
3.1.29)	<i>SetNotificationPulseOutput()</i> :	18
3.1.30)	<i>SetTriggerTine ()</i> :	18
3.1.31)	<i>SetTIDParameter ()</i> :	19
3.1.32)	<i>OpenNetPort ()</i> :	19
3.1.33)	<i>CloseNetPort ()</i> :	19

3.1.34) <i>ChangeATMode</i> ():	19
3.1.35) <i>TransparentCMD</i> ():	20
3.1.36) <i>GetSeriaNo</i> ():	20
3.2) EPCC1-G2 FUNCTION:	20
3.2.1) <i>Inventory_G2</i> ():	20
3.2.2) <i>ReadData_G2</i> ():	22
3.2.3) <i>WriteData_G2</i> ():	23
3.2.4) <i>BlockErase_G2</i> ():	24
3.2.5) <i>Lock_G2</i> ():	25
3.2.6) <i>KillTag_G2</i> ():	26
3.2.7) <i>WriteEPC_G2</i> ():	27
3.2.8) <i>SetPrivacyByEPC_G2</i> ():	27
3.2.9) <i>SetPrivacyWithoutEPC_G2</i> ():	28
3.2.10) <i>ResetPrivacy_G2</i> ():	28
3.2.11) <i>CheckPrivacy_G2</i> ():	29
3.2.12) <i>EASConfigure_G2</i> ():	29
3.2.13) <i>EASAlarm_G2</i> ():	30
3.2.14) <i>BlockLock_G2</i> ():	30
3.2.15) <i>BlockWrite_G2</i> ():	31
3.3) 18000-6B FUNCTION:	32
3.3.1) <i>InventorySingle_6B</i> ():	32
3.3.2) <i>InventoryMultiple_6B</i> ():	33
3.3.3) <i>ReadData_6B</i> ():	34
3.3.4) <i>WriteData_6B</i> ():	34
3.3.5) <i>CheckLock_6B</i> ():	35
3.3.6) <i>Lock_6B</i> ():	35
4. RETURN VALUE DEFINITION	36
5.ERRORCODE DEFINITION	38

SID_UF2861.DLL is a dynamic link library designed to facilitate EPCC1-G2 and 18000-6B protocol UHF tag application software development.

1. Operation System Requirement:

WINDOWS 2000/XP/7/8

2. Function List:

SID_UF2861.DLL includes the following functions:

2.1) General Function:

- 1)int AutoOpenComPort(int *port,unsigned char * ComAdr,unsigned char baud, int *FrmHandle);
- 2)int OpenComPort(int Port, unsigned char *ComAdr, unsigned char Baud, int *FrmHandle);
- 3)int CloseComPort(void);
- 4)int CloseSpecComPort(int FrmHandle);
- 5)int 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, unsigned char *Ant, unsigned char *BeepEn, unsigned char *OutputRep, unsigned char * Reserve,int FrmHandle);
- 6)int SetAddress(unsigned char *ComAdr, unsigned char ComAdrData, int FrmHandle);
- 7)int SetInventoryScanTime(unsigned char *ComAdr, unsigned char ScanTime, int FrmHandle);
- 8)int SetRfPower(unsigned char *ComAdr, unsigned char powerDbm,int FrmHandle);
- 9)int SetRegion(unsigned char *ComAdr, unsigned char dmaxfre, unsigned char dminfre,int FrmHandle);
- 10)int SetBaudRate (unsigned char *ComAdr, unsigned char baud, int FrmHandle);
- 11)int BuzzerAndLEDControl (unsigned char *ComAdr, unsigned char AvtiveTime, unsigned char SilentTime, unsigned char Times,int FrmHandle);
- 12)int SetWorkMode (unsigned char *ComAdr, unsigned char Read_mode, int FrmHandle);
- 13)ong GetSystemParameter (unsigned char *ComAdr, unsigned char *Read_mode, unsigned char *Accuracy, unsigned char *RepCondition, unsigned char* RepPauseTime, unsigned char *ReadPauseTim, unsigned char *TagProtocol, unsigned char *MaskMem, unsigned char* MaskAdr, unsigned char *MaskLen, unsigned char* MaskData , unsigned char *TriggerTime, unsigned char

*AdrTID,unsigned char *LenTID,int FrmHandle);

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

15)int SetEASSensitivity (unsigned char *ComAdr, unsigned char Accuracy, int FrmHandle);

16)int SetMask(unsigned char *ComAdr, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData,int FrmHandle);

17)int SetResponsePamametersofAuto_runningMode(unsigned char *ComAdr, unsigned char RepCondition, unsigned char RepPauseTime, int FrmHandle);

18)int SetInventoryInterval(unsigned char *ComAdr, unsigned char ReadPauseTim, int FrmHandle);

19)int SelectTagType(unsigned char *ComAdr, unsigned char Protocol, int FrmHandle);

20)int SetAntennaMultiplexing(unsigned char *ComAdr, unsigned char Ant, int FrmHandle);

21)int SetBeepNotification(unsigned char *ComAdr, unsigned char BeepEn, int FrmHandle);

22)int SetReal_timeClock(unsigned char *ComAdr, unsigned char*ClockTime, int FrmHandle);

23)int GetTime (unsigned char *ComAdr, unsigned char* ClockTime, int FrmHandle);

24)int GetTagBufferInfo(unsigned char *ComAdr, unsigned char* Data,int dataLength, int FrmHandle);

25)int ClearTagBuffer(unsigned char *ComAdr, int FrmHandle);

26)int SetRelay(unsigned char *ComAdr, unsigned char RelayTime, int FrmHandle);

27)int SetGPIO(unsigned char *ComAdr, unsigned char OutputPin, int FrmHandle);

28)int GetGPIOStatus(unsigned char *ComAdr, unsigned char *OutputPin, int FrmHandle);

29)int SetNotificationPulseOutput(unsigned char *ComAdr, unsigned char OutputRep, int FrmHandle);

30)int SetTriggerTime (unsigned char *ComAdr, unsigned char TriggerTime, int FrmHandle);

31)int SetTIDParameter (unsigned char *ComAdr, unsigned char TIDAddr, unsigned char TIDLen, int FrmHandle);

32)int OpenNetPort (int Port, LPSTR IPAddr,unsigned char *ComAdr, int FrmHandle);

33)int CloseNetPort (int FrmHandle);

34) int ChangeATMode(unsigned char *ComAdr, unsigned char ATMode, int FrmHandle);

35) int TransparentCMD(unsigned char *ComAdr, unsigned char timeout, unsigned char cmdlen,unsigned char* cmdData,unsigned char recvlen, unsigned char* recvData,int FrmHandle);

36) int GetSeriaNo(unsigned char *ComAdr, unsigned char *SeriaNo, int FrmHandle);

2.2) EPCC1-G2 Function:

1)int Inventory_G2(unsigned char *ComAdr, unsigned char QValue, unsigned char Session,unsigned char MaskMem,unsigned char *MaskAdr,unsigned char MaskLen,unsigned char *MaskData, unsigned char MaskFlag, unsigned char AdrTID, unsigned char LenTID, unsigned char TIDFlag, unsigned char Target, unsigned char InAnt, unsigned char Scantime, unsigned char FastFlag,unsigned char * EPClenandEPC, unsigned char *Ant, int * Totallen, int *CardNum,int FrmHandle);

2)int ReadData_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char * Password , unsigned char MaskMem, unsigned char*MaskAdr, unsigned char MaskLen, unsigned char*MaskData, unsigned char * Data ,unsigned char * errorcode,int FrmHandle);

3)int WriteData_G2(unsigned char *ComAdr, unsigned char * EPC, unsigned char Wnum, unsigned char Enum,unsigned char Mem, unsigned char WordPtr,unsigned char *Writedata,unsigned char * Password, unsigned char MaskMem,unsigned char*MaskAdr,unsigned char MaskLen,unsigned char * MaskData,unsigned char * errorcode,int FrmHandle);

4)int BlockErase_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char *Password,unsigned char MaskMem,unsigned char* MaskAdr, unsigned char MaskLen,unsigned char * MaskData, unsigned char * errorcode,int FrmHandle);

5)int Lock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum, unsigned char select, unsigned char setprotect, unsigned char * Password, unsigned char MaskMem,unsigned char* MaskAdr, unsigned char MaskLen,unsigned char * MaskData, unsigned char * errorcode,int FrmHandle);

6)int KillTag_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char * Password, unsigned char MaskMem,unsigned char* MaskAdr, unsigned char MaskLen,unsigned char * MaskData, unsigned char * errorcode,int FrmHandle);

7)int WriteEPC_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char *

WriteEPC, unsigned char Enum, unsigned char * errorcode,int FrmHandle);

8)int SetPrivacyByEPC_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char *Password, unsigned char MaskMem,unsigned char *MaskAdr,unsigned char MaskLen,unsigned char * MaskData,unsigned char * errorcode,int FrmHandle);

9)int SetPrivacyWithoutEPC_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * errorcode,int FrmHandle);

10)int ResetPrivacy_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * errorcode,int FrmHandle);

11)int CheckPrivacy_G2 (unsigned char *ComAdr, unsigned char *readpro, unsigned char * errorcode,int FrmHandle);

12)int EASConfigure_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char * Password, unsigned char EAS, unsigned char MaskMem,unsigned char *MaskAdr,unsigned char MaskLen,unsigned char * MaskData, unsigned char * errorcode,int FrmHandle);

13)int EASAlarm_G2(unsigned char *ComAdr, unsigned char * errorcode,int FrmHandle);

14)int BlockLock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char * Password, unsigned char WrdPointer, unsigned char MaskMem,unsigned char *MaskAdr,unsigned char MaskLen,unsigned char * MaskData,unsigned char * errorcode,int FrmHandle);

15)int BlockWrite_G2(unsigned char *ComAdr, unsigned char * EPC, unsigned char Wnum, unsigned char Enum,unsigned char Mem, unsigned char WordPtr,unsigned char *Writedata,unsigned char * Password, unsigned char MaskMem,unsigned char *MaskAdr,unsigned char MaskLen,unsigned char * MaskData,unsigned char * errorcode,int FrmHandle);

2.3) 18000-6B Function:

1)int InventorySingle_6B(unsigned char *ComAdr, unsigned char *Ant, unsigned char * ID_6B ,int FrmHandle);

2)int InventoryMultiple_6B (unsigned char *ComAdr, unsigned char Condition, unsigned char StartAddress, unsigned char mask , unsigned char * ConditionContent, unsigned char *Ant, unsigned char * ID_6B , int * Cardnum,int FrmHandle);

3)int ReadData_6B (unsigned char *ComAdr, unsigned char * ID_6B,unsigned char StartAddress, unsigned char Num, unsigned char * Data, unsigned char * errorcode, int FrmHandle);

4)int WriteData_6B(unsigned char *ComAdr, unsigned char * ID_6B,unsigned char StartAddress, unsigned char *Writedata, unsigned char Writedatalen, unsigned char *writtenbyte, unsigned char * errorcode, int FrmHandle);

5)int Lock_6B(unsigned char *ComAdr, unsigned char *ID_6B,unsigned char Address,unsigned char *errorcode,int FrmHandle);

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

3. Function Explanation:

3.1) General Function:

3.1.1) AutoOpenComPort():

Function description:

This function is used to automatically detect the communication port unoccupied by other application and attached with a reader. The function try to establish the connection between them. The protocol default parameters are 57600bps, 8 data bits, 1 start bit, 1 stop bit, no parity bit.

If the connection is established successfully, the function will open the communication port and return a valid handle, otherwise the function will return an error code with a invalid handle(value as -1).

Usage:

Int AutoOpenComPort(int* Port, unsigned char *ComAdr, unsigned char Baud,int* FrmHandle);

Parameter:

Port: Pointed to the communication port number (COM1~COM9) that the reader is detected and connected.

ComAdr: Pointed to the address of the reader.

When using broadcasting address 0xFF as ComAdr to call the function, the port number to which the reader is detected and the address of the reader will be writed back to parameter Port and ComAdr;

When using a designated address 0x00~0xFE as ComAdr to call the function, the port number to which the reader with the specified address is detected will be writed back to parameter Port.

Constants COM1~COM9 are defined as follows:

```
#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
```

Baud: This value set the baud rate of the serial communication control.

baudrate	Actual baud rate
0	9600bps
1	19200 bps
2	38400 bps
5	57600 bps
6	115200 bps

FrmHandle: Pointed to the communication handle which is binding with the communication port opened successfully. The application software should use this handle to manipulate the reader connected to the port.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.2) OpenComPort():

Function description:

This function is used to establish the connection between the reader and a specified communication port. The protocol parameters are 19200bps, 8 data bits, 1 start bit, 1 stop bit, no parity bit.

Usage:

Int OpenComPort(int Port, unsigned char *ComAdr, unsigned char Baud, int* FrmHandle);

Parameter:

Port: Communication port number which is a constant from COM1 to COM9 defined as following:

```
#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
```

ComAdr: Pointed to the address of the reader.

When using broadcasting address 0xFF as ComAdr to call the function, the address of the reader will be writed back to parameter ComAdr;

When using a designated address 0x00~0xFE as ComAdr to call the function, the function will detect whether a specified address reader is connected to the designaged communication port.

Baud: This value set the baud rate of the serial communication control.

baudrate	Actual baud rate
0	9600bps
1	19200 bps
2	38400 bps
5	57600 bps
6	115200 bps

FrmHandle: Pointed to the communication handle which is binding with the communication port opened successfully. The application software should use this handle to manipulate the reader connected to the port.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.3) CloseComPort():

Function description:

This function is used to disconnect the reader and release the corresponding communication port resources. In some development environment, the communication port resources must be released before exiting. Otherwise the operation system will become unstable.

Usage:

```
Int CloseComPort(void);
```

Parameter: None.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.4) CloseSpecComPort():

Function description:

This function is used to disconnect the reader with the designated communication port and release the corresponding resources.

Usage:

```
Int CloseSpecComPort(int FrmHandle);
```

Parameter:

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.5)GetReaderInformation():

Function description:

This function is used to get reader-related information such as reader address(ComAdr), firmware version, supported protocol type , InventoryScanTime ,dmaxfre dminfre, powerdBm, ScanTime, Ant, BeepEn and OutputRep.

Usage:

```
Int 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, unsigned char *Ant, unsigned char *BeepEn, unsigned char *OutputRep, unsigned char * Reserve, int FrmHandle);
```

Parameter:

ComAdr: Pointed to the address of the reader.

VersionInfo: Pointed to 2 bytes firmware version information. The first byte is version number, the second byte is sub-version number.

ReaderType: Pointed to the reader type byte. 0x0A lines on SID_UF2861

TrType: Pointed to One bytes supported protocol information.

dmaxfre: Output variable, Bit7-Bit6 band set for use; Bit5-Bit0 that the current maximum frequency reader to work, the specific definitions, see the user manual.

dminfre: Output variable, Bit7-Bit6 band set for use; Bit5-Bit0 reader work that the current minimum frequency, the specific definitions, see the user manual.

PowerdBm: The output power of reader. Range is 0 to 18, when PowerdBm is 0x00, it means the output power of reader unknown.

ScanTime: Point to the value of time limit for inventory command. Please refer to RRU1861 User's manual for details.

Ant: Output variable, Antenna configuration information.

BeepEn: Output variable, Buzzer information.

OutputRep: Output variable, output port parameters information.

Reserve: Reserve.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.6) SetAddress ():

Function description:

This function is used to set a new address of the reader. The address value will store in reader's inner nonvolatile memory. Default address value is 0x00. The value range is 0x00~0xFE. The address 0xFF is reserved as the broadcasting address. When user try to write a 0xFF to ComAdr, the reader will set the value to 0x00 automatically.

Usage:

Int SetAddress(unsigned char *ComAdr, unsigned char ComAdrData, int FrmHandle);

Parameter:

ComAdr: Pointed to the original address of the reader.

ComAdrData: Pointed to the new address of the reader.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.7) SetInventoryScanTime ():

Function description:

This function is used to set a new value to InventoryScanTime of an appointed reader. The range is 3~255 corresponding to 3*100ms~255*100ms InventoryScanTime. The default value of InventoryScanTime is 30*100ms.

Usage:

Int SetInventoryScanTime(unsigned char *ComAdr, unsigned char ScanTime, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

InventoryScanTime: Pointed to the value of InventoryScanTime.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.8) SetRfPower ():

Function description:

The function is used to set the power of reader.

Usage:

Int SetRfPower (unsigned char *ComAdr, unsigned char powerDbm, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Powerdbm: The output power of reader. range is 0~18

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.9) SetRegion ():

Function description:

The function is used to set the reader working of the lower limit and the upper limit of frequency.

Usage:

Int SetRegion (unsigned char *ComAdr, unsigned char dmaxfre, unsigned char dminfre, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

dmaxfre: Input variable, Bit7-Bit6 band set for use; Bit5-Bit0 that the current maximum frequency reader to work, the specific definitions, see the user manual.

dminfre: Input variable, Bit7-Bit6 band set for use; Bit5-Bit0 reader work that the current minimum frequency, the specific definitions, see the user manual.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.10) SetBaudRate ():

Function description:

The function is used to change the serial port baud rate.

Usage:

Int SetBaudRate (unsigned char *ComAdr, unsigned char * baud, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Baud: After reader power on, the baud rate of reader is 57600bps. Range is 0~6.

baudrate	Actual baud rate
0	9600bps
1	19200 bps
2	38400 bps
5	57600 bps
6	115200 bps

Reader support 43000bps baud rate, but ApdComPort control in DLL is not support 43000bps.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.11) BuzzerAndLEDControl ():

Function description:

The function is used to control LED and Buzzer statue.

Usage:

Int BuzzerAndLEDControl (unsigned char *ComAdr, unsigned char AvtiveTime, unsigned char SilentTime, unsigned char Times, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

AvtiveTime: Input variables, LED lights and buzzer chirping time (ActiveT * 50ms), the default value is zero. $0 \leq \text{ActiveT} \leq 255$.

SilentTime: input variables, LED lamp and buzzer silent time (SilentT * 50ms), the default value is zero. $0 \leq \text{SilentT} \leq 255$.

Times: Input variables, LED lights and buzzer chirping number ($0 \leq$ to send the ≤ 255) to zero by default.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.12) SetWorkMode ():

Function description:

The function is used to set work mode.

Usage:

Int SetWorkMode(unsigned char *ComAdr, unsigned char Read_mode, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Read_mode: Input variables

Work mode selection:

Bit1	Bit0	Work mode
0	0	Answer mode
0	1	Auto_running mode
1	0	Trigger mode(Low-Level)
1	1	Trigger mode(High-Level)

Other bits reserved, the default value is 0.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.13) GetSystemParameter ():

Function description:

The function is used to get work mode parameter.

Usage:

int GetSystemParameter (unsigned char *ComAdr, unsigned char *Read_mode, unsigned char *Accuracy, unsigned char *RepCondition, unsigned char *RepPauseTime, unsigned char *ReadPauseTim, unsigned char *TagProtocol, unsigned char *MaskMem, unsigned char *MaskAdr, unsigned char *MaskLen, unsigned char *MaskData , unsigned char *TriggerTime, unsigned char *AdrTID,unsigned char *LenTID,int FrmHandle)

Parameter:

ComAdr: Pointed to the address of the reader.

Read_mode: Output variables, one byte, work mode information.

Accuracy: Output variables, one byte, EAS Accuracy information.

RepCondition, RepPauseTime: Output variables, one byte, Reader response way parameters.

ReadPauseTim: Output variables, one byte, Reader response pause time.

TagProtocol: Output variables, one byte, query tag type.

MaskMem、MaskAdr、MaskLen、MaskData: In active mode, reader query G2 tag editor mask condition argument. Among them MaskMem for one byte, MaskAdr for two bytes, high byte before, MaskLen for a byte, MaskData fixed for 32 bytes, more than MaskLen instructions content is zero.

TriggerTime: Output, Trigger time, range (0~255)*1s.

AdrTID: Output, read TID's start address.

LenTID: Output, read TID's word number.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.14) ReadActiveModeData ():

Function description:

The function is used to read data with Active mode.

Usage:

Int ReadActiveModeData(unsigned char *ActiveModeData, unsigned char * Datalength, int FrmHandle);

Parameter:

ActiveModeData: Point to the output array variable, read the active mode, the reader sends the data, the size of Datalength bytes.

Datalength: Output ,the Length of ActiveModeData.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.15) SetEASSensitivity ():

Function description:

The function is used to Set EAS Accuracy .

Usage:

Int SetEASSensitivity (unsigned char *ComAdr, unsigned char Accuracy, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Accuracy: Input variables.

Bit7=1:Relay close 3s when EAS detected

Bit7=0:None.

Bit6~Bit0: Rang is 0-8, EAS accuracy.

FrmHandle: Handle of the corresponding communication port the reader is connected. The

handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.16) SetMask ():

Function description:

The function is used to set query G2 tag conditions.

Usage:

```
Int SetMask(unsigned char *ComAdr, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, int FrmHandle);
```

Parameter:

ComAdr: Pointed to the address of the reader.

MaskMem: Input variables, one byte, Mask memory.

0x01: EPC;

0x02: TID;

0x03: User.

MaskAdr: Input Array, two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen: Input variables, one bytes, the bit length of Mask.

MaskData: Input Array, the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.17) SetResponseParametersofAuto_runningMode ():

Function description:

The function is used to Set reader response mode in active mode.

Usage:

```
Int SetResponseParametersofAuto_runningMode(unsigned char *ComAdr, unsigned char RepCondition, unsigned char RepPauseTime, int FrmHandle);
```

Parameter:

ComAdr: Pointed to the address of the reader.

RepCondition: Input, one byte, response mode.

0x00: Command notify;

0x01: Timer notify;

0x02: Add-in notify;

0x03: Delete notify;

0x04: Change notify;

RepPauseTime: Input, one byte, In Timer notify mode, Reader notify PC pause time. (0~255)*1s.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.18) SetInventoryInterval ():

Function description:

The function is used to Set Reader query tag pause time.

Usage:

Int SetInventoryInterval(unsigned char *ComAdr, unsigned char ReadPauseTim, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

ReadPauseTim: Input.one byte, Pause time.

ReadPauseTim	Time
0x00	10ms
0x01	20ms
0x02	30ms
0x03	50ms
0x04	100ms

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.19) SelectTagType ():

Function description:

The function is used to Set Reader query tag type.

Usage:

Int SelectTagType(unsigned char *ComAdr, unsigned char Protocol, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Protocol: Input.one byte,

Bit7:Tag protocol select bit;

Bit7=0:G2(18000-6C);

Bit7=1:18000-6B;

TagProtocol=Bit6Bit5...Bit0

When Bit7=0:

TagProtocol=0x00:query G2 tag;

TagProtocol=0x01:Check EAS before query Tag;

TagProtocol=0x02:Check EAS.

When Bit7=1:

TagProtocol=0x00:query 18000-6B tag.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.20) SetAntennaMultiplexing ():

Function description:

The function is used to Set Reader antenna config information..

Usage:

Int SetAntennaMultiplexing(unsigned char *ComAdr, unsigned char Ant, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Ant: Input.one byte, Antenna config information.

Bit0: antenna 1 config bits. Bit0 = 0 antenna 1 closed down, Bit0 = 1 antenna 1 open.

Bit1: antenna 2 config bits. Bit1 = 0 antenna 2 closed down, Bit1 = 1 antenna 2 open.

Bit2: antenna 3 config bits. Bit2 = 0 antenna 3 closed down, Bit2 = 1 antenna 3 open.

Bit3: antenna 4 config bits. Bit3 = 0 antenna 4 closed down, Bit3 = 1 antenna 4 open.

Bit4 ~ Bit7: reserved, the default value is 0

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.21) SetBeepNotification ():

Function description:

The function is used to Set Reader'Beep.

Usage:

Int SetBeepNotification(unsigned char *ComAdr, unsigned char BeepEn, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Ant: Input.one byte, Antenna config information.

Bit0: antenna 1 config bits. Bit0 = 0 antenna 1 closed down, Bit0 = 1 antenna 1 open.

Bit1: antenna 2 config bits. Bit1 = 0 antenna 2 closed down, Bit1 = 1 antenna 2 open.

Bit2: antenna 3 config bits. Bit2 = 0 antenna 3 closed down, Bit2 = 1 antenna 3 open.

Bit3: antenna 4 config bits. Bit3 = 0 antenna 4 closed down, Bit3 = 1 antenna 4 open.

Bit4 ~ Bit7: reserved, the default value is 0

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.22) SetReal_timeClock ():

Function description:

The function is used to Set Reader' really-time-clock.

Usage:

Int SetReal_timeClock(unsigned char *ComAdr, unsigned char*ClockTime, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

ClockTime: Input array.6 bytes.

Year	Month	Date	Hour	Minute	Second
------	-------	------	------	--------	--------

Year:range is 0-63.(2000-2063) ;

Month:range is 0-12;

Date:range is 0-31;

Hour:range is 0-23;

Minute:range is 0-59;

Second:range is 0-59;

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.23) GetTime ():

Function description:

The function is used to Get Reader' really-time-clock.

Usage:

Int GetTime(unsigned char *ComAdr, unsigned char*ClockTime, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

ClockTime: Output array.6 bytes.

Year	Month	Date	Hour	Minute	Second
------	-------	------	------	--------	--------

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.24) GetTagBufferInfo ():

Function description:

The function is used to Get all tags information in memery..

Usage:

Int GetTagBufferInfo(unsigned char *ComAdr, unsigned char* Data,int dataLength, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Data: Output array. dataLength bytes,all tags information in memery.each tag information include:4 bytes first time+4 bytes last time+2 bytes read times+1 byte antenna + 1byte EPC length+EPC.

dataLength: Output,one byte,the length of dataLength.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.25) ClearTagBuffer ():

Function description:

The function is used to clear tags information in memory..

Usage:

```
Int ClearTagBuffer(unsigned char *ComAdr, int FrmHandle);
```

Parameter:

ComAdr: Pointed to the address of the reader.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.26) SetRelay():

Function description:

The function is used to set relay release time.

Usage:

```
Int SetRelay(unsigned char *ComAdr, unsigned char RelayTime, int FrmHandle);
```

Parameter:

ComAdr: Pointed to the address of the reader.

RelayTime: Input, Relay release time.(0~255)*50ms.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.27) SetGPIO():

Function description:

The function is used to set PGIO output.

Usage:

```
Int SetGPIO(unsigned char *ComAdr, unsigned char OutputPin, int FrmHandle);
```

Parameter:

ComAdr: Pointed to the address of the reader.

OutputPin: Input,GPIO output statue.bit0-bit7Respectively controlling GPIO0-GPIO7.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.28) GetGPIOStatus():

Function description:

The function is used to Get PGIO output.

Usage:

```
Int GetGPIOStatus(unsigned char *ComAdr, unsigned char *OutputPin, int FrmHandle);
```

Parameter:

ComAdr: Pointed to the address of the reader.

OutputPin: Output,GPIO output statue.bit0-bit7Respectively controlling GPIO0-GPIO7.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.29) SetNotificationPulseOutput():

Function description:

The function is used to Set Whether the corresponding ports output a 2ms low pulse when query one tag.

Usage:

Int SetNotificationPulseOutput(unsigned char *ComAdr, unsigned char OutputRep, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

OutputRep: Input,

Bit0: corresponding antenna 1 and Out1 pins.:

Bit0 = 1, when antenna 1 query one tag, Out1 pins output a 2ms low pulse,

Bit0 = 0, no signal output;

Bit1: corresponding antenna 2 and Out2 pins.:

Bit1 = 1, when antenna 2 query one tag, Out2 pins output a 2ms low pulse,

Bit1 = 0, no signal output;

Bit2: corresponding antenna 3 and Out3 pins.:

Bit2 = 1, when antenna 3 query one tag, Out3 pins output a 2ms low pulse,

Bit2 = 0, no signal output;

Bit3: corresponding antenna 4 and Out4 pins.:

Bit3 = 1, when antenna 4 query one tag, Out4 pins output a 2ms low pulse,

Bit3 = 0, no signal output;

Other bits reserved, the default value is 0.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.30) SetTriggerTime ():

Function description:

The function is used to set Trigger time.

Usage:

Int SetTriggerTime(unsigned char *ComAdr, unsigned char TriggerTime, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

TriggerTime: Input, Trigger time,range (0~255)*1s.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.31) SetTIDParameter ():

Function description:

The function is used to set TID's parameter when query TID.

Usage:

Int SetTIDParameter(unsigned char *ComAdr, unsigned char TIDAddr, unsigned char TIDLen, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

TIDAddr: Input, read TID's start address.

TIDLen: Input, read TID's word number.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.32) OpenNetPort ():

Function description:

The function is used to open net port.

Usage:

Int OpenNetPort(int Port, LPSTR IPAddr, unsigned char *ComAdr, int FrmHandle);

Parameter:

Port: Pointed to the net port of the reader.

IPAddr: Pointed to string of reader IP.

ComAdr: Pointed to the address of the reader.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function OpenNetPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.33) CloseNetPort ():

Function description:

The function is used disconnected net port.

Usage:

Int CloseNetPort (int FrmHandle);

Parameter:

FrmHandle: Handle of the corresponding communication net port the device is connected. The handle value is got when calling function OpenNetPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.34) ChangeATMode ():

Function description:

The function is used to change reader to AT mode or exit.

Usage:

Int ChangeATMode(unsigned char *ComAdr, unsigned char ATMode, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

ARMode: Input,0x01 AT mode,0x00,exit AT mode.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.1.35) TransparentCMD ():

Function description:

The function is used to send AT command to reader.

Usage:

Int TransparentCMD(unsigned char *ComAdr, unsigned char timeout, unsigned char cmdlen, unsigned char* cmdData, unsigned char recvlen, unsigned char* recvData, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Timeout:Input,time out (1~255)*10ms.

Cmdlen: Input,AT command length.

cmdData:Input array. AT command.

Recvlen:Output:receive ATcommand length.

recvData: Output array,receive AT command.

ARMode: Input,0x01 AT mode,0x00,exit AT mode.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred

3.1.36) GetSeriaNo ():

Function description:

The function is used to get reader's serial number .

Usage:

Int GetSeriaNo(unsigned char *ComAdr, unsigned char *SeriaNo,int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

SeriaNo: Output array,4 bytes serial number.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred

3.2) EPCC1-G2 Function:

3.2.1) Inventory_G2 ():

Function description:

The function is used to detect tags in the inductive area and get their EPC values.

Usage:

Int Inventory_G2(unsigned char *ComAdr, unsigned char QValue, unsigned char Session, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, unsigned char MaskFlag, unsigned char AdrTID, unsigned char LenTID, unsigned char TIDFlag, unsigned char Target, unsigned char InAnt, unsigned char Scantime, unsigned char Fastflag, unsigned char * EPClenandEPC,, unsigned char *Ant, int * Totallen, int *CardNum,int FrmHandle);

Parameter:

ComAdr: Input, pointed to the address of the reader.

QValue: Input variable, One byte, Q value,

Bit7: Statistical data packets flag;

0: not send the flag packets when query finished.

1: send the flag packets when query finished.

Bit6-Bit0: range is 0-15. Q value setting should be field tag quantity is approximately equal to 2 Q.

other value return parameter error.

Session: Input variable, One byte, Session value,

0x00:S0;

0x01:S1;

0x02:S2;

0x03:S3;

0xFF: Auto config session. (Only EPC query effective).

MaskMem: Input variables, one byte, Mask memory.

0x01:EPC;

0x02:TID;

0x03:User.

MaskAdr: Input Array, two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen: Input variables, one bytes, the bit length of Mask.

MaskData: Input Array, the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

MaskFlag: Input, EPC masking Flag.

0x00:disabled;

0x01:enabled;

AdrTID: Input value, one byte, read first address of TID.

LenTID: Input value, one byte, read length of TID.

TIDFlag: Input value, one byte, the flag of read EPC or TID.

0x01: read TID of tag;

0x00:read EPC of tag.

Target: Input value, one byte. the Target value when query tag

0x00:Target is A.

0x01:Target is B.

InAnt: Input value, one byte. which ant used to query.

0x80:ant 1;

0x81:ant 2;

0x82:ant 3;

0x83:ant 4;

Scantime: Input value,one byte.the range is 3 to 255;

Fastflag: Input value,one byte.;

0:disabled

1:enabled;

EPCLenandEPC: Output, Pointed to the array storing the inventory result. It is the EPC data of tag Reader read. The unit of the array includes 1 byte EPCLen and N (the value of EPCLen) bytes EPC + 1 byte Reserve. It is the former high-word, low word in the EPC of each tag. It is the former high-byte, low byte in the each word.

Ant: Output,the antenna query tag.For example:0x04,binary is 00000100,means antenna 3 query tag.

Totallen: Output. Pointed to the byte count of the EPCLenandEPC.

CardNum: Output. Pointed to the number of tag detected.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

None zero value when successfully, value:

0x01 Return before Inventory finished

0x02 the Inventory-scan-time overflow

0x26 inventory finished and return rate.

others when error occurred.

3.2.2) ReadData_G2 ():

Function description:

The function is used to read part or all of a Tag's Password, EPC, TID, or User memory. To the word as a unit, start to read data from the designated address.

Usage:

Int ReadData_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char * Password , unsigned char MaskMem, unsigned char*MaskAdr, unsigned char MaskLen, unsigned char*MaskData, unsigned char * Data ,unsigned char * errorcode,int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Enum: Input,when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and mask.. if Enum=0xFF,not mask.

Mem: Input, Pointed to select the memory area to read.

0x00: Password area;

0x01: EPC memory area;

0x02: TID memory area;

0x03: User's memory area;

Other value when error occurred.

WordPtr: Input, Pointed to the address of tag data to read (Word/Hex). Such as, 0x00 stand in

start to read data from first word, 0x01 stand in start to read data from second word, and so on.
Num: Input, Pointed to the number of word to read. Can not set 0 or 120, otherwise, return the parameter error information. Num <= 120

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

MaskMem: Input variables, one byte, Mask memory.

0x01: EPC;

0x02: TID;

0x03: User.

MaskAdr: Input Array, two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen: Input variables, one bytes, the bit length of Mask.

MaskData: Input Array, the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Data: Output. Pointed to the array of the data read from tag.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, the Read data is in **Data**, non-zero value when error occurred.

3.2.3) WriteData_G2 ():

Function description:

The function is used to write several words in a Tag's Reserved, EPC, TID, or User memory.

Usage:

Int WriteData_G2(unsigned char *ComAdr, unsigned char * EPC, unsigned char Wnum, unsigned char Enum, unsigned char Mem, unsigned char WordPtr, unsigned char *Writedata, unsigned char * Password, unsigned char MaskMem, unsigned char * MaskAdr, unsigned char MaskLen, unsigned char * MaskData, unsigned char * errorcode, int FrmHandle)

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Wnum: Input, the length of Writedata.

Enum: Input, when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and mask.. if Enum=0xFF, not mask.

Mem: Input, Pointed to select the memory area to read.

0x00: Password area

0x01: EPC memory area

0x02: TID memory area

0x03: User's memory area

Other value when error occurred.

WordPtr: Input, Pointed to the starting address of tag data to write (Word/Hex). If write in the EPC area, it will ignore the start address, and start to write at the address 0x02.

Writedata: Input, Pointed to the array of the word to be written. For example, WordPtr equal 0x02, then the first word in Data write in the address 0x02 of designated Mem, the second word write in 0x03, and so on.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

MaskMem: Input variables, one byte, Mask memory.

0x01: EPC;

0x02: TID;

0x03: User.

MaskAdr: Input Array, two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen: Input variables, one bytes, the bit length of Mask.

MaskData: Input Array, the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.4) BlockErase_G2 ():

Function description:

The function is used to erase multiple words in a Tag's Password, EPC, TID, or User memory.

Usage:

```
Int BlockErase_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum, unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char *Password, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char * MaskData, unsigned char * errorcode, int FrmHandle);
```

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Enum: Input, when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and mask.. if Enum=0xFF, not mask.

Mem: Input, Pointed to select the memory area to read.

0x00: Password area

0x01: EPC memory area

0x02: TID memory area

0x03: User's memory area

Other value when error occurred.

WordPtr: Input, Pointed to the address of tag data to erase (Word/Hex). Such as, 0x00 stand in start to erase data from first word, 0x01 stand in start to erase data from second word, and so on. When erase EPC area, **WordPtr** must be greater than or equal to 0x02. Otherwise, return the parameter error information.

Num: Input, Pointed to the number of word to erase. Can not set 0, otherwise, return the

parameter error information.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

MaskMem: Input variables, one byte, Mask memory.

0x01: EPC;

0x02: TID;

0x03: User.

MaskAdr: Input Array, two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen: Input variables, one byte, the bit length of Mask.

MaskData: Input Array, the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.5) Lock_G2 ():

Function description:

The function is used to set Password area as readable and writeable from any state, readable and writeable from the secured state, permanently readable and writeable, never readable and writeable. It used to set EPC, TID or User as writeable from any state, writeable from the secured state, permanently writeable, never writeable.

Usage:

Int Lock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum, unsigned char select, unsigned char setprotect, unsigned char * Password, unsigned char MaskMem, unsigned char* MaskAdr, unsigned char MaskLen, unsigned char * MaskData, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Enum: Input, when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and mask.. if Enum=0xFF, not mask.

Select: Input.

0x00, Control kill password protection setting.

0x01, Control access password protection setting.

0x02, Control EPC memory protection setting.

0x03, Control TID memory protection setting.

0x04, Control User memory protection setting.

Other value when error occurred.

Setprotect: Input.

When **Select** is 0x00 or 0x01, **SetProtect** means as follows:

0x00: readable and writeable from any state.

0x01: permanently readable and writeable.

0x02: readable and writeable from the secured state.

0x03: never readable and writeable

When **Select** is 0x02, 0x03 or 0x04, **SetProtect** means as follows:

0x00: writeable from any state.

0x01: permanently writeable.

0x02: writeable from the secured state.

0x03: never writeable.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

MaskMem: Input variables, one byte, Mask memory.

0x01: EPC;

0x02: TID;

0x03: User.

MaskAdr: Input Array, two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen: Input variables, one byte, the bit length of Mask.

MaskData: Input Array, the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.6) KillTag_G2 ():

Function description:

The function is used to destroy tag. After the tag destroyed, it never process command.

Usage:

Int KillTag_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum, unsigned char * Password, unsigned char MaskMem, unsigned char* MaskAdr, unsigned char MaskLen, unsigned char * MaskData, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Enum: Input, when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and mask.. if Enum=0xFF, not mask.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

MaskMem: Input variables, one byte, Mask memory.

0x01: EPC;

0x02: TID;

0x03: User.

MaskAdr: Input Array, two bytes, the start bit's address (unit: Bits) of Mask, high byte in

advance. Range 0 ~ 16383.

MaskLen: Input variables, one bytes, the bit length of Mask.

MaskData: Input Array, the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.7) WriteEPC_G2 ():

Function description:

The function is used to write EPC value in a Tag's EPC memory. Random write one tag in the antenna.

Usage:

Int WriteEPC_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * WriteEPC, unsigned char Enum, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

WriteEPC: Input, Pointed to the array of the new tag's EPC value to overwrite the old tag's EPC value.

Enum: Input, the word number of EPC length.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.8) SetPrivacyByEPC_G2 ():

Function description:

The function is used to set designated tag read protection. After the tag destroyed, it never process command. Even if inventory tag, reader can not get the EPC number. Only NXP's UCODE EPC G2X tags valid.

Usage:

Int SetPrivacyByEPC_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum, unsigned char * Password, unsigned char MaskMem, unsigned char * MaskAdr, unsigned char MaskLen, unsigned char * MaskData, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Enum: Input, when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and

mask.. if Enum=0xFF,not mask.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

MaskMem:Input variables,one byte,Mask memery.

0x01:EPC;

0x02:TID;

0x03:User.

MaskAdr:Input Array,two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen:Input variables,one bytes, the bit length of Mask.

MaskData:Input Array,the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.9) SetPrivacyWithoutEPC_G2 ():

Function description:

The function is used to random set one tag read protection in the antenna.The tag must be have the same access password.Only NXP's UCODE EPC G2X tags valid.

Usage:

Int SetPrivacyWithoutEPC_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.10) ResetPrivacy_G2 ():

Function description:

The function is used to reset only one tag read protection in the antenna.The tag must be have the same access password.Only NXP's UCODE EPC G2X tags valid.

Usage:

Int ResetPrivacy_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

Note: If tag does not support the read protection setting, it must be unprotected.

3.2.11) CheckPrivacy_G2 ():**Function description:**

The function is used to check only one tag in the antenna whether the tag is protected. It can not check the tag whether the tag support protection setting. Only NXP's UCODE EPC G2X tags valid.

Usage:

Int CheckPrivacy_G2 (unsigned char *ComAdr, unsigned char *Readpro, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

Readpro: Output.

0x00 Tag is unprotected

0x01 Tag is protected

Note: If tag does not support the read protection setting, it must be unprotected.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

Note: If tag does not support the read protection setting, it must be unprotected.

3.2.12) EASConfigure_G2 ():**Function description:**

The function is used to set or reset the EAS status bit of designated tag. Only NXP's UCODE EPC G2X tags valid.

Usage:

Int EASConfigure_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum, unsigned char * Password, unsigned char EAS, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char * MaskData, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Enum: Input, when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and mask.. if Enum=0xFF, not mask.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

EAS: Input. The bit0 of **EAS** is 0, it means setting EAS closed.

The bit0 of **EAS** is 1, it means setting EAS opened.

bit1 ~ bit7 default is 0.

MaskMem: Input variables, one byte, Mask memory.

0x01: EPC;

0x02: TID;

0x03: User.

MaskAdr: Input Array, two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen: Input variables, one byte, the bit length of Mask.

MaskData: Input Array, the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.13) EASAlarm_G2 ():

Function description:

The function is used to check EAS status bit of any tag in the antenna. Only NXP's UCODE EPC G2X tags valid.

Usage:

Int EASAlarm_G2 (unsigned char *ComAdr, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when EAS alarm,

0xFB value when no EAS alarm,

Otherwise, return other value when error occurred.

3.2.14) BlockLock_G2 ():

Function description:

The function is used to permanently lock the designated data in designated tag's user memory. The locked data can be read only, but not written and not erased. Only NXP's UCODE EPC G2X tags valid.

Usage:

```
Int BlockLock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char
Enum,unsigned char * Password, unsigned char WrdPointer, unsigned char
MaskMem,unsigned char *MaskAdr,unsigned char MaskLen,unsigned char *
MaskData,unsigned char * errorcode,int FrmHandle);
```

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Enum: Input,when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and mask.. if Enum=0xFF,not mask.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

WordPointer: Input. Pointed to the address to lock:

0 or 1	permanently lock	block data 0 and 1
2 or 3	permanently lock	block data 2 and 3
4 or 5	permanently lock	block data 4 and 5
6 or 7	permanently lock	block data 6 and 7
8 or 9	permanently lock	block data 8 and 9
10 or 11	permanently lock	block data 10 and 11
12 or 13	permanently lock	block data 12 and 13

MaskMem:Input variables,one byte,Mask memery.

0x01:EPC;

0x02:TID;

0x03:User.

MaskAdr:Input Array,two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen:Input variables,one bytes, the bit length of Mask.

MaskData:Input Array,the Most number is 32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.2.15) BlockWrite_G2 ():**Function description:**

The function is used to write words in a Tag's Reserved, EPC, TID, or User memory.

Usage:

```
Int WriteData_G2(unsigned char *ComAdr, unsigned char * EPC, unsigned char Wnum,
```

unsigned char Enum,unsigned char Mem, unsigned char WordPtr,unsigned char *Writedata,unsigned char * Password, unsigned char MaskMem,unsigned char * MaskAdr,unsigned char MaskLen,unsigned char * MaskData,unsigned char * errorcode,int FrmHandle)

Parameter:

ComAdr: Input. Pointed to the address of the reader.

EPC: Input, Pointed to the array of tag's EPC value. It is the EPC number of tag.

Wnum: Input,the length of Writedata.

Enum: Input,when range is 0x00 to 0x0F, word for the unit. EPC length is smaller 15 words and mask.. if Enum=0xFF,not mask.

Mem: Input, Pointed to select the memory area to read.

0x00: Password area

0x01: EPC memory area

0x02: TID memory area

0x03: User's memory area

Other value when error occurred.

WordPtr: Input, Pointed to the starting address of tag data to write (Word/Hex).If write in the EPC area, it will ignore the start address, and start to write at the address 0x02.

Writedata: Input, Pointed to the array of the word to be written. For example, WordPtr equal 0x02, then the first word in Data write in the address 0x02 of designated Mem, the second word write in 0x03, and so on.

Password: Input, Pointed to the 8 bytes of tag's accesspassword value. From left to right it is the former high-word, low word in the accesspassword.

MaskMem:Input variables,one byte,Mask memery.

0x01:EPC;

0x02:TID;

0x03:User.

MaskAdr:Input Array,two bytes, the start bit's address (unit: Bits) of Mask, high byte in advance. Range 0 ~ 16383.

MaskLen:Input variables,one bytes, the bit length of Mask.

MaskData:Input Array,the Most number is32 bytes. MaskData data byte length is MaskLen / 8. If MaskLen not 8 integer times, then MaskData length of data for [MaskLen/8] integer add 1. Not enough in low fill 0.

Errorcode: Output, Pointed to an explanation byte when the function return value equals 0xFC.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.3) 18000-6B Function:

3.3.1) InventorySingle_6B ():

Function description:

The function is used to detect only one tag in the inductive area and get their ID values.If more than one tag in the inductive area at the same time, reader may be detect nothing.

Usage:

Int InventorySingle_6B (unsigned char *ComAdr, unsigned char *Ant,unsigned char * ID_6B , int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

Ant: Output,the antenna query tag.For example:0x04,binary is 00000100,means antenna 3 get tag.

ID_6B: Output, Pointed to the array storing the inventory result. The array is 9 bytes ID.The first byte is ID length,the other is ID.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.3.2) InventoryMultiple_6B ():**Function description:**

The function is used to according to the given conditions detect tags in the inductive area and get their ID values.

Usage:

Int InventoryMultiple_6B (unsigned char *ComAdr, unsigned char * Condition , unsigned char StartAddress, unsigned char mask , unsigned char * ConditionContent , unsigned char *Ant ,unsigned char * ID_6B , int * Cardnum, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

Condition: Input, the condition of detecting tags.

0x00: equal to condition.

0x01: unequal to condition.

0x02: greater than to condition.

0x03: less than to condition.

StartAddress: Input, the tag's start address to compare.

Mask: Input, mask. Pointed to the data is used to compare. Highest bit in the mask correspond with the far-left byte in the ConditionContent.The corresponding bit in the mask is 1 to compare the bit in the ConditionContent with the corresponding byte in the tag.The corresponding bit in the mask is 0, not compare.

ConditionContent: Input, Pointed to the array is used to compare.The array is 8 bytes

Ant: Output,the antenna query tag.For example:0x04,binary is 00000100,means antenna 3 get tag.

ID_6B: Output, Pointed to the array storing the inventory result. The unit of the array is 9 bytes ID length+ID.

Cardnum: Output, the count of tag

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, value:

0x15 Return before Inventory finished

0x16 the Inventory-scan-time overflow
0x17 More Data
0x18 Reader module MCU is Full
others when error occurred.

3.3.3) ReadData_6B ():

Function description:

The function is used to start to read several bytes from the designated address.

Usage:

Int ReadData_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char StartAddress, unsigned char Num, unsigned char * Data, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

ID_6B: Input, Pointed to the array storing the inventory result. The unit of the array is 8 bytes ID.

StartAddress: Input, the start address to read data. Range is 8~223. If the address more than range, reader will return parameter error information.

Num: Input, pointed to the number of bytes to read, range is 1~32. If $\text{StartAddress} + \text{Num}$ greater than 224, or Num greater than 32 or is zero, reader will return parameter error information.

Data: Output, Pointed to the array storing the read result.

Errorcode: Output, reservation.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.3.4) WriteData_6B ():

Function description:

The function is used to start to write several bytes from the designated address.

Usage:

Int WriteData_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char StartAddress, unsigned char * Writedata, unsigned char Writedatalen, unsigned char * writtenbyte, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

ID_6B: Input, Pointed to the array storing the inventory result. The unit of the array is 8 bytes ID.

StartAddress: Input, pointed to the start address to read data. Range is 8~223. If the address more than range, reader will return parameter error information.

Writedata: Input, pointed to the array to write, range is 1~32. If $\text{Address} + \text{Writedatalen}$ greater than 224, or Writedata greater than 32 or is zero, reader will return parameter error information. The high bytes of Writedata write in the low address in tag.

Writedatalen: Input, pointed to the number of bytes to write.

Writtenbyte: Output. Pointed to the number of bytes written successfully start from the high byte in the **Writedata**.

Errorcode: Output, reservation.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function **AutoOpenComPort** or **OpenComPort**.

Returns:

Zero value when successfully, non-zero value when error occurred.

Note: If tag does not support the read protection setting, it must be unprotection.

3.3.5) CheckLock_6B ():

Function description:

The function is used to check whether the designated byte is locked.

Usage:

Int CheckLock_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char Address, unsigned char * ReLockState, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input, pointed to the address of the reader.

ID_6B: Input, Pointed to the array storing the inventory result. The unit of the array is 8 bytes ID.

Address: Input. Pointed to the address is used to check whether is locked. Range is 0~223. Beyond this range, reader will return parameter error.

ReLockState: Output.

0x00, the byte is unlocked.

0x01, the byte is locked, cannot lock it again.

Errorcode: Output, reservation.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function **AutoOpenComPort** or **OpenComPort**.

Returns:

Zero value when successfully, non-zero value when error occurred.

3.3.6) Lock_6B ():

Function description:

The function is used to lock the designated byte.

Usage:

Int Lock_6B (unsigned char *ComAdr, unsigned char * ID_6B , unsigned char Address, unsigned char * errorcode, int FrmHandle);

Parameter:

ComAdr: Input, pointed to the address of the reader.

ID_6B: Input, Pointed to the array storing the inventory result. The unit of the array is 8 bytes ID.

Address: Input, pointed to the address to lock. Range is 8~223. Beyond this range, reader will return parameter error.

Errorcode: Output, reservation.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function **AutoOpenComPort** or **OpenComPort**.

Returns:

Zero value when successfully, non-zero value when error occurred.

4. Return Value Definition

```
#define InventoryReturnEarly_G2      0x01  Return before Inventory finished
```

#define InventoryTimeOut_G2	0x02	the Inventory-scan-time overflow
#define InventoryMoreData_G2	0x03	More Data
#define ReadermoduleMCUFull_G2	0x04	Reader module MCU is Full
#define AccessPasswordError	0x05	Access password error
#define DestroyPasswordError	0x09	Destroy password error
#define DestroyPasswordCannotZero	0x0a	Destroy password error cann't be Zero
#define TagNotSupportCMD	0x0b	Tag Not Support the command
#define AccessPasswordCannotZero	0x0c	Use the commmand,Access Password Cann't be Zero
#define TagProtectedCannotSetAgain	0x0d	Tag is protected,cannot set it again
#define TagNoProtectedDonotNeedUnlock	0x0e	Tag is unprotected,no need to reset it
#define ByteLockedWriteFail	0x10	There is some locked bytes,write fail
#define CannotLock	0x11	can not lock it
#define LockedCannotLockAgain	0x12	is locked,cannot lock it again
#define ParameterSaveFailCanUseBeforeNoPower	0x13	Save Fail,Can Use Before Power
#define CannotAdjust	0x14	Cannot adjust
#define InventoryReturnEarly_6B	0x15	Return before Inventory finished
#define InventoryTimeOut_6B	0x16	Inventory-Scan-Time overflow
#define InventoryMoreData_6B	0x17	More Data
#define ReadermoduleMCUFull_6B	0x18	Reader module MCU is full
#define NotSupportCMDOrAccessPasswordCannotZero	0x19	Not Support Command Or AccessPassword Cannot be Zero
#define CMDExecuteErr	0xF9	Command execute error
#define GetTagPoorCommunicationCannotOperation	0xFA	Get Tag,Poor Communication,Inoperable
#define NoTagOperation	0xFB	No Tag Operable
#define TagReturnErrorCode	0xFC	Tag Return ErrorCode
#define CMDLengthWrong	0xFD	Command length wrong
#define IllegalCMD	0xFE	Illegal command
#define ParameterError	0xFF	Parameter Error
#define CommunicationErr	0x30	Communication error
#define RetCRCErr	0x31	CRC checksumat error
#define RetDataErr	0x32	Return data length error
#define CommunicationBusy	0x33	Communication busy
#define ExecuteCmdBusy	0x34	Busy,command is being executed
#define ComPortOpened	0x35	ComPort Opened
#define ComPortClose	0x36	ComPort Closed
#define InvalidHandle	0x37	Invalid Handle
#define InvalidPort	0x38	Invalid Port
#define RecmdErr	0xEE	Return command error

5.ErrorCode Definition

#define	OtherError	0x00	Other error
#define	MemoryOutPcNotSupport	0x03	Memory out or pc not support
#define	MemoryLocked	0x04	Memory Locked and unwritable
#define	NoPower	0x0b	No Power,memory write operation cannot be executed
#define	NotSpecialError	0x0f	Not Special Error,tag not support special errorcode