UHFPrimeREADER.DLL

Dynamic Link Library User Guide

V1.0

Contents

Contents 2
1.Introduction 4
2.Application Integration 5
3.API 6
3.1 Connect/Close Reader 6
3.1.1 OpenDevice 6
3.1.1 OpenNetConnection 6
3.1.2 CloseDevice 7
3.2. 18000-6C (EPC G2) 9
3.2.1. InventoryContinue 9
3.2.2. GetTagInfo 10
3.2.3. InventoryStop 11
3.2.4. ReadTag 12
3.2.5. GetReadTagResp 13
3.2.6. WriteTag 14
3.2.7. GetTagResp 15
3.2.8. LockTag 16
3.2.9. KillTag 17
3.2.10. SetSelectMask 18
3.2.11. SetCoilPRM 19
3.2.12. GetCoilPRM 20
3.2.13. SelectOrSortSet 21
3.2.14. SelectOrSortGet 22
3.2.15. QueryCfgSet 23
3.2.16. QueryCfgGet【Query】 24
3.7. Reader Custom Comamand 25
3.7.1. GetInfo 25
3.7.1. GetDeviceInfo 26
3.7.1. GetDevicePara 27
3.7.1. SetDevicePara 28
3.7.2 RebootDevice 29
3.7.3 SetRFPower 29
3.7.4. GetRFPower 30
3.7.5. SetFreq 31
3.7.6. GetFreq 32
3.7.9. SetRFIDType 33
3.7.10. GetRFIDType 34
3.7.11. GetTemperature 35
3.7.12. SetTemperature 36
3.7.13. GetNetInfo 36
3.7.14. SetNetInfo 37

3.7.15. GetRemoteNetInfo 37
3.7.16. SetRemoteNetInfo 38
3.7.17. GetPermissonPara 38
3.7.18. SetPermissonPara 39
3.7.19. GetGpioPara 40
3.7.20. SetGpioPara 411
Appendix 1, Return value error code sheet 422
Appendix 2， ErrorCode sheet 43

1.Introduction

The SDK supports C, C++ and other languages that can call C library interfaces, Delphi, C#, VB6.0, VB.NET etc.
At present, the SDK only can support Windows Operation system (32 bits & 64 bits)

2.Application Integration

The SDK includes the file below：

File name	Application
UHFPrimeReader.dll	DLL，Include app API port

Language	Description
C#	UHF_RFID_API.cs copy to project contents, and join project, copy UHFPrimeReader.DLL to exe file output contents

3.API

3.1 Connect/Close Reader

3.1.1 OpenDevice

Definition	Int OpenDevice(HANDLE *hComm, char *ComPort, byte Baudrate);		
Description	Open serial port connected to reader, default baud rate is 115200, 8 data bits, 4 stop bits		
Parameter	Name	Type	Remarks
	hComm	HANDLE *	Return the handle connected to reader, all API operation need use this handle
	strComPort	char *	Input reader address, if don' t know real address, can input broadcast address 0xFF, after call library success, this parameter will return real address
	Baudrate	Byte	Set communication baud rate 0x00：9600； 0x01：19200；0x02：38400；0x03：57600；0x04：115200（Default）；
Return (int)	Success： 0 ； Failure： NOT 0 ； （Check return value error code sheet）		

Parameter Code	string[] ports = SerialPort.GetPortNames();
	HANDLE m_handler = HANDLE.Zero;
	int state = OpenDevice(out m_handler, port);		

3.1.1 OpenNetConnection

Definition	Int OpenNetConnection(HANDLE *hComm, char *ip, WORD port,DWORD timeoutMs)		
Description	Open network port connected to reader, the default IP address is 192.168.1.200, defaultport is 2022. use tcpclient way connect reader		
Parameter	Nmae	Type	Remarks
	hComm	HANDLE *	Return the handle connected to reader, all API operation need use this handle
	ip	char *	Input reader IP address, the reader default IP address is 192.168.1.200
	port	WORD	Input reader default port, 0~65535;
	timeoutMs	DWORD	The connection timeout period set, unit :ms。
Return (int)	Success： 0 ； Failure： NOT 0 ； （Check return value error code sheet）		
Parameter	NO		

3.1.2 CloseDevice

Definition	int CloseDevice(HANDLE hComm);		
Description	Close serial port or network port connected to reader		
Parameter	Nmae	Type	Remarks
	hComm	HANDLE	A handle connected with reader, when open serial port returned handle
Return (int)	Success: 0 ; Failure: NO 0 ; (Check return value error code sheet)		
Parameter Code	int state = UHF_RFID_API. CloseDevice (m_handler);		

3.1.3 OpenHidConnection 【Connect USB】

Definition	Int OpenHidConnection (HANDLE *hComm, WORD index)		
Illustration	Open the device corresponding to the HID device serial number and return the handle.		
Parameter	Name	Type	Remark
	hComm	HANDLE *	Returns the handle to the connection to the reader/writer, which is required for all API operations thereafter
	index	WORD	Equipment serial number
Return (int)	Success: 0 ; Fail: Not 0 ; (View the return value error code table)		
Reference Code	None		

3.1.4 CFHid_GetUsbCount 【Obtain the number of USB devices】

Definition	Int CFHid_GetUsbCount(void);		
Illustration	Obtain the HID device with a vid of 0x0483 and a pid of 0x5750, and return the number of HID devices.		
Parameter	Name	Type	Remark
	-	-	-
Return (int)	Success: Not 0 , This value is the number of HID devices; Fail: 0 ;		
Reference Code	None		

3.1.5 CFHid_GetUsbInfo 【Obtain the path of the corresponding USB device】

Definition	Int CFHid_GetUsbInfo(WORD index, char * pucDeviceInfo);		
Illustration	Obtain the address of the specified device. The device address suffix is kbd, which is the keyboard. Unable to connect.		
Parameter	Name	Type	Remark
	index	WORD	Specify the equipment serial number, which is determined by CFHide_ GetUsbCount Get;
	pucDeviceInfo	char *	Device address.
Return (int)	Success: 0 ; Fail: Not 0 ; (View the return value error code table)		
Reference Code	None		

3.2. 18000-6C (EPC G2)

3.2.1. InventoryContinue

Definition	int InventoryContinue(HANDLE hComm, BYTE btInvCount, DWORD dwInvParam);			
Description	Check whether have tag comppliant with the protocol exists in the valid range			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	A handle connected with reader
	invCount	BYTE	[in]	Inventory type：0x00： Inventory according to time: 0x03： Custom inventory， select and query parameter use SelectOrSortSet and QueryCfgSet command set
	invParam	DWORD	[in]	Inventory parameter:4 byte, unit: S, value as 0 will continue inventory When inventory type as custom inventory, max inventory 256S
Return (int)	Default no Return （Check return value error code sheet）			

Parameter Code	Int state =0;
	Int count =0;
	Int invParam =0;
	state =InventoryContinue(m_handler, count, invParam);// Set cycle inventory			

3.2.2. GetTagInfo

Definition	int GetTagInfo(HANDLE hComm, TagInfo* tag_info, WORD timeout);			
Descriptio n	Check whether have tag comppliant with the protocol exists in the valid range			
Parameter	Nmae	Type	Directio n	Remarks
	hComm	HANDLE	[in]	The handle connected to reader, all API operation need use this handle

	tag_info	TagInfo	[out]	`typedef struct`
				`{`
				`WORD reserve;`
				`SHORT rssi;`
				`BYTE antenna;`
				`BYTE channel;`
				`BYTE reserve;`
				`BYTE reserve;`
				`BYTE codeLen;`
				`BYTE code[255];`
				`}TagInfo;`

	timeout	WORD	[in]	Waiting data time, Unit:ms
Return (int)	STAT_OK: Command operation success, at the same time return inventoried other tag data. STAT_CMD_INVENTORY_STOP: finish inventory or no tag around			

Parameter Code	`TagInfo info;`
	`int state = UHF_RFID_API.GetTagUii(m_handler, out info, 1000);`			

3.2.3. InventoryStop

Definition	int InventoryStop(HANDLE hComm, WORD timeout);			
Description	Initialize the callback library After initializing the API, inventory command and mix inventory command received tag can upload by callback way			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	Timeout	WORD	[in]	Waiting data time, Unit: mS
Return	NO			
Parameter Code	`Int state = UHF_RFID_API.InventoryStop(m_handler, 10000);`			

3.2.4. ReadTag

Definition	`int ReadTag(HANDLE hComm, byte option, BYTE* accPwd, byte memBank, WORD wordPtr, WORD wordCount);`			
Description	This command read tag whole or part reserved area, EPC storage, TID storage or data in USER storage			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	option	BYTE	[in]	0x00，Retain Byte 。
	accPwd	BYTE*	[in]	4 bytes, access password, to allow tag enter safety state, default as 0x00000000.
	memBank	BYTE	[in]	1 byte， select storage area need read. 0x00 – Reserved area； 0x01 – EPC Storage area； 0x02 – TID Storage area； 0x03 –USER Storage area, other value reserve If appear other value among command, will return parameter error message
	WordPtr	WORD	[in]	1 byte, the start word address to read (word)
	wordCount	BYTE	[in]	1 byte, the data length to read (word)
Return (int)	0: Success, NOT 0: Failure，(check the return value error code sheet)			

Parameter Code	`BYTE* m_arrPwd = new BYTE* {0, 0, 0, 0};`
	`byte  memBank = 0;`
	`byte wordPtr = 0;`
	`byte wordcount = 2;`
	`int state = ReadTag(m_handler, 0, accPwd,`
	`(byte)memBank, wordPtr, wordCount);//`
	`Read the kill password`			

3.2.5. GetReadTagResp

Definition	`int GetReadTagResp(HANDLE hComm, TagResp* resp, byte wordCount, BYTE* readData, WORD timeout);`			
Description	Use this command after the ReadTag command to retrieve the tag data			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	resp	TagResp	[out]	Return read data。
				`typedef struct`
				`{`
				`    BYTE tagStatus;`
				`    BYTE antenna;`
				`    BYTE crc[2];`
				`    BYTE pc[2];`
				`    BYTE codeLen;`
				`    BYTE code[255];`
				`}TagResp;`

	wordCount	BYTE	[out]	1 byte, return the number of tag data words that read
	readData	BYTE*	[out]	The tag data that read, length is WordCount×2 bytes
	timeout	WORD	[in]	Waiting tag return time, unit: mS
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	No			

3.2.6. WriteTag

Definition	int WriteTag(HANDLE hComm, byte option, BYTE* accPwd, byte memBank, WORD wordPtr, byte wordCount, BYTE* writeData);			
Description	This command can write a few words to reserved memory, EPC storage, TID storage or USER storage			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	option	BYTE	[in]	0x00，Retain byte。
	accPwd	BYTE*	[in]	4 bytes, access password, to allow tag enter safety state, default as 0x00000000。
	memBank	BYTE	[in]	1 byte， select storage area need read 0x00 – Reserved area； 0x01 – EPC Storage area； 0x02 – TID Storage area； 0x03 –USER Storage area, other value reserve If appear other value among command, will return parameter error message
	WordPtr	WORD	[in]	2 bytes, the start word address to read
	wordCount	BYTE	[in]	1 byte, the word length to read
	Writedata	BYTE*	[in]	wordcount*2 bytes， data need to be written
Return (int)	0: success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.2.7. GetTagResp

Definition	int GetTagResp(HANDLE hComm, WORD cmd, out TagResp resp, WORD timeout);			
Description	USE this command to verify write, kill and lock commands			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	cmd	WORD	[in]	Command word, Write：0x0004; lock：0x0005; Kill：0x0006;

	resp	TagResp	[out]	`typedef struct`
				`{`
				`    BYTE tagStatus;`
				`    BYTE antenna;`
				`    BYTE crc[2];`
				`    BYTE pc[2];`
				`    BYTE codeLen;`
				`    BYTE code[255];`
				`}TagResp;`

	timeout	WORD	[in]	Waiting command respond time, unit: mS
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.2.8. LockTag

Definition	int LockTag(HANDLE hComm, BYTE* accPwd, byte erea, byte action);			
Description	This command can set reserved area as readable/writable, always readable/writable, with password readable/writable; can separate set EPC storage, TID storage and USER storage as read/writeable, always writable, with password writable, always not writable; EPC storage, TID storage or USER storage are always readable. NOTE: TID storage read only			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	accPwd	byte[4]	[in]	4 bytes, access password, to allow tag enter safety state, default as 0x00000000
	erea	BYTE	[in]	0x00 ： Kill password area ； 0x01 ： Access password area； 0x02： EPC； 0x03： TID； 0x04： User；
	action	BYTE	[in]	1 Byte ， When select as 0x00 or 0x01， SetProtect value represent the definition as below: 0x00 – Set as readable/writable 0x01 – Set as always readable/writable 0x02 – Set as with password readable/writable 0x03 – Set as non-readable/non-writable When select as 0x02 、 0x03 、 0x04 ， SetProtectvalue represent the definition as below: 0x00 – Set as writable 0x01 – Set as always writable 0x02 –Set as with password writable 0x03 –Set as always non-writable
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet）			
Parameter Code	NO			

3.2.9. KillTag

Definition	KillTag(HANDLE hComm, BYTE* accPwd);			
Description	This command is used to destroy a tag, after destroyed the tag will not respond to any more commands or inquiries. (need select tag before using this command)			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	accPwd	byte[4]	[in]	4 bytes, access password, to allow tag enter safety state, default as 0x00000000
Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.2.10. SetSelectMask

Definition	int SetSelectMask(HANDLE hComm, WORD maskPtr, byte maskBits, BYTE* mask);			
Description	This command can select tag			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	maskPtr	WORD	[in]	2 bytes, default as 0x0000。
	maskBits	byte	[in]	1 byte, need match EPC number bit length, default as 0x00 ; if maskBits as 0 , means didn't indicated tag, multiple tag operation (will operate on all tags in the area)
	mask	BYTE*	[in]	Need matching data, effective data length as length bit, if length is odd, need to add 0 to the lower part of the mask code
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO CF FF 00 07 0F 00 00 60 E2 80 68 94 00 00 40 0B 19 B6 16 01 CB D0 Select tag with EPC number as E2 80 68 94 00 00 40 0B 19 B6 16 01.			

3.2.11. SetCoilPRM

Definition	int SetCoilPRM(HANDLE hComm, byte qVal, byte reserved);			
Description	This command can set Q value size			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	qVal	byte	[in]	1 byte, Q value size have relation to tags quantity around, the number of tags is 2 to the Q
	reserved	byte	[in]	1 byte, Reserved, default as 0
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.2.12. GetCoilPRM

Definitio n	GetCoilPRM(HANDLE hComm, out byte pqVal, out byte reserved);			
Descripti on	This command is used to obtain Q value			
Paramet er	**Name**	**Type**	**Directio n**	**Remarks**
	hComm	HANDL E	[in]	Connected reader handle, all API operation need use this handle
	qVal	byte	[out]	1 byte, Q value size have relation to tags quantity around, the number of tags is 2 to the Q
	reserved	byte	[in]	1 byte, Reserved, default as 0
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Paramet er Code	NO			

3.2.13. SelectOrSortSet

Definition	int SelectOrSortSet(HANDLE hComm, byte prot, SelectSortParam param);			
Description	Check whether a tag exist that complies with protocol in the valid range			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	prot	byte	[in]	Protocol No： Set as 0x00;

	param	SelectSort Param	[in]	`typedef struct`
				`{`
				`    BYTE target;`
				`    BYTE trucate;`
				`    BYTE action;`
				`    BYTE membank;`
				`    WORD m_ptr;`
				`    BYTE len;`
				`    BYTE mask[31];`
				`}SelectSortParam;`

Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.2.14. SelectOrSortGet

Definition	int SelectOrSortGet(HANDLE hComm, byte proto, SelectSortParam param);			
Description	Check whether a tag exist that complies with protocol in the valid range			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	prot	byte	[in]	Protocol No： Set as 0x00;

	param	SelectSort Param	[out]	`typedef struct`
				`{`
				`    BYTE target;`
				`    BYTE trucate;`
				`    BYTE action;`
				`    BYTE membank;`
				`    WORD m_ptr;`
				`    BYTE len;`
				`    BYTE mask[31];`
				`}SelectSortParam;`

Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.2.15. QueryCfgSet

Definition	int QueryCfgSet(HANDLE hComm, byte proto, QueryParam param);			
Description	Check whether exist tag that comply with protocol in the valid range			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	prot	byte	[in]	Protocl No：Set as 0x00;

	param	QueryParam	[in]	`typedef struct`
				`{`
				`    BYTE condition;`
				`    BYTE session;`
				`    BYTE target;`
				`}QueryParam;`

Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.2.16. QueryCfgGet【Query】

Definition	int QueryCfgGet(HANDLE hComm, byte proto, QueryParam param);			
Description	Check whether a tag exist that complies with protocol in the valid range			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	prot	byte	[in]	Protocol No：Set as 0x00;

	param	QueryParam	[out]	`typedef struct`
				`{`
				`    BYTE condition;`
				`    BYTE session;`
				`    BYTE target;`
				`}QueryParam;`

Return (int)	0: Success, NOT 0: Failure，　(check the return value error code sheet)			
Parameter Code	NO			

3.7. Reader Custom Comamand

3.7.1. GetInfo

Definition	int GetInfo (HANDLE hComm, DeviceInfo* devInfo);			
Description	Obtain reader information, reader software version and other information			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle

	devInfo	DeviceInfo	[out]	typedef struct
				{
				    BYTE firmVersion[32];
				    BYTE hardVersion[32];
				    BYTE SN[12];
				    BYTE PARA[12];
				}DeviceInfo;
				
				struct PARA
				{
				    BYTE RFIDPRO;
				    WORD STRATFREI;
				    WORD STRATFRED;
				    WORD STEPFRE;
				    BYTE CN;
				    BYTE POWER;
				    BYTE ANTENNA;
				    BYTE REGION;
				    BYTE RESERVED;
				};

Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.1. GetDeviceInfo

Definition	int GetDeviceInfo(HANDLE hComm, DeviceFullInfo* devInfo);			
Description	Obtain reader information, reader software version and other information			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle

	devInfo	DeviceFullInfo	[out]	`typedef struct`
				`{`
				`    BYTE DevicehardVersion[32];`
				`    BYTE DevicefirmVersion[32];`
				`    BYTE DeviceSN[12];`
				`    BYTE hardVersion[32];`
				`    BYTE firmVersion[32];`
				`    BYTE SN[12];`
				`}DeviceFullInfo;`

Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	No			

3.7.1. GetDevicePara

Definition	int GetDevicePara(HANDLE hComm, DevicePara* devInfo);			
Description	Obtain reader information, reader software version and other information			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	devInfo	DevicePara	[in]	typedef struct { BYTE DEVICEARRD; BYTE RFIDPRO; BYTE WORKMODE; BYTE INTERFACE; BYTE BAUDRATE; BYTE WGSET; BYTE ANT; BYTE REGION; BYTE STRATFREI[2]; BYTE STRATFRED[2]; BYTE STEPFRE[2]; BYTE CN; BYTE RFIDPOWER; BYTE INVENTORYAREA; BYTE QVALUE; BYTE SESSION; BYTE ACSADDR; BYTE ACSDATALEN; BYTE FILTERTIME; BYTE TRIGGLETIME; BYTE BUZZERTIME; BYTE INTENERLTIME; }DevicePara;
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.7.1. SetDevicePara

Definition	int SetDevicePara(HANDLE hComm, DevicePara devInfo);			
Description	Obtain reader information, reader software version and other information			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle

	devInfo	DevicePara	[in]	typedef struct
				{
				    BYTE DEVICEARRD;
				    BYTE RFIDPRO;
				    BYTE WORKMODE;
				    BYTE INTERFACE;
				    BYTE BAUDRATE;
				    BYTE WGSET;
				    BYTE ANT;
				    BYTE REGION;
				    BYTE STRATFREI[2];
				    BYTE STRATFRED[2];
				    BYTE STEPFRE[2];
				    BYTE CN;
				    BYTE RFIDPOWER;
				    BYTE INVENTORYAREA;
				    BYTE QVALUE;
				    BYTE SESSION;
				    BYTE ACSADDR;
				    BYTE ACSDATALEN;
				    BYTE FILTERTIME;
				    BYTE TRIGGLETIME;
				    BYTE BUZZERTIME;
				    BYTE INTENERLTIME;
				}DevicePara;

Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.2 RebootDevice

Definition	int RebootDevice(HANDLE hComm);			
Description	After executing command, reader restores factory default parameters, including frequency, power, antenna enable etc.			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.3 SetRFPower

Definition	int SetRFPower(HANDLE hComm, byte power, byte reserved);			
Description	Set reader output power			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	power	BYTE	[in]	1 byte , reader power , range 0~33dBm。
	reserved	byte	[in]	Reserved
Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.4. GetRFPower

Definition	int GetRFPower(HANDLE hComm, out byte power, out byte reserved);			
Description	Obtain reader power			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle。
	power	BYTE	[out]	1 byte， reader power ， range 0~33dBm。
	reserved	byte	[out]	Reserved
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.7.5. SetFreq

Definition	int SetFreq(HANDLE hComm, ref FreqInfo frqInfo);			
Description	This command sets the upper limit and lower limit of the reader working frequency. The upper frequency must be greater than or equal to the lower frequency.			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	frqInfo	FreqInfo	[in]	`typedef struct {` `BYTE region;` `WORD StartFreq;` `WORD StopFreq;` `WORD StepFreq;` `BYTE cnt;` `}FreqInfo;`
Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

Frequency set as below：

0x00: User defined.

0x01：US [902.75~927.25]

0x02：Korea [917.1~923.5]

0x03：EU [865.1~868.1]

0x04：JAPAN [952.2~953.6]

0x05：MALAYSIA [919.5~922.5]

0x06：EU3 [865.7~867,5]

0x07：CHINA_BAND1 [840.125~844.875]

0x08：CHINA_BAND2 [920.125~924.875]

Each frequency band calculation formula：

Chinese band2：	*Fs = 920.125 + N * 0.25 (MHz) N∈[0, 19]。*
US band：	*Fs = 902.75 + N * 0.5 (MHz) N∈[0,49]。*
Korean band：	*Fs = 917.1 + N * 0.2 (MHz) N∈[0, 31]。*
EU band:	*Fs = 865.1 + N*0.2(MHz) N∈[0, 14]。*
Ukraine band:	*Fs = 868.0 + N*0.1(MHz) N∈[0, 6]。*
Chinese band1：	*Fs = 840.125 + N * 0.25 (MHz) N∈[0, 19]。*
US band3：	*Fs = 902 + N * 0.5 (MHz) N∈[0,52]。*

3.7.6. GetFreq

Definition	int SetBaudRate(BYTE * ComAddr, BYTE baud, int FrmHandle);			
Description	Get reader communication frequency band			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle

	frqInfo	FreqInfo	[in]	`typedef struct`
				`{`
				`    BYTE region;`
				`    WORD StartFreq;`
				`    WORD StopFreq;`
				`    WORD StepFreq;`
				`    BYTE cnt;`
				`}FreqInfo;`

Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.7.9. SetRFIDType

Definition	int SetRFIDType(HANDLE hComm, byte type);			
Description	This command is used to set module protocol type			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	type	byte	[in]	1 byte， 0x00：ISO 18000-6C；0x01：GB/T 29768；0x02：GJB 7377.1； At present only support ISO 18000-6C。
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.7.10. GetRFIDType

Definition	int GetRFIDType(HANDLE hComm, out byte type);			
Description	This command is used to obtain module protocol type			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	type	byte	[out]	1 byte，0x00：ISO 18000-6C； 0x01：GB/T 29768；0x02：GJB 7377.1；At present only support ISO 18000-6C。
Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.7.11. GetTemperature

Definition	int GetTemperature(HANDLE handler, out byte tempCur, out byte tempLimit);			
Description	This command is used to obtain the current temperature and threshold			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	tempCur	byte	[out]	1-byte, current temperature unit as degrees Celsius 。
	tempLimit	byte	[out]	1-byte, current temperature threshold, over this temperature, module will stop working to wait for a reduction in temperature
Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.12. SetTemperature

Definition	int SetTemperature(HANDLE handler, byte tempLimit, byte resv);			
Description	This command is used to set temperature and threshold			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	tempLimit	byte	[in]	1-byte, current temperature threshold, over this temperature, module will stop working to wait for a reduction in temperature. Generally set between 50~90℃
	resv	byte	[in]	
Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.13. GetNetInfo

Definition	int GetNetInfo(HANDLE hComm, NetInfo *type);			
Description	This command is used to obtain device network parameters			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle
	type	NetInfo	[out]	`typedef struct { BYTE IP[4]; BYTE MAC[6]; BYTE PORT[2]; BYTE NetMask[4]; BYTE Gateway[4]; }NetInfo;`
Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	No			

3.7.14. SetNetInfo

Definition	int SetNetInfo(HANDLE hComm, NetInfo type);			
Description	This command is used to set device network parameters			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	type	NetInfo	[in]	`typedef struct`
				`{`
				`BYTE IP[4];`
				`BYTE MAC[6];`
				`BYTE PORT[2];`
				`BYTE NetMask[4];`
				`BYTE Gateway[4];`
				`}NetInfo;`

Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.15. GetRemoteNetInfo

Definition	int GetRemoteNetInfo(HANDLE hComm, RemoteNetInfo *type);			
Description	This command is used to obtain remote network parameters			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle

	type	NetInfo	[out]	`typedef struct`
				`{`
				`BYTE Enable;`
				`BYTE IP[4];`
				`BYTE PORT[2];`
				`BYTE HeartTime;`
				`}RemoteNetInfo;`

Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.16. SetRemoteNetInfo

Definition	Int SetRemoteNetInfo(HANDLE hComm, RemoteNetInfo type);			
Description	This command is used to set remote network parameters			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle

	type	NetInfo	[in]	typedef struct
				{
				BYTE Enable;
				BYTE IP[4];
				BYTE PORT[2];
				BYTE HeartTime;
				}RemoteNetInfo;

Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.7.17. GetPermissonPara

Definition	Int GetPermissonPara(HANDLE hComm, PermissonPara* PermissonPara);			
Description	This command is used to obtain reading permission parameters			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle

	PermissonPara	PermissonPara	[out]	typedef struct
				{
				BYTE CodeEn;
				BYTE Code[4];
				BYTE MaskEn;
				BYTE StartAdd;
				BYTE MaskLen;
				BYTE MaskData[12];
				BYTE MaskCondition;
				}PermissonPara;

Return (int)	0: Success, NOT 0: Failure， (check the return value error code sheet)			
Parameter Code	NO			

3.7.18. SetPermissonPara

Definition	Int SetPermissonPara(HANDLE hComm, PermissonPara PermissonPara);			
Description	This command is used to set reading permission parameters			
Parameter	Name	Type	Directio n	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operations need to use this handle

	PermissonPara	PermissonPara	[in]	typedef struct
				{
				    BYTE CodeEn;
				    BYTE Code[4];
				    BYTE MaskEn;
				    BYTE StartAdd;
				    BYTE MaskLen;
				    BYTE MaskData[12];
				    BYTE MaskCondition;
				}PermissonPara;;

Return (int)	0: Success, NOT 0: Failure, (check the return value error code sheet)			
Parameter Code	NO			

3.7.19. GetGpioPara

Definition	int GetGpioPara(HANDLE hComm, GpioPara* GpioPara);			
Description	This command use to obtain GPIO parameter			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	GpioPara	NetInfo	[out]	typedef struct
				{
				BYTE KCEn;
				BYTE RelayTime;
				BYTE KCPowerEn;
				BYTE TriggleMode;
				BYTE BufferEn;
				BYTE ProtocolEn;
				BYTE ProtocolType;
				BYTE ProtocolFormat[10];
				}GpioPara;

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.7.20. SetGpioPara

Definition	Int SetGpioPara(HANDLE hComm, GpioPara GpioPara);			
Description	This command use to set GPIO parameter			
Parameter	Name	Type	Direction	Remarks
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	GpioPara	NetInfo	[in]	`typedef struct`
				`{`
				`    BYTE KCEn;`
				`    BYTE RelayTime;`
				`    BYTE KCPowerEn;`
				`    BYTE TriggleMode;`
				`    BYTE BufferEn;`
				`    BYTE ProtocolEn;`
				`    BYTE ProtocolType;`
				`    BYTE ProtocolFormat[10];`
				`}GpioPara;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

Appendix 1, Return value error code sheet

Definition		Annotation
#define STAT_OK	0x00000000	
#define STAT_PORT_HANDLE_ERR	0xFFFFFF01	Handle error, or input serial port parameter error
#define STAT_PORT_OPEN_FAILED	0xFFFFFF02	Open serial port failure
#define STAT_DLL_INNER_FAILED	0xFFFFFF03	Internal dynamic library error
#define STAT_CMD_PARAM_ERR	0xFFFFFF04	Parameter value incorrect or out of bounds, or module do not support that parameter value
#define STAT_CMD_SERIAL_NUM_EXIT	0xFFFFFF05	Serial number existed
#define STAT_CMD_INNER_ERR	0xFFFFFF06	The command execution failed due to an internal error in the module procedure
#define STAT_CMD_INVENTORY_STOP	0xFFFFFF07	Didn't inventoried tag or inventory finished
#define STAT_CMD_TAG_NO_RESP	0xFFFFFF08	Tag response timeout
#define STAT_CMD_DECODE_TAG_DATA_FAIL	0xFFFFFF09	Failed to call tag data
#define STAT_CMD_CODE_OVERFLOW	0xFFFFFF0A	Tag data exceed the max transmission length of serial port
#define STAT_CMD_AUTH_FAIL	0xFFFFFF0B	authentication failure
#define STAT_CMD_PWD_ERR	0xFFFFFF0C	Command error
#define STAT_CMD_SAM_NO_RESP	0xFFFFFF0D	SAM card no response
#define STAT_CMD_SAM_CMD_FAIL	0xFFFFFF0E	PSAM card command execute failure
#define STAT_CMD_RESP_FORMAT_ERR	0xFFFFFF0F	reader response format incorrect
#define STAT_CMD_HAS_MORE_DATA	0xFFFFFF10	Command executed successfully, but subsequent data didn't finish transfer
#define STAT_CMD_BUF_OVERFLOW	0xFFFFFF11	In data buffer overflow
#define STAT_CMD_COMM_TIMEOUT	0xFFFFFF12	Command timeout
#define STAT_CMD_COMM_WR_FAILED	0xFFFFFF13	Data to serial port failure
#define STAT_CMD_COMM_RD_FAILED	0xFFFFFF14	Read serial port data failure
#define STAT_CMD_NOMORE_DATA	0xFFFFFF15	No more data
#define STAT_DLL_UNCONNECT	0xFFFFFF16	Network connect have not been established
#define STAT_DLL_DISCONNECT	0xFFFFFF17	Network already disconnect
#define STAT_CMD_RESP_CRC_ERR	0xFFFFFF18	CRC check error

Appendix 2， ErrorCode sheet

Erroe code		Description
#defineSTAT_GB_TAG_LOW_POWER	0xFFFFF0	Tag power supply insufficient, and tag do not have enough power to completed the operation
#define STAT_GB_TAG_OPR_LIMIT	0xFFFFFF41	Insufficient tag operation permissions, unauthorized access
#define STAT_GB_TAG_MEM_OVF	0xFFFFFF42	Tag operation store overflows, or the target store does not exist
#define STAT_GB_TAG_MEM_LCK	0xFFFFFF43	The tag storage area locked, write operation or erase operation to locked unwritable storage area, to do read operation for locked unreadable storage area
#define STAT_GB_TAG_PWD_ERR	0xFFFFFF44	Tag operation command error, access command error
#define STAT_GB_TAG_AUTH_FAIL	0xFFFFFF45	Tag failed to be authenticated
#define STAT_GB_TAG_UNKNW_ERR	0xFFFFFF46	Tag operation occurred unknown error