AsReaderDesktop

SDK Interface Reference for GPIO

Asterisk, Inc.

Revision History

Version	Description	Revised by	Date
V1.0	Initial version	MRX	2019/12/19
V1.1	Add function: SetRegionWithDefaultChannel (): SetRegionAndChannel (): GetChannelList():	MRX	2021/11/05

Contents

AsReaderDesktopGPIO.dll is a dynamic link library designed to facilitate EPCC1-G2 protocol UHF tag application software development.

1 Operation System Requirement:

WINDOWS 2000/XP/7

2 Function List:

AsReaderDesktopGPIO.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 * CheckAnt, 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 SetBeepNotification(unsigned char *ComAdr, unsigned char BeepEn, int FrmHandle);

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

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

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

15. int SetSaveLen(unsigned char *ComAdr, unsigned char SaveLen, int FrmHandle);

16. int GetSaveLen(unsigned char *ComAdr, unsigned char *SaveLen, int FrmHandle);

17. int ReadBuffer_G2(unsigned char *ComAdr, int *Totallen, int *CardNum, unsigned char *pEPCList, int FrmHandle);

18. int ClearBuffer_G2(unsigned char *ComAdr, int FrmHandle);

19. int GetBufferCnt_G2(unsigned char *ComAdr, int *Count, int FrmHandle);

20. int SetTagCustomFunction(unsigned char *address, unsigned char *InlayType, int FrmHandle);

21. int SetRegionWithDefaultChannel(unsigned char* address, int region, int FrmHandle);

22. int SetRegionAndChannel(unsigned char* address, int region, int dminChannel, int dmaxChannel, int FrmHandle);

23. int GetChannelList(int region, char* fre[]);

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, int * 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, int * 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, int * 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, int * 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, int * errorcode, int FrmHandle);

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

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

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

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

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

14. 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, int * errorcode, int FrmHandle);

15. int ExtReadData_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, int * errorcode, int FrmHandle);

16. int ExtWriteData_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, int * errorcode, int FrmHandle);

17. int InventoryBuffer_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, int * BufferCount, int *TagNum, int FrmHandle);

18. int GetMonza4QTWorkParamter_G2(unsigned char *address, unsigned char *EPC, unsigned char ENum, unsigned char *Password, unsigned char

MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, unsigned char *QTcontrol, int *Errorcode, int FrmHandle);

19. int SetMonza4QTWorkParamter_G2(unsigned char *address, unsigned char *EPC, unsigned char ENum, unsigned char QTcontrol, unsigned char *Password, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, int *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 default parameters are 57600bps, 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 * CheckAnt, 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. 0x0F lines on RRU9816

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 unkown.

ScanTime: Point to the value of time limit for inventory command.

Ant：Output variable, reserve.

BeepEn: Output variable, Buzzer information.

OutputRep: Output variable, reserve.

CheckAnt: Output variable, 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 0~26

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 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.

BeepEn: Input.one byte, Beep enabled flag. 1: Enabled;0: Disabled;

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 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-bit1Respectively controlling GPIO1-GPIO2.

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 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-bit1Respectively controlling INT1-INT2, bit4-bit5Respectively controlling OUT1-OUT2.

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 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

3.1.15 SetSaveLen ():

Function Description:

The function is used to set the saved EPC/TID’s length of tag in the buffer.

Usage:

int SetSaveLen(unsigned char *ComAdr, unsigned char SaveLen, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

SaveLen: Input variables, EPC/TID length.

0x00: 128bits;

0x01: 496bits;

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.

3.1.16 GetSaveLen ():

Function Description:

The function is used to get the EPC/TID's length of tag in the buffer.

Usage:

int GetSaveLen(unsigned char *ComAdr, unsigned char *SaveLen, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

SaveLen: Output variables, EPC/TID length.

0x00: 128bits;

0x01: 496bits;

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.

3.1.17 ReadBuffer_G2 ():

Function Description:

The function is used to get data from buffer.

Usage:

int ReadBuffer_G2(unsigned char *ComAdr, int *Totallen, int *CardNum, unsigned char *pEPCList, int FrmHandle)

Parameter:

ComAdr: Pointed to the address of the reader.

Totallen: Output, the length of array pEPCList.

CardNum: Output, the tag number of this single response.

pEPCList: Output array, the EPC/TID data in the buffer.

EPC-n				
Ant	Len	EPC/TID	RSSI	Count
0x01	0xXX	nBytes	0xXX	0xXX

Ant: It describes from which antenna the tag EPC is collected. For example, this module is 0x01.

Len: EPC/TID length of this tag.

EPC/TID: the tag's EPC/TID data.length is **Len**.

RSSI: the signal strength of the tag for the first Read time

Count: The number of the successful reading tag, the value 0xFF said number greater than or equal to 255 times.

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 ClearBuffer_G2():

Function Description:

The function is used to clear tag data in the buffer.

Usage:

int ClearBuffer_G2(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.

3.1.19 GetBufferCnt_G2():

Function Description:

The function is used to get tag number in the buffer.

Usage:

int GetBufferCnt_G2(unsigned char *ComAdr, int *Count, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

Count: Output, tag number in the buffer.

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 SetTagCustomFunction ():

Function Description:

The function is used to set tag custom function.

Usage:

int SetTagCustomFunction(unsigned char *address, unsigned char *InlayType, int FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

InlayType: Input/Output.0: Not specify tag type.1: Enable peek function.

0x00: Disable

0x01: Enable

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 SetRegionWithDefaultChannel():

Function Description：

This function sets the frequency of the reader to the desired region.

Usage：

int SetRegionWithDefaultChannel(unsigned char *ComAdr, int region, int FrmHandle);

Parameter：

ComAdr: Pointed to the address of the reader.
Region: The region value of the country
Chinese band2: 1
US band : 2
Korean band: 3
EU band: 4
Japan band: 5
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function OpenUSBPort.

Returns：

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

3.1.22 SetRegionAndChannel():

Function Description：

It is used to set frequency for the reader according to the channel.

Usage：

int SetRegionAndChannel(unsigned char *ComAdr, int region, int dminChannel, int dmaxChannel, int FrmHandle);

Parameter：

ComAdr: Pointed to the address of the reader.
region: The region value of the country
Chinese band2: 1
US band: 2
Korean band: 3
EU band: 4
Japan band: 5
dminChannel: The value of the channel corresponding to the minimum frequency selected.
dmaxChannel: The value of the channel corresponding to the maxmum frequency selected.

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

Returns：

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

3.1.23 GetChannelList():

Function Description：

This command is used to get the frequency list.

Usage：

int GetChannelList(int region, char* fre[]);

Parameter：

region: The region value of the country
Chinese band2: 1
US band: 2
Korean band: 3
EU band: 4
Japan band: 5
fre: Output variable. List of frequency values.

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, range is 0-15.

Session: Input variable, one byte, Session value, range is 0-3.

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.

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.this module is 0x80;

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. 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.this module is 0x01.

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:

Zero value when successfully, value:

0x01 Return before Inventory finished

0x02 the Inventory-scan-time overflow

0x03 More Data

0x04 Reader module MCU is Full

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, int * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of 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 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.

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, int * 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.

3.2.4 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, int * 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 ease 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 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.

3.2.5 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, int * 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 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.

3.2.6 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, int * 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 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.

3.2.7 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, int * 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, int * 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 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.

3.2.9 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, int * 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, int * 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 chek 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, int * 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 unprotection.

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, int * 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 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 byte, 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.

3.2.13 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 tag is valid.

Usage:

Int EASAlarm_G2 (unsigned char *ComAdr, int * 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 BlockWrite_G2 ():

Function Description:

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

Usage:

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, int * 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.

3.2.15 ExtReadData_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 ExtReadData_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, int * errorcode, int FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of 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 array, 2bytes, pointed to the address of tag data to read (Word/Hex). Such as, 0x0000 stand in start to read data from first word, 0x0001 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 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.

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.

3.2.16 ExtWriteData_G2 ():

Function Description:

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

Usage:

Int ExtWriteData_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, int * 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 array, 2bytes, 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 0x0002.

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.

3.2.17 InventoryBuffer_G2 ():

Function Description:

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

Usage:

int InventoryBuffer_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, int * BufferCount, int *TagNum, int FrmHandle);

Parameter:

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

QValue: Input variable, one byte, Q value, range is 0-15.

Session: Input variable, one byte, Session value, range is 0-3.

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.

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, this module is 0x80.

Scantime: Input value, one byte.the range is 3 to 255(*100ms);

Fastflag: Input value, one byte;

0: disabled

1: enabled;

BufferCount: Output, total number of tag record in buffer, same EPC/TID data tag will be deemed to be the same. If not clear the buffer, tag number for many times the number of office operation accumulation.

TagNum: Output. Read tag times current query, do not distinguish whether read many times with same 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:

0x01 Return before Inventory finished

0x03 More Data

others when error occurred.

3.2.18 GetMonza4QTWorkParamter_G2 ():

Function Description:

The function is used to get tag work parameter. only effective with **Monza 4QT** of **Impinj**.

Usage:

int GetMonza4QTWorkParamter_G2(unsigned char *address, unsigned char *EPC, unsigned char ENum, unsigned char *Password, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, unsigned char *QTcontrol, int *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 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.

QTcontrol: Output.Tag work parameters.

Bit0：mirror page.bit0=0: private;bit0=1: public.

Bit1：enabled distance protect or not.bit1=0: diabled;bit1=1: enabled.

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.

3.2.19 SetMonza4QTWorkParamter_G2 ():

Function Description:

The function is used to set tag work parameter. only effective with **Monza 4QT** of **Impinj**.

Usage:

int SetMonza4QTWorkParamter_G2(unsigned char *address, unsigned char *EPC, unsigned char ENum, unsigned char QTcontrol, unsigned char *Password, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, int *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.

QTcontrol: Input.Tag work parameters.

Bit0：mirror page.bit0=0: private;bit0=1: public.

Bit1：enabled distance protect or not.bit1=0: diabled;bit1=1: enabled.

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.

4 Return Value Definition

#define InventoryReturnEarly_G2	0x01 Return before inventory finished
#define InventoryTimeOut_G2	0x02 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 failed, 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 access password cannot be zero
#define ANTCheckError	0xF8 Antenna checked error.
#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 error code
#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 checksummat 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