AsReaderBox.DLL

SDK Interface Reference V2.0

Asterisk, Inc.

Contents

1 Introduction

The SDK is designed for C, C++ and other C function interface compatible languages, such as Delphi, C#, VB6.0 and VB.NET etc. It supports both 32-bit and 64-bit WINDOWS operation system.

By loading the "ASREADERBOX.DLL", host application can control the 18000-6C and 18000-6B formats R2000 readers. Host application can access multiple readers via different communication ports simultaneously.

2 SDK summary

This SDK includes the following contents:

Files	Description
ASREADERBOX.dll	Dynamic Link Library (DLL), including all the API interfaces.
RWDev.cs	C# function class files.

Programming language	Description
C#	Copy "RWDev.cs" to project directory and add the file to project. Copy "ASREADERBOX.dll" to the exe file output directory.

3 Programming interface

3.1 Connect/disconnect reader

3.1.1 AutoOpenComPort

Definition	int AutoOpenComPort(int* port, BYTE* ComAdr, BYTE baud, int* FrmHandle);		
Description	Automatically identify reader connected serial port and initialise serial connection to reader. This function only supports serial from COM1 to COM 9.		
Parameters	**Name**	**Type**	**Notes**
	port	Int*	Serial port number for reader connection. 1 - COM1; 2 - COM2, etc.
	ComAdr	BYTE*	Target reader address. In the case when reader address is unknown, the broadcasting address 0xFF can be used to initialise the serial connection. Once the function is called successfully, the actual address of the reader will be returned as the value of this parameter.
	baud	BYTE	Serial baud rate. 0 - 9600 bps; 1 - 19200 bps; 2 - 38400 bps; 5 - 57600 bps; 6 - 115200 bps.
	FrmHandle	Int*	Reader connected handle will be returned as the value of this parameters. The value of handle will be used in all the future API operation.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).		

Sample code	int port=0;
	BYTE address=255;// broadcasting address
	BYTE baud=5;//57600
	int FrmHandle=-1;
	int fCmdRet =AutoOpenComPort(&port,&address, baud, &FrmHandle);		

3.1.2 OpenComPort

<table>
<tr><td>Definition</td><td colspan="3">int OpenComPort(int port, BYTE* ComAdr, BYTE baud, int* FrmHandle) ;</td></tr>
<tr><td>Description</td><td colspan="3">Open the serial connection to reader</td></tr>
<tr><td rowspan="5">Parameters</td><td>Name</td><td>Type</td><td>Notes</td></tr>
<tr><td>port</td><td>Int</td><td>Serial port number for reader connection.
1 - COM1;
2 - COM2, etc.</td></tr>
<tr><td>ComAdr</td><td>BYTE*</td><td>Target reader address.
In the case when reader address is unknown, the broadcasting address 0xFF can be used to initialise the serial connection. Once the function is called successfully, the actual address of the reader will be returned as the value of this parameter.</td></tr>
<tr><td>baud</td><td>BYTE</td><td>Serial baud rate.
0 - 9600 bps;
1 - 19200 bps;
2 - 38400 bps;
5 - 57600 bps;
6 - 115200 bps.</td></tr>
<tr><td>FrmHandle</td><td>Int*</td><td>Reader connected handle will be returned as the value of this parameters. The value of handle will be used in all the future API operation.</td></tr>
<tr><td>Return value (int)</td><td colspan="3">Succeed: 0;
Failed: non zero; (for detail error definition, refer to reader error code table).</td></tr>
<tr><td>Sample code</td><td colspan="3">int port=1;//open serial com1
BYTE address=255;//broadcasting address
BYTE baud=5;//57600
int FrmHandle=-1;
int fCmdRet =OpenComPort(port,&address, baud, &FrmHandle);
if(fCmdRet==0)
{
 // Reader is connected successfully
 .....
}</td></tr>
</table>

3.1.3 CloseComPort

Definition	int CloseComPort();			
Description	Close all ports to disconnect with reader and release associated resource.			
Parameters (N/A)	**Name**	**Type**	**Orientation**	**Notes**
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to error code table) .			
Sample code	N/A			

3.1.4 CloseSpecComPort

Definition	int CloseSpecComPort(int FrmHandle);		
Description	Close a specific port to disconnect with reader and release associated resource.		
Parameters	**Name**	**Type**	**Notes**
	FrmHandle	int	Reader connected handle, obtained from the OpenComPort function.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to error code table) .		
Sample code	N/A		

3.1.5 OpenNetPort

<table>
<tr><td>Definition</td><td colspan="3">Int OpenNetPort(int Port,LPCTSTR IPaddr, BYTE*ComAdr, int *Frmhandle);</td></tr>
<tr><td>Description</td><td colspan="3">Open the TCP connection to reader</td></tr>
<tr><td rowspan="5">Parameters</td><td>Name</td><td>Type</td><td>Notes</td></tr>
<tr><td>port</td><td>int</td><td>TCP port number for reader connection.</td></tr>
<tr><td>IPaddr</td><td>LPCTSTR</td><td>IP address of the reader.</td></tr>
<tr><td>ComAdr</td><td>BYTE*</td><td>Target reader address, or simply use broadcasting address 0xff for TCP connection.</td></tr>
<tr><td>FrmHandle</td><td>Int*</td><td>Reader connected handle will be returned as the value of this parameters. The value of handle will be used in all the future API operation.</td></tr>
<tr><td>Return value (int)</td><td colspan="3">Succeed: 0;
Failed: non zero; (for detail error definition, refer to reader error code table).</td></tr>
<tr><td>Sample code</td><td colspan="3">int port=6000;
LPCTSTR ipAddr =”192.168.1.190”;
BYTE ComAdr =255; // broadcasting address
int FrmHandle=-1;
int fCmdRet =OpenNetPort(port, ipAddr ,&ComAdr, &FrmHandle);
if(fCmdRet==0)
{
 // Reader is connected successfully
 .....
}</td></tr>
</table>

3.1.6 CloseNetPort

Definition	int CloseNetPort(int FrmHandle);		
Description	Close the TCP connection with reader and release associated resource		
Parameters	**Name**	**Type**	**Notes**
	FrmHandle	int	Reader connected handle, obtained from the OpenNetPort function.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).		
Sample code	N/A		

3.2 18000-6C commands

3.2.1 Inventory_G2

<table>
<tr><td>Definition</td><td colspan="4">int Inventory_G2(BYTE * ComAddr,BYTE QValue,BYTE Session,BYTE MaskMem,BYTE *MaskAdr,BYTE MaskLen,BYTE *MaskData,BYTE MaskFlag,BYTE AdrTID,BYTE LenTID,BYTE TIDFlag,BYTE Target,BYTE InAnt,BYTE Scantime,BYTE FastFlag,BYTE *pEPCList,BYTE *Ant,int *Totallen, int *CardNum,int FrmHandle);</td></tr>
<tr><td>Description</td><td colspan="4">Inquire RFID tags with corresponding protocol within the effective field.</td></tr>
<tr><td rowspan="3">Parameters</td><td>Name</td><td>Type</td><td>Orientation</td><td>Notes</td></tr>
<tr><td>ComAddr</td><td>BYTE*</td><td>[in/out]</td><td>Reader address, obtained from open port commands.</td></tr>
<tr><td>QValue</td><td>BYTE</td><td>[in]</td><td>1 byte.
bit7: Statistic data packet flag;
0 – after inventory，DO NOT deliver statistic data packet of inventory process;
1 – after inventory， deliver statistic data packet of inventory process.
bit5: FastID inventory indicator.
This function is used to rapidly obtain tag EPC and TID, only supported by some tags from the Impinj Monza series;
0 – disable;
1 – enable.
bit4 ~ bit0: the original Q-value of EPC tag inventory.
NOTE:
1. The setting of Q-value should follow the rule: 2^Q ≈ total amount of tags within the effective field. The range of Q-value is 0 ~ 15, if other value is delivered in this field, reader will return a parameter error status in the response frame.
2. Inventory with statistic data packet is SLOWER than inventory without statistic data packet.</td></tr>
<tr><td>Session</td><td>BYTE</td><td>[in]</td><td>1 byte, the Session-value of EPC tag inventory.
0x00 – apply S0 as Session value;
0x01 – apply S1 as Session value;
0x02 – apply S2 as Session value;
0x03 – apply S3 as Session value.
0xff – apply reader smart configuration (only valid in EPC inventory).</td></tr>
</table>

				Inventory for single tag or small amount of tag, S0 is the **recommended** setting.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	MaskFlag	BYTE	[in]	1 byte, mask flag. 0 – disable mask; 1 – enable mask.
	AdrTID	BYTE	[in]	1 byte, entry address of inventory TID memory.
	LenTID	BYTE	[in]	1 byte, data length for TID inventory operation, the valid range of LenTID is 0 ~ 15.
	TIDFlag	BYTE	[in]	1 byte, inventory purpose indicator. 0 – EPC inventory; 1 – TID inventory.
	Target	BYTE	[in]	1 byte, the Target value of EPC tag inventory. 0x00 – apply Target A; 0x01 – apply Target B.
	InAnt	BYTE	[in]	1 byte, antenna selection for the current inventory. 0x80 – antenna 1; 0x81 – antenna 2; 0x82 – antenna 3; 0x83 – antenna 4; InAnt is 0x08 for single port reader.
	Scantime	BYTE	[in]	1 byte, the maximum operation time for inventory. The valid range of Scantime is 0 ~ 255, corresponding to (0 ~ 255)*100ms. For Scantime = 0, operation time is not limited.
	FastFlag	BYTE	[in]	1 byte, express inventory indicator 0 – disable express inventory, Target, InAnt and Scantime are NOT essential, can be set to default value 0; 1 – enable express inventory, Target, InAnt and Scantime are need to be defined.
	pEPCList	BYTE*	[out]	The inquired tag data, the data block follows the format stated below: EPC/TID length + EPC/TID No. + RSSI.

				Data of multiple tags is formed by several identical data blocks in sequence
	Ant	BYTE*	[out]	1 byte indicates which antenna had inquired a tag. 4 antenna ports reader: Every bit in Ant represents one corresponding antenna. For example, 0x04 is 0000 0100 in binary. This indicates Antenna 3 had inquired this specific tag. 12 antenna ports reader Ant value 0 ~ 11 corresponding to Antenna 1 to Antenna 12. For example, Ant = 0 represents Antenna 1 had inquired this specific tag.
	Totallen	int*	[out]	The total length of the received data stored in pEPCList.
	CardNum	int*	[out]	The total amount of tag inquired during the current inventory. Note: Totallen and CardNum are 0 for call back return.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	0x01: inventory completed, tag data will be delivered; 0x02: inventory timeout, operation is aborted, inquired tag data will be delivered. 0xF8: error occurred in antenna check, the target antenna might be disconnected. For description on other error, refer to reader error code table.			
Sample code	N/A			

3.2.2 InventoryMix_G2

Definition	int InventoryMix_G2(BYTE * ComAddr,BYTE QValue,BYTE Session,BYTE MaskMem,BYTE *MaskAdr,BYTE MaskLen,BYTE *MaskData,BYTE MaskFlag,BYTE ReadMem,BYTE* ReadAdr,BYTE ReadLen,BYTE* Psd,BYTE Target,BYTE InAnt,BYTE Scantime,BYTE FastFlag,BYTE *pEPCList,BYTE *Ant,int *Totallen, int *CardNum,int FrmHandle);			
Description	Inquire RFID tags with corresponding protocol within the effective field, and obtain specific tag data once EPC number is inquired.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	QValue	BYTE	[in]	1 byte. **bit7**: Statistic data packet flag. 0 – after inventory，DO NOT deliver statistic data packet of inventory process; 1 – after inventory， deliver statistic data packet of inventory process. **bit6 ~ bit0**: the original Q-value of EPC tag inventory. **NOTE**: 1. The setting of Q-value should follow the rule: $2^Q \approx$ total amount of tags within the effective field. The range of Q-value is 0 ~ 15, if other value is delivered in this field, reader will return a parameter error status in the response frame. 2. Inventory with statistic data packet is **SLOWER** than inventory without statistic data packet.
	Session	BYTE	[in]	1 byte, the Session-value of inventory. 0x00 – apply S0 as Session value; 0x01 – apply S1 as Session value; 0x02 – apply S2 as Session value; 0x03 – apply S3 as Session value. Inventory for single tag or small amount of tag, S0 is the **recommended** setting.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).

	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	MaskFlag	BYTE	[in]	1 byte, mask flag. 0 – disable mask; 1 – enable mask.
	ReadMem	BYTE	[in]	1 byte, the type of target memory. 0x00 – reserved memory; 0x01 – EPC memory; 0x02 – TID memory; 0x03 – user memory. All other values are reserved, reader will return a parameter error status in the response frame if other value is delivered in this field.
	ReadAdr	BYTE*	[in]	2 bytes, the initial word address of target, most-significant byte first.
	ReadLen	BYTE	[in]	1 byte, the amount of word to be read in this operation.
	Psd	BYTE*	[in]	4 bytes, the access password.
	Target	BYTE	[in]	1 byte, the Target value of tag inventory. 0x00 – apply Target A; 0x01 – apply Target B.
	InAnt	BYTE	[in]	1 byte, antenna selection for the current inventory. 0x80 – antenna 1; 0x81 – antenna 2; 0x82 – antenna 3; 0x83 – antenna 4. InAnt is 0x08 for single port reader.
	Scantime	BYTE	[in]	1 byte, the maximum operation time for inventory. The valid range of Scantime is 0 ~ 255, corresponding to (0 ~ 255)*100ms. For Scantime = 0, operation time is not limited.
	FastFlag	BYTE	[in]	1 byte, express inventory indicator 0 – disable express inventory, Target, InAnt and Scantime are NOT essential, can be set to default value 0. 1 – enable express inventory, Target, InAnt and Scantime are need to be defined.
	pEPCList	BYTE*	[out]	The inquired tag data, the data block follows the format stated below: Rolling code + EPC length/data length + EPC No. /data + RSSI. Data of multiple tags is formed by several

				identical data blocks in sequence. The most-significant bit of rolling code: 0 – this block contains EPC; 1 – this block contains tag memory data. Firstly EPC will be uploaded, followed by tag memory data. If two sets of lower 7 bits of rolling code are consecutive, the corresponding data comes from the same tag. The lower 7 bits of rolling code ranges from 0 to 127. The code will increase for every block uploading and will recount again from 0 after it exceeds 127.
	Ant	BYTE*	[out]	1 byte indicates which antenna had inquired a tag. 4 antenna ports reader: Every bit in Ant represents one corresponding antenna. For example, 0x04 is 0000 0100 in binary. This indicates Antenna 3 had inquired this specific tag. 12 antenna ports reader Ant value 0 ~ 11 corresponding to Antenna 1 to Antenna 12. For example, Ant = 0 represents Antenna 1 had inquired this specific tag.
	Totallen	int*	[out]	The total length of the received data stored in pEPCList.
	CardNum	int*	[out]	The total amount of tag inquired during the current inventory. Note: Totallen and CardNum are 0 for call back return.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	0x01: inventory completed, tag data will be delivered; 0x02: inventory timeout, operation is aborted, inquired tag data will be delivered. 0xF8: error occurred in antenna check, the target antenna might be disconnected. For description on other error, refer to reader error code table.			
Sample code	N/A			

3.2.3 InitRFIDCallBack

Definition	void InitRFIDCallBack(pRFIDCallBack pUIDback,BOOL isBackUID,int FrmHandle);			
Description	Initialise call back function, after the initialisation of this API, tag inquired from Inventory and Mix Inventory commands can be uploaded via call back.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**

	pUIDback	pRFIDCallBack	[in]	Call back interface pointer.
				typedef struct _RFIDTag
				{
				// packet type indicator
				BYTE PacketParam;
				BYTE LEN; // length
				LPCTSTR UID; //EPC or UID
				BYTE RSSI;
				BYTE ANT;
				int FrmHandle;
				}MyStruct,*RFIDTagStruct;
				Typedefvoid(CALLBACK*pRFIDCallBack)(RFIDTagStruct lpRfid, INT32 nEvt);

	isBackUID	BOOL	[in]	Reserved.
	FrmHandle	DWORD	[in]	Reader connected handle, obtained from open port commands.
Return value	N/A			
Sample code	Refer to Examples\C#\multiple antenna test\ MutipleAntDemo			

3.2.4 ReadData_G2

Definition	int ReadData_G2(BYTE * ComAddr, BYTE *EPC, BYTE ENum, BYTE Mem, BYTE WordPtr, BYTE Num, BYTE *Password, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, BYTE *Data, int *Errorcode, int FrmHandle);			
Description	Read a part of data or the whole memory area from tag reserved memory, EPC memory, TID memory, user memory. Read operation starts from a specific address. The unit of data is word.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Mem	BYTE	[in]	1 byte, the type of target memory. 0x00 – reserved memory 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory. All other values are reserved. Reader will return a parameter error status in the response frame if other value is delivered in this field.
	WordPtr	BYTE	[in]	1 byte, the initial word address of target.
	Num	BYTE	[in]	1 byte, the amount of words to be read in this operation.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	Data	BYTE*	[out]	Inquired tag data, with a length stated in Num.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.

	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.2.5 WriteData_G2

Definition	int WriteData_G2(BYTE *ComAdr, BYTE * EPC, BYTE Wnum, BYTE Enum, BYTE Mem, BYTE WordPtr,BYTE*Writedata,BYTE* Password, BYTE MaskMem,BYTE*MaskAdr,BYTE MaskLen,BYTE * MaskData,int * errorcode,int FrmHandle);			
Description	Write multiple words to reserved memory, EPC memory, TID memory or user memory in one communication cycle.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	Wnum	BYTE	[in]	1 byte, the amount of words to be written. For ENum = 255, mask will be enabled.
	Enum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Mem	BYTE	[in]	1 byte, the type of target memory. 0x00 – reserved memory 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory. All other values are reserved. Reader will return a parameter error status in the response frame if other value is delivered in this field.
	WordPtr	BYTE	[in]	1 byte, the initial word address of target.
	Writedata	BYTE*	[in]	Data to be written to tag, with a length stated in WNum.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory;

				0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.2.6 BlockWrite_G2

Definition	int BlockWrite_G2(BYTE *ComAdr, BYTE * EPC, BYTE Wnum, BYTE Enum, BYTE Mem, BYTE WordPtr,BYTE*Writedata,BYTE* Password, BYTE MaskMem,BYTE*MaskAdr,BYTE MaskLen,BYTE * MaskData,int * errorcode,int FrmHandle);			
Description	Write multiple words to reserved memory, EPC memory, TID memory or user memory in one communication cycle.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	Wnum	BYTE	[in]	1 byte, the amount of words to be written. For ENum = 255, mask will be enabled.
	Enum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Mem	BYTE	[in]	1 byte, the type of target memory. 0x00 – reserved memory 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory. All other values are reserved. Reader will return a parameter error status in the response frame if other value is delivered in this field.
	WordPtr	BYTE	[in]	1 byte, the initial word address of target.
	Writedata	BYTE*	[in]	Data to be written to tag, with a length stated in WNum.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail

				description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.2.7 ExtReadData_G2

Definition	int ExtReadData_G2(BYTE * ComAddr, BYTE *EPC, BYTE ENum, BYTE Mem, BYTE* WordPtr, BYTE Num, BYTE *Password, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, BYTE *Data, int *Errorcode, int FrmHandle);			
Description	Read data from reserved memory, EPC memory, TID memory, user memory of tag. Read operation starts from a defined address, and the unit of data is word.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Mem	BYTE	[in]	1 byte, the type of target memory. 0x00 – reserved memory 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory. All other values are reserved. Reader will return a parameter error status in the response frame if other value is delivered in this field.
	WordPtr	BYTE*	[in]	2 bytes, the initial word address of target, most-significant byte first.
	Num	BYTE	[in]	1 byte, the amount of word to be read in this operation.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication.

				0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	Data	BYTE*	[out]	Inquired tag data, with a length stated in Num.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.2.8 ExtWriteData_G2

Definition	int WriteData_G2(BYTE *ComAdr, BYTE * EPC, BYTE Wnum, BYTE Enum, BYTE Mem, BYTE* WordPtr,BYTE*Writedata,BYTE* Password, BYTE MaskMem,BYTE*MaskAdr, BYTE MaskLen,BYTE * MaskData,int * errorcode,int FrmHandle);			
Description	Write multiple words to reserved memory, EPC memory, TID memory or user memory in one communication cycle.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	Wnum	BYTE	[in]	1 byte, the amount of words to be written.
	Enum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Mem	BYTE	[in]	1 byte, the type of target memory. 0x00 – reserved memory 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory. All other values are reserved. Reader will return a parameter error status in the response frame if other value is delivered in this field.
	WordPtr	BYTE*	[in]	2 byte, the initial word address of target.
	Writedata	BYTE*	[in]	Data to be written to tag, with a length stated in WNum.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.

	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.2.9 WriteEPC_G2

Definition	int WriteEPC_G2(BYTE * ComAddr,BYTE *Password, BYTE *EPC, BYTE ENum, int *Errorcode, int FrmHandle);			
Description	Write EPC number to a tag. During the operation, only one tag is allowed to be place in the antenna effective area.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Password	BYTE*	[in]	4 bytes, the access password.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the length of EPC number.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.2.10 BlockErase_G2

Definition	int BlockErase_G2(BYTE * ComAddr, BYTE *EPC, BYTE ENum, BYTE Mem, BYTE WordPtr, BYTE Num, BYTE *Password, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, int *Errorcode, int FrmHandle);			
Description	Erase multiple words from reserved memory, EPC memory, TID memory or user memory.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Mem	BYTE	[in]	1 byte, the type of target memory. 0x00 – reserved memory 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory. All other values are reserved. Reader will return a parameter error status in the response frame if other value is delivered in this field.
	WordPtr	BYTE*	[in]	1 byte, the initial word address of target.
	Num	BYTE	[in]	1 byte, the amount of words to be erased in this operation.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.

Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).
Sample code	N/A

3.2.11 Lock_G2

Definition	int Lock_G2(BYTE * ComAddr,BYTE *EPC, BYTE ENum, BYTE select, BYTE setprotect, BYTE *Password,BYTE MaskMem,BYTE *MaskAdr,BYTE MaskLen,BYTE *MaskData, int *Errorcode,int FrmHandle);			
Description	The read/write protection status for the following memories: • Reserved memory Readable/ writable without protection, permanently readable/ writable, readable/writeable with password protected or permanently unreadable/ unwritable; • EPC memory /user reserved memory Writable without protection, permanently writable, writeable with password protected or permanently unwritable; EPC memory, user reserved memory and TID memory are permanently readable. Furthermore TID memory is readable only, permanently unwritable.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	select	BYTE	[in]	1 byte. 0x00 – modify kill password read/write protection; 0x01 – modify access password read/write protection; 0x02 – modify EPC memory read/write protection; 0x03 – modify TID memory read/write protection; 0x04 – modify user memory read/write protection. All other values are reserved, reader will not execute the command and will return a parameter error status in the response frame if other value is delivered in this field.
	setprotect	BYTE	[in]	1 byte. **For Select = 0x00 or 0x01**: the kill password or

				access password protection setting. SetProtect = 0x00 – set to readable/ writable without protection; 0x01 – set to permanently readable/ writable; 0x02 – set to readable/writeable with password protected; 0x03 – set to permanently unreadable/ unwritable. **For Select = 0x02, 0x03, and 0x04**: the EPC, TID and user memory protection setting. SetProtect = 0x00 – set to writable without protection; 0x01 – set to permanently writable; 0x02 – set to writeable with password protected; 0x03 – set to permanently unwritable. All other values of Select and SetProtect are reserved, reader will not execute the command and will return a parameter error status in the response frame if other value is delivered in this field.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.2.12 KillTag _G2

Definition	int KillTag_G2(BYTE * ComAddr, BYTE *EPC, BYTE ENum, BYTE *Killpwd, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, int *Errorcode, int FrmHandle);			
Description	Tag killing command. After the kill operation, the tags will no longer process any command from reader.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Killpwd	BYTE*	[in]	4 bytes, the kill password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.2.13 SingleTagInventory_G2

Definition	int SingleTagInventory_G2(BYTE * ComAddr, BYTE* EPC, int *EPCLength, int *CardNum,int FrmHandle);			
Description	Inquire RFID tag with corresponding protocol within the effective field.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[out]	EPCLength+2 bytes, follows the format stated below: Antenna number + EPC No. + RSSI.
	EPCLength	int*	[out]	1 byte, the length of EPC number.
	CardNum	int*	[out]	1 byte, amount of tag inquired during the current inventory.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3 Buffer operation commands

3.3.1 InventoryBuffer_G2

Definition	int InventoryBuffer_G2(BYTE * ComAddr,BYTE QValue,BYTE Session,BYTE MaskMem,BYTE *MaskAdr,BYTE MaskLen,BYTE *MaskData,BYTE MaskFlag,BYTE AdrTID,BYTE LenTID,BYTE TIDFlag,BYTE Target,BYTE InAnt,BYTE Scantime,BYTE FastFlag ,int * BufferCount,int *CardNum,int FrmHandle);			
Description	Inquire RFID tag with corresponding protocol within the effective field.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	QValue	BYTE	[in]	1 byte. The original Q-value of EPC tag inventory. **NOTE**: 1. The setting of Q-value should follow the rule: 2^Q ≈ total amount of tags within the effective field. The range of Q-value is 0 ~ 15, if other value is delivered in this field, reader will return a parameter error status in the response frame.
	Session	BYTE	[in]	1 byte, the Session-value of EPC tag inventory. 0x00 – apply S0 as Session value; 0x01 – apply S1 as Session value; 0x02 – apply S2 as Session value; 0x03 – apply S3 as Session value. 0xff – apply reader smart configuration (only valid in EPC inventory). Inventory for single tag or small amount of tag, S0 is the **recommended** setting.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	MaskFlag	BYTE	[in]	1 byte, mask flag. 0 – disable mask;

				1 – enable mask.
	AdrTID	BYTE	[in]	1 byte, entry address of inventory TID memory.
	LenTID	BYTE	[in]	1 byte, data length for TID inventory operation, the valid range of LenTID is 0 ~ 15.
	TIDFlag	BYTE	[in]	1 byte, inventory purpose indicator. 0 – EPC inventory; 1 – TID inventory.
	Target	BYTE	[in]	1 byte, the Target value of EPC tag inventory. 0x00 – apply Target A; 0x01 – apply Target B.
	InAnt	BYTE	[in]	1 byte, antenna selection for the current inventory. 0x80 – antenna 1; 0x81 – antenna 2; 0x82 – antenna 3; 0x83 – antenna 4. InAnt is 0x08 for single port reader.
	Scantime	BYTE	[in]	1 byte, the maximum operation time for inventory. The valid range of Scantime is 0 ~ 255, corresponding to (0 ~ 255)*100ms. For Scantime = 0, operation time is not limited.
	FastFlag	BYTE	[in]	1 byte, express inventory indicator 0 – disable express inventory, Target, InAnt and Scantime are NOT essential, can be set to default value 0; 1 – enable express inventory, Target, InAnt and Scantime are need to be defined.
	BufferCount	int*	[out]	2 bytes, the total amount of tag stored in the memory buffer, tags with identical EPC/TID data will be treated as one tag. BufferCount is the sum of tag amount from multiple inventories, until the memory buffer is being cleared.
	CardNum	int*	[out]	2 bytes, the amount of tag inquired in the current inventory, a tag being accessed multiple times will also increase the amount.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	0x01: inventory completed, tag data will be delivered; 0x02: inventory timeout, operation is aborted, inquired tag data will be delivered. 0xF8: error occurred in antenna check, the target antenna might be disconnected. For description on other error, refer to reader error code table.			
Sample code	N/A			

3.3.2 ReadBuffer_G2

Definition	int ReadBuffer_G2(BYTE *ComAdr, int *Totallen, int *CardNum, BYTE *pEPCList, int FrmHandle);			
Description	Obtain all the tag information from reader memory. This command will not cause affection on the status of the stored data. Data can be access multiple times.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Totallen	int*	[out]	The total length of the received data stored in pEPCList.
	CardNum	int *	[out]	The total amount of tag inquired during the current inventory.
	pEPCList	BYTE*	[out]	The inquired tag data, with a length stated in Totallen. Data of multiple tags is formed by several identical data blocks in sequence, every data block follows the format stated below: ANT + LEN + EPC/TID + RSSI + Count;
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3.3 ClearBuffer_G2

Definition	int ClearBuffer_G2(BYTE *ComAdr, int FrmHandle);			
Description	Clear all the stored tag data from the memory buffer.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3.4 GetBufferCnt_G2

Definition	int GetBufferCnt_G2(BYTE *ComAdr, int *Count, int FrmHandle);			
Description	Obtain the total tag amount stored in memory buffer.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Count	int*	[out]	The total amount of tag stored in the memory buffer.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3.5 SetSaveLen

Definition	int SetSaveLen(BYTE *ComAdr, BYTE SaveLen, int FrmHandle);			
Description	Modify the maximum EPC/TID length for inventory with memory buffer. This modification will clear all the tag data previously stored in the memory buffer.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	SaveLen	BYTE	[in]	1 byte, the maximum EPC/TID length. 0x00 – 128bit length, i.e. 16 bytes. 0x01 – 496bit length, i.e. 62 bytes.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3.6 GetSaveLen

Definition	int GetSaveLen(BYTE *ComAdr, BYTE *SaveLen, int FrmHandle);			
Description	Load the configuration of maximum EPC/TID length for reader memory buffer			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	SaveLen	BYTE*	[out]	1 byte, the maximum EPC/TID length. 0x00 – 128bit length, i.e. 16 bytes. 0x01 – 496bit length, i.e. 62 bytes.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.4 NXP Tag commands

3.4.1 SetPrivacyByEPC _G2

Definition	int SetPrivacyByEPC_G2(BYTE * ComAddr, BYTE *EPC, BYTE ENum, BYTE *Password, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, int *Errorcode, int FrmHandle);			
Description	Setup read protection for a tag with particular EPC number; hence the tag will be unreadable by device. Device will not able to inquire EPC number of this tag via inventory operation. This command only valid for NXP UCODE EPC G2X tags.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int*	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.4.2 SetPrivacyWithoutEPC_G2

Definition	int SetPrivacyWithoutEPC_G2(BYTE * ComAddr, BYTE *Password, int *Errorcode, int FrmHandle);			
Description	Setup read protection for tags within the effective field. Unlike the pervious command in 3.4.1, this command will perform operation on multiple inquired tags without tag identification. In order to perform operation on multiple tags, it is vital to keep access password consistent on those tags.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Password	BYTE*	[in]	4 bytes, the access password.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value(int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.4.3 ResetPrivacy_G2

Definition	int ResetPrivacy_G2(BYTE * ComAddr, BYTE *Password, int *Errorcode, int FrmHandle);			
Description	Unlock read protection of a tag. Only one tag is allowed to be place in the antenna effective area.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Password	BYTE*	[in]	4 bytes, the access password.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value(int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.4.4 CheckPrivacy_G2

Definition	int CheckPrivacy_G2(BYTE * ComAddr, BYTE * readpro, int *Errorcode, int FrmHandle);			
Description	This command is NOT ABLE TO identify if a specific tag supports the read protection function, the command ONLY inquire the read protection status of the tag. For tags with no read protection function, the default status will be unlocked. Only one tag is allowed to be place in the antenna effective area.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	readpro	BYTE*	[in]	1 byte. 0x00 – read protection is disabled for the tag 0x01 – read protection is enabled for the tag Note For tags with no read protection function, the default status will be unlocked.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.4.5 EASConfigure_G2

Definition	int EASConfigure_G2(BYTE * ComAddr,BYTE *EPC,BYTE ENum,BYTE *Password,BYTE EAS, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, int *Errorcode,int FrmHandle);			
Description	Modify or reset EAS status.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the length of EPC number. For ENum = 255, mask will be enabled.
	Password	BYTE*	[in]	4 bytes, the access password.。
	EAS	BYTE	[in]	1 byte. 0x00 – EAS alert is disabled; 0x01 – EAS alert is enabled.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.4.6 EASAlarm_G2

Definition	int EASAlarm_G2(BYTE * ComAddr,int *Errorcode, int FrmHandle);			
Description	Detect EAS alert.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.5 Impinj Monza4 QT tag commands

3.5.1 GetMonza4QTWorkParamter_G2

Definition	int GetMonza4QTWorkParamter_G2(BYTE * ComAddr, BYTE *EPC, BYTE ENum, BYTE *Password, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, BYTE *QTcontrol, int *Errorcode, int FrmHandle) ;			
Description	Obtain the current working parameters of a tag.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the amount of words to be written. For ENum = 255, mask will be enabled.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	QTcontrol	BYTE*	[out]	Working parameters of tag. **bit0**: the current mirror page setting. 0 – private; 1 – public. **bit1**: distance protection setting. 0 – disabled; 1 – enabled. All other values are reserved.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return	Succeed: 0;			

value (int)	Failed: non zero; (for detail error definition, refer to reader error code table).
Sample code	N/A

3.5.2 SetMonza4QTWorkParamter_G2

Definition	int SetMonza4QTWorkParamter_G2(BYTE * ComAddr, BYTE *EPC, BYTE ENum, BYTE QTcontrol, BYTE *Password, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, int *Errorcode, int FrmHandle);			
Description	Modify the current working parameters of a tag, only valid for Monza 4QT tags from Impinj.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	EPC	BYTE*	[in]	Tag EPC number, with a length stated in Enum.
	ENum	BYTE	[in]	1 byte, the amount of words to be written. For ENum = 255, mask will be enabled.
	QTcontrol	BYTE	[in]	Working parameters of tag. **bit0**: the current mirror page setting. 0 – private; 1 – public. **bit1**: distance protection setting. 0 – disabled; 1 – enabled. All other values are reserved.
	Password	BYTE*	[in]	4 bytes, the access password.
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	ErrorCode	int *	[out]	If return value = 0xFC, ErrorCode is the error

				code returned from tag. Refer to Tag ErrorCode table for detail description.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.5.3 Inventory_QT_G2

Definition	int Inventory_QT_G2(BYTE * ComAddr, BYTE QValue, BYTE Session, BYTE Target, BYTE InAnt,BYTE Scantime, BYTE FastFlag, BYTE *pEPCList, BYTE *Ant, int *Totallen, int *CardNum, int FrmHandle);			
Description	Inquire RFID tags with corresponding protocol within the effective field.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	QValue	BYTE	[in]	1 byte. The original Q-value of EPC tag inventory. **NOTE**: 1. The setting of Q-value should follow the rule: $2^Q \approx$ total amount of tags within the effective field. The range of Q-value is 0 ~ 15, if other value is delivered in this field, reader will return a parameter error status in the response frame.
	Session	BYTE	[in]	1 byte, the Session-value of EPC tag inventory. 0x00 – apply S0 as Session value; 0x01 – apply S1 as Session value; 0x02 – apply S2 as Session value; 0x03 – apply S3 as Session value. Inventory for single tag or small amount of tag, S0 is the **recommended** setting.
	Target	BYTE	[in]	1 byte, the Target value of EPC tag inventory. 0x00 – apply Target A; 0x01 – apply Target B.
	InAnt	BYTE	[in]	1 byte, antenna selection for the current

				inventory. 0x80 – antenna 1; 0x81 – antenna 2; 0x82 – antenna 3; 0x83 – antenna 4. InAnt is 0x08 for single port reader.
	Scantime	BYTE	[in]	1 byte, the maximum operation time for inventory. The valid range of Scantime is 0 ~ 255, corresponding to (0 ~ 255)*100ms. For Scantime = 0, operation time is not limited.
	FastFlag	BYTE	[in]	1 byte, express inventory indicator 0 – disable express inventory, Target, InAnt and Scantime are NOT essential, can be set to default value 0; 1 – enable express inventory, Target, InAnt and Scantime are need to be defined.
	pEPCList	BYTE*	[out]	The inquired tag data, the data block follows the format stated below: EPC/TID length + EPC/TID No. + RSSI. Data of multiple tags is formed by several identical data blocks in sequence
	Ant	BYTE*	[out]	1 byte indicates which antenna had inquired a tag. 4 antenna ports reader: Every bit in Ant represents one corresponding antenna. For example, 0x04 is 0000 0100 in binary. This indicates Antenna 3 had inquired this specific tag. 12 antenna ports reader Ant value 0 ~ 11 corresponding to Antenna 1 to Antenna 12. For example, Ant = 0 represents Antenna 1 had inquired this specific tag.
	Totallen	int*	[out]	The total length of the received data stored in pEPCList.
	CardNum	int*	[out]	The total amount of tag inquired during the current inventory. Note: Totallen and CardNum are 0 for call back return.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	0x01: inventory completed, tag data will be delivered; 0x02: inventory timeout, operation is aborted, inquired tag data will be delivered. 0xF8: error occurred in antenna check, the target antenna might be disconnected. For description on other error, refer to reader error code table.			
Sample code	N/A			

3.5.4 SetTagCustomFunction_G2

Definition	int SetTagCustomFunction(BYTE * ComAddr, BYTE *InlayType, int FrmHandle);			
Description	launch the customised utilities of some particular tags, to achieve specific tag function			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	InlayType	BYTE*	[in/out]	1 byte, the type of tag. The valid value of this parameter is 0 ~ 254. 0 – tag type is not specified. 1 – launch the Peek function of Monza4QT tag (the status of tag will temporary change from public to private). Launching this function will make affection on the data reading/writing, block writing, protection word writing and EPC number writing operation. All other values are reserved.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.6 18000-6B Commands

3.6.1 InventorySingle_6B

Definition	int InventorySingle_6B(BYTE *address, BYTE *Ant, BYTE* ID_6B, int FrmHandle);			
Description	Inquire a single tag. If multiple tags are placed within the effective field, reader may fail to inquire any tag.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Ant	BYTE*	[out]	1 byte. For example, 0x04 is 0000 0100 in binary, indicates Antenna 3 had inquired this specific tag. For 0x08, it is 0000 1000 in binary, indicates Antenna 4 had inquired tag information.
	ID_6B	BYTE*	[out]	10 bytes, the 1st byte is the length of UID, 0x08. The 2nd ~ 9th byte is the tag UID number, least-significant byte first. The 10th byte is RSSI.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.6.2 InventoryMultiple_6B

Definition	int InventoryMultiple_6B(BYTE * ComAddr, BYTE Condition, BYTE StartAddress, BYTE mask, BYTE *ConditionContent, BYTE *Ant, BYTE* ID_6B, int *Cardnum, int FrmHandle);			
Description	Inquire tags according to specified condition.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Condition	BYTE	[in]	the condition for inventory; 0x00 – equal to the reference; 0x01 – no equal to the reference; 0x02 – larger than the reference 0x03 – smaller than the reference
	StartAddress	BYTE	[in]	1 byte, entry address of the target.
	mask	BYTE	[in]	1 byte, specify the data for comparison. Every bit in mask is corresponding to a byte in ConditionContent. The most-significant bit (bit7) corresponds to the 1st byte in ConditionContent (from left to right). Consequently, the least- significant bit (bit0) corresponds to the last byte in ConditionContent (from left to right). ConditionContent is the reference data for comparison with data in tag starts from the location stated in StartAddress. The 1st byte of ConditionContent will be compare with the data store in Address, the last byte of ConditionContent will be compare with the data store in Address+7. Set the byte corresponding bit of Mask to 1, reader will perform comparison on that particular byte. Set the bit to 0, no comparison will be performed on that particular byte.
	ConditionContent	BYTE*	[in]	8 bytes, the value of reference.
	Ant	BYTE*	[out]	1 byte. For example, 0x04 is 0000 0100 in binary, indicates Antenna 3 had inquired this specific tag. For 0x08, it is 0000 1000 in binary, indicates Antenna 4 had inquired tag information.

	ID_6B	BYTE*	[out]	Cardnum*10 bytes. UID of the tag. The length of every UID data block is 10 bytes. The 1st byte is the length value of UID number, i.e. 0x08. The 2nd ~ 9th bytes are the tag UID number. The 10th byte is the RSSI.
	Cardnum	int*	[in]	1 byte, the total amount of tag inquired during the current inventory.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	0x15: inventory completed, tag data will be delivered; 0x16: inventory timeout, operation is aborted, inquired tag data will be delivered. 0xF8: error occurred in antenna check, the target antenna might be disconnected. For description on other error, refer to reader error code table.			
Sample code	N/A			

3.6.3 ReadData_6B

Definition	int ReadData_6B(BYTE * ComAddr, BYTE* ID_6B, BYTE StartAddress, BYTE Num, BYTE *Data, int *Errorcode, int FrmHandle);			
Description	Read multiple bytes from a specific address.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	ID_6B	BYTE*	[in]	8 bytes, UID of the target tag.
	StartAddress	BYTE	[in]	1 byte, the entry address of target. The valid value range of StartAddress is 0 ~ 223. Reader will return a parameter error status in the response frame if incorrect address is delivered.
	Num	BYTE	[in]	1 byte, the amount of words to be read in this operation. The valid value range of Num is 1 ~ 32. If the sum of StartAddress and Num is larger than 224 or Num is equal to 0 or larger than 32, reader will return a parameter error status in the response frame.
	Data	BYTE*	[out]	Inquired tag data, with a length stated in Num.
	Errorcode	int*	[out]	1 byte, reserved.

	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.6.4 WriteData_6B

Definition	int WriteData_6B(BYTE * ComAddr, BYTE *ID_6B, BYTE StartAddress, BYTE *Writedata,BYTE Writedatalen, BYTE *writtenbyte, int *Errorcode, int FrmHandle);			
Description	Write multiple bytes to a specific tag.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	ID_6B	BYTE*	[in]	8 bytes, UID of the target tag.
	StartAddress	BYTE	[in]	1 byte, the entry address of target. The valid value range of StartAddress is 0 ~ 223. Reader will return a parameter error status in the response frame if incorrect address is delivered.
	Writedata	BYTE*	[in]	Data to be written to tag, with a length stated in Writedatalen.
	Writedatalen	BYTE	[in]	1 byte, the amount of words to be written in this operation.
	writtenbyte	BYTE*	[out]	1 byte, reserved.
	Errorcode	int*	[out]	1 byte, reserved.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.6.5 CheckLock_6B

Definition	int CheckLock_6B(BYTE * ComAddr, BYTE *ID_6B, BYTE StartAddress, BYTE *ReLockState, int *Errorcode, int FrmHandle);			
Description	Obtain the lock status of a specific byte.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	ID_6B	BYTE*	[in]	8 bytes, UID of the target tag.
	StartAddress	BYTE	[in]	1 byte, the entry address of target. The valid value range of StartAddress is 0 ~ 223. Reader will return a parameter error status in the response frame if incorrect address is delivered.
	ReLockState	BYTE*	[out]	1 byte. 0x00: the byte is unlocked; 0x01: the byte is already locked.
	Errorcode	int*	[out]	1 byte, reserved.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.6.6 Lock_6B

Definition	int Lock_6B(BYTE * ComAddr, BYTE *ID_6B, BYTE StartAddress, int *Errorcode, int FrmHandle);			
Description	Lock a specific byte in a tag.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	ID_6B	BYTE*	[in]	8 bytes, UID of the target tag.
	StartAddress	BYTE	[in]	1 byte, the entry address of target. The valid value range of StartAddress is 0 ~ 223. Reader will return a parameter error status in the response frame if incorrect address is delivered.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7 Reader customised commands

3.7.1 GetReaderInformation

Definition	int GetReaderInformation(BYTE* ComAdr,BYTE* VersionInfo, BYTE* ReaderType, BYTE* TrType, BYTE* dmaxfre, BYTE* dminfre, BYTE* powerdBm, BYTE* ScanTime, BYTE*Ant, BYTE*BeepEn, BYTE*OutputRep, BYTE*CheckAnt, int FrmHandle);			
Description	Obtain reader information, including reader address (ComAdr), firmware version (VersionInfo) and other reader information.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.。
	VersionInfo	BYTE*	[out]	2 bytes, firmware version. **The 1st byte**: the main version number; **The 2nd byte**: the subversion number.
	ReaderType	BYTE*	[out]	1 byte. Reader Type code.
	TrType	BYTE*	[out]	1 byte, reader supported protocol.
	dmaxfre	BYTE*	[out]	1 byte. **bit7 ~ bit6**: frequency band configuration; **bit5 ~ bit0**: current maximum frequency point.
	dminfre	BYTE*	[out]	1 byte. **bit7 ~ bit6**: frequency band configuration; **bit5 ~ bit0**: current minimum frequency point.
	powerdBm	BYTE*	[out]	1 byte, Output RF power. The range is 0 ~ 30 and the unit is dBm.
	ScanTime	BYTE*	[out]	1 byte, inventory time. Reader will respond to the inventory command delivered from host within this specific inventory time.
	Ant	BYTE*	[out]	1 byte, antenna configuration.
	BeepEn	BYTE*	[out]	1 byte, buzzer status information.
	OutputRep	BYTE*	[out]	1 byte, GPIO status for notification.
	CheckAnt	BYTE*	[out]	1 byte, Antenna check configuration.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

Frequency band configuration table

MaxFre (bit7)	MaxFre (bit6)	MinFre (bit7)	MinFre (bit6)	FreqBand
0	0	0	1	Chinese band2
0	0	1	0	US band
0	0	1	1	Korean band
0	1	0	0	EU band
0	1	1	0	Ukraine band
1	0	0	0	Chinese band1

3.7.2 SetAddress

Definition	int SetAddress(BYTE * ComAddr, BYTE ComAdrData, int FrmHandle);			
Description	Reader will apply the user provided value as its reader address and write this value to EEPROM for preservation purpose. Default setting of reader address is 0x00, the valid value of the setting is 0x00 ~ 0x0FE. Reader address will be reset to 0x00, if the provided address value is 0xFF.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	ComAdrData	BYTE	[in]	1 byte, the new address setting.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.3 SetInventoryScanTime

Definition	int SetInventoryScanTime(BYTE *address, BYTE ScanTime, int FrmHandle);			
Description	Modify maximum operation time for inventory command.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	ScanTime	BYTE	[in]	1 byte, operation time for inventory. The acceptable range of Scantime is 0 ~ 255, corresponding to (0 ~ 255)*100ms. For Scantime = 0, operation time is not limited until a whole reading cycle is completed.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.4 SetRfPower

Definition	int SetRfPower(BYTE *address, BYTE PowerDbm, int FrmHandle);			
Description	Modify reader RF power			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	PowerDbm	BYTE	[in]	1 byte, reader RF power. The range is 0 ~ 30 and the unit is dBm.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.5 SetRegion

Definition	int SetRegion(BYTE *address, BYTE dmaxfre, BYTE dminfre, int FrmHandle);			
Description	Select maximum and minimum frequency. Maximum frequency must be larger or equal the minimum frequency.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	dmaxfre	BYTE	[in]	1 byte. **bit7 ~ bit6**: frequency band configuration; **bit5 ~ bit0**: maximum frequency point.
	dminfre	BYTE	[in]	1 byte. **bit7 ~ bit6**: frequency band configuration; **bit5 ~ bit0**: minimum frequency point.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table)			
Sample code	N/A			

Frequency band configuration table

MaxFre (bit7)	MaxFre (bit6)	MinFre (bit7)	MinFre (bit6)	FreqBand
0	0	0	1	Chinese band2
0	0	1	0	US band
0	0	1	1	Korean band
0	1	0	0	EU band
0	1	1	0	Ukraine band
1	0	0	0	Chinese band1

Formulae for different frequency bands:

Chinese band2: Fs = 920.125 + N * 0.25 (MHz) where N∈ [0, 19].

US band: Fs = 902.75 + N * 0.5 (MHz) where N∈ [0, 49].

Korean band: Fs = 917.1 + N * 0.2 (MHz) where N∈ [0, 31].

EU band: Fs = 865.1 + N*0.2(MHz) where N∈ [0, 14].

Ukraine band: Fs = 868.0 + N*0.1(MHz) where N∈ [0, 6].

Chinese band1: Fs = 840.125 + N * 0.25 (MHz) where N∈ [0, 19].

3.7.6 SetBaudRate

Definition	int SetBaudRate(BYTE * ComAddr, BYTE baud, int FrmHandle);			
Description	Modify serial baud rate.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	baud	BYTE	[in]	1 byte, serial baud rate. 0 – 9600bps; 1 – 19200bps; 2 – 38400bps; 5 – 57600bps; 6 – 115200bps.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.7 BuzzerAndLEDControl

Definition	int BuzzerAndLEDControl(BYTE * ComAddr, BYTE AvtiveTime, BYTE SilentTime, BYTE Times, int FrmHandle);			
Description	Control the TAG LED and buzzer to perform specific action.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	AvtiveTime	BYTE	[in]	1 byte, TAG LED and buzzer switch ON time (ActiveT*50ms). The range of ActiveT is 0 ~ 255. Default setting is 0.
	SilentTime	BYTE	[in]	1 byte, TAG LED and buzzer switch OFF time (SilentT*50ms). The range of SilentT is 0 ~ 255. Default setting is 0.
	Times	BYTE	[in]	TAG LED and buzzer action time, The range of Times is 0 ~ 255. Default setting is 0.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.8 SetAntennaMultiplexing

Definition	int SetAntennaMultiplexing(BYTE *ComAdr, BYTE Ant, int FrmHandle);			
Description	Modify antenna configuration of 4 ports reader.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Ant	BYTE	[in]	1 byte, antenna configuration information. 1 antenna is the MINIMUM requirement for operation. **bit0**: antenna 1 configuration bit. 0 – disable antenna 1; 1 – enable antenna 1. **bit1**: antenna 2 configuration bit. 0 – disable antenna 2; 1 – enable antenna 2. **bit2**: antenna 3 configuration bit. 0 – disable antenna 3; 1 – enable antenna 3. **bit3**: antenna 4 configuration bit. 0 – disable antenna 4; 1 – enable antenna 4.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.9 SetBeepNotification

Definition	int SetBeepNotification(BYTE *ComAdr,BYTE BeepEn, int FrmHandle);			
Description	Enable/disable buzzer.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	BeepEn	BYTE	[in]	1 byte. 0x00 – disable buzzer; 0x01 – enable buzzer.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.10 SetRelay

Definition	int SetRelay(BYTE *ComAdr, BYTE RelayTime, int FrmHandle);			
Description	Control the state of the built-in relay.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	RelayTime	BYTE	[in]	Configuration of the relay pickup time, the valid range of the parameter is 1 ~ 255, corresponds to (1 ~ 255)*50ms
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.11 GetGPIOStatus

Definition	int GetGPIOStatus(BYTE *ComAdr, BYTE *OutputPin, int FrmHandle);			
Description	Obtain the input and output states of GPIO.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	OutputPin	BYTE*	[out]	1 byte, bit0 ~ bit1 represent IN1 to IN2 and bit4 ~ bit5 represent Out1 to Out2 respectively.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.12 SetGPIO

Definition	int SetGPIO(BYTE *ComAdr, BYTE OutputPin, int FrmHandle);			
Description	Control the output of GPIO. The default output is high TTL level.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	OutputPin	BYTE	[in]	1 byte, the output state of GPIO (pin Out1 to Out2). Bit0 ~ bit1 control Out1 to Out1 respectively. Bit2 ~ bit7 are reserved.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.13 SetNotificationPulseOutput

Definition	int SetNotificationPulseOutput(BYTE *ComAdr, BYTE OutputRep, int FrmHandle);			
Description	Enable / disable a 2ms low level notification pulse on the associated port for every tag detection.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	OutputRep	BYTE	[in]	1 byte. **bit0**: associated with antenna 1 and Out1 pin. 1 – Out1 will output a 2ms low level notification pulse for every tag detection from antenna 1; 0 – disable notification signal. **bit1**: associated with antenna 2 and Out2 pin. 1 – Out2 will output a 2ms low level notification pulse for every tag detection from antenna 2; 0 – disable notification signal. **bit2**: associated with antenna 3 and Out3 pin. 1 – Out3 will output a 2ms low level notification pulse for every tag detection from antenna 3; 0 – disable notification signal. **bit3**: associated with antenna 4 and Out4 pin. 1 – Out4 will output a 2ms low level notification pulse for every tag detection from antenna 4; 0 – disable notification signal. All other values are reserved, default is 0.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.14 GetSeriaNo

Definition	int GetSeriaNo(BYTE * ComAddr, BYTE* SeriaNo, int FrmHandle);			
Description	Obtain 4 bytes reader unique serial number.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	SeriaNo	BYTE*	[out]	4 bytes, the unique serial number of reader.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.15 SetCheckAnt

Definition	int SetCheckAnt(BYTE *address, BYTE CheckAnt, int FrmHandle);			
Description	Enable antenna check during tag reading/writing operations.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	CheckAnt	BYTE	[in]	1 byte, antenna check switch. 0x00 – disable antenna check; 0x01 – enable antenna check.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.16 ReadActiveModeData

Definition	int ReadActiveModeData(BYTE * LoadData, int *Datalength, int FrmHandle);			
Description	Obtain reader uploaded data in real time mode.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	LoadData	BYTE*	[out]	Reader uploaded raw data, with a length stated in Datalength. Refer to user manual for detail format.
	Datalength	BYTE*	[out]	Byte length of LoadData.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.17 WriteRfPower

Definition	int WriteRfPower(BYTE * ComAddr, BYTE PowerDbm, int FrmHandle);			
Description	Modify RF power configuration separately for write operations			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	PowerDbm	BYTE	[in]	1 byte, separate reader RF power for write operation. The range is 0 ~ 30 and the unit is dBm. bit7 = 0: disable; bit7 = 1: enable.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.18 ReadRfPower

Definition	int WriteRfPower(BYTE * ComAddr, BYTE PowerDbm, int FrmHandle);			
Description	Obtain RF power configuration for write operations			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	PowerDbm	BYTE*	[out]	1 byte, separate reader RF power for write operation. The range is 0 ~ 30 and the unit is dBm. bit7 = 0: disabled; bit7 = 1: enabled.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.19 RetryTimes

Definition	int RetryTimes(BYTE * ComAddr, BYTE *Times, int FrmHandle);			
Description	Modify or load the maximum operation retry time configuration			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Times	BYTE*	[out]	1 byte. bit7 = 1: modify maximum write retry time; bit7 = 0: load maximum write retry time The valid range of configuration is 0 ~ 7.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.20 SetReadMode

Definition	int SetReadMode(BYTE *ComAdr,BYTE ReadMode ,int FrmHandle);			
Description	Select reader working mode.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	ReadMode	BYTE	[in]	1 byte. 0x00 – answer mode; 0x01 – real time inventory mode; 0x02 – trigger mode.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.21 SetReadParameter

Definition	int SetReadParameter(BYTE *ComAdr, BYTE *Parameter, BYTE MaskMem, BYTE *MaskAdr, BYTE MaskLen, BYTE *MaskData, BYTE MaskFlag, BYTE AdrTID, BYTE LenTID, BYTE TIDFlag, int FrmHandle);			
Description	Modify the real time inventory associated parameters.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Parameter	BYTE*	[in]	Parameters definition 5 bytes. **Parameter[0]**: 1 byte, tag type definition for real time inventory. 0 – EPC C1G2 (ISO18000-6C) tags. 1 – ISO18000-6B tags. All other values are reserved, reader will return a parameter error status in the response frame if other value is delivered in this field. **Parameter[1]**: 1 byte, time break between 2 real time inventories. 0x00 – 10ms; 0x01 – 20ms; 0x02 – 30ms; 0x03 – 50ms; 0x04 – 100ms. **Parameter[2]**: 1 byte, tag filtering time. The valid value of this parameter is 0 ~ 255, corresponds to (0 ~ 255)*1s. In real time inventory, if reader detects a particular tag for more than 1 time, reader will only upload tag information of this tag once within the pre-defined filtering time. For FliterTime = 0, disable tag filtering function. **Parameter[3]**: 1 byte, the original Q-value of the EPC tag inventory, the setting of Q-value should follows the rule: $2^Q \approx$ total amount of tags within the effective field. The range of Q-value is 0 ~ 15, if other value is delivered in this field, reader will return a parameter error status in the response frame. **Parameter[4]**: 1 byte, the Session-value of the EPC tag inventory. 0x00 – apply S0 as Session value;

				0x01 – apply S1 as Session value; 0x02 – apply S2 as Session value; 0x03 – apply S3 as Session value; 0xff – apply reader smart configuration (only valid in EPC inventory).
	MaskMem	BYTE	[in]	1 byte, mask area indication. 0x01 – EPC memory; 0x02 – TID memory; 0x03 – User memory.
	MaskAdr	BYTE*	[in]	2 bytes, entry bit address of the mask. The valid range of MaskAdr is 0 ~ 16383.
	MaskLen	BYTE	[in]	1 byte, bit length of mask (unit: bits).
	MaskData	BYTE*	[in]	N bytes, mask data. N = MaskLen/8. If MaskLen is not a multiple of 8 integer, N= int[MaskLen/8]+1. Non-specified lower significant figures should be filled up with 0.
	AdrTID	BYTE	[in]	1 byte, entry address of TID memory inventory.
	LenTID	BYTE	[in]	1 byte, data length for TID inventory operation, the valid range of LenTID is 0 ~ 15.
	TIDFlag	BYTE	[in]	1 byte, inventory purpose indicator. 0 – EPC inventory; 1 – TID inventory.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.7.22 GetReadParameter

Definition	int GetReadParameter (BYTE *ComAdr,BYTE* Parameter,int FrmHandle);			
Description	Obtain the real time inventory associated parameters.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	Parameter	BYTE*	[out]	Parameters definition **ReadMode**: 1 byte, the current working mode of reader. **TagProtocol**: 1 byte, tag type definition of real time inventory. **ReadPauseTime**: 1 byte, time break between 2 real time inventories. **FliterTime**: 1 byte, tag filtering time of real time inventory. **QValue**: 1 byte, the original Q-value of real time mode for EPC inventory. **Session**: 1 byte, the Session-value of real time mode for EPC inventory. **MaskMem, MaskAdr, MaskLen, MaskData**: mask condition for **EPC C1G2** tag inventory in real time mode. MaskMem and MaskLen are 1 byte long respectively. MaskAdr is 2 bytes long, most-significant byte first. MaskData has constant length of 32 bytes, fill the over MaskLen value content with 0. **AdrTID**: entry address of TID memory inventory. **LenTID**: data length for TID inventory operation.
	FrmHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3.23 SetDRM

Definition	int SetDRM(BYTE* ComAddr, BYTE DRM, int PortHandle);			
Description	Modify DRM status.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	DRM	BYTE	[in]	1 byte, DRM status. 0x00 – disable; 0x01 – enable.
	PortHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3.24 GetDRM

Definition	int GetDRM(BYTE*, BYTE* DRM, int PortHandle);			
Description	Obtain DRM status.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE*	[in/out]	Reader address, obtained from open port commands.
	DRM	BYTE*	[out]	1 byte, DRM status. 0x00 – disabled; 0x01 – enabled.
	PortHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3.25 GetReaderTemperature

Definition	int GetReaderTemperature (BYTE* ComAddr, BYTE* PlusMinus, BYTE* Temperature, int PortHandle);			
Description	Obtain current reader temperature.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE *	[in/out]	Reader address, obtained from open port commands.
	PlusMinus	BYTE *	[out]	1 byte, temperature range. 0 – below 0 °c; 1 – above 0 °c.
	Temperature	BYTE *	[out]	1 byte, temperature value, the unit is °c.
	PortHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

3.3.26 MeasureReturnLoss

Definition	int MeasureReturnLoss(BYTE* ComAddr, BYTE* TestFreq, BYTE Ant, BYTE * ReturnLoss, int PortHandle);			
Description	Measure antenna return loss.			
Parameters	**Name**	**Type**	**Orientation**	**Notes**
	ComAddr	BYTE *	[in/out]	Reader address, obtained from open port commands.
	TestFreq	BYTE *	[in]	4 bytes, target frequency point.
	Ant	BYTE	[in]	1 byte, target antenna for this measurement 0 – antenna 1; 1 – antenna 2; 2 – antenna 3; 3 – antenna 4.
	ReturnLoss	BYTE *	[out]	1 byte, result of the return loss measurement, the unit of ReturnLoss is dB.
	PortHandle	int	[in]	Reader connected handle, obtained from open port commands.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).			
Sample code	N/A			

Appendix 1: Reader Error code table

Error code	Description
0x00	API is called successfully.
0x01	Tag inventory is completed successfully and reader is able to deliver data response within the predefined inventory time.
0x02	Inventory timeout.
0x05	Access password error.
0x09	Kill password error.
0x0A	All-zero tag killing password is invalid.
0x0B	Command is not support by the tag
0x0C	All-zero tag access password is invalid for such command.
0x0D	Fail to setup read protection for a protection enabled tag.
0x0E	Fail to unlock a protection disabled tag.
0x10	Some bytes stored in the tag are locked.
0x11	Lock operation failed.
0x12	Already locked, lock operation failed.
0x13	Fail to store the value of some preserved parameters. Configuration will still valid before reader shut down.
0x14	Modification failed.
0x15	Response within the predefined inventory time.
0x17	Further data is waiting to be delivered.
0x18	Reader memory is full.
0x19	All-zero access password is invalid for such command or command is not supported by the tag.
0xF8	Error detected in antenna check.
0xF9	Operation failed.
0xFA	Tag is detected, but fails to complete operation due to poor communication.
0xFB	No tag is detected.
0xFC	Error code returned from tags.
0xFD	Command length error.
0xFE	Illegal command.
0xFF	Parameter error.
0x30	Communication error.
0x33	Reader is busy, operation in process.
0x35	Port is already opened.
0x37	Invalid handle.

Appendix 2: Tag ErrorCode table

ErrorCode	Description
0x00	Other errors, all other Errorcode non-specified error.
0x03	Memory overload, location is not found or unsupported PC value.
0x04	Memory is being temporarily / permanently locked, unable to perform write operation.
0x0B	Unable to perform write operation due to insufficient power supply to tag.
0x0F	Undefined or tag unsupported errors.