AsReader P31N

SDK Reference Guide V1.2

Asterisk, Inc.

Revision History

Version	Description	Date
V1.0	Initial version	2019/04/18
V1.1	1， updates appendix2 2， Adds function: SetStaticIP	2019/6/17
V1.2	Adds function GetSdkVersion	2019/6/21

Contents

System Requirements

Windows version:

Development software: Visual Studio 2012+

Development language: C#

1 SDK Usage

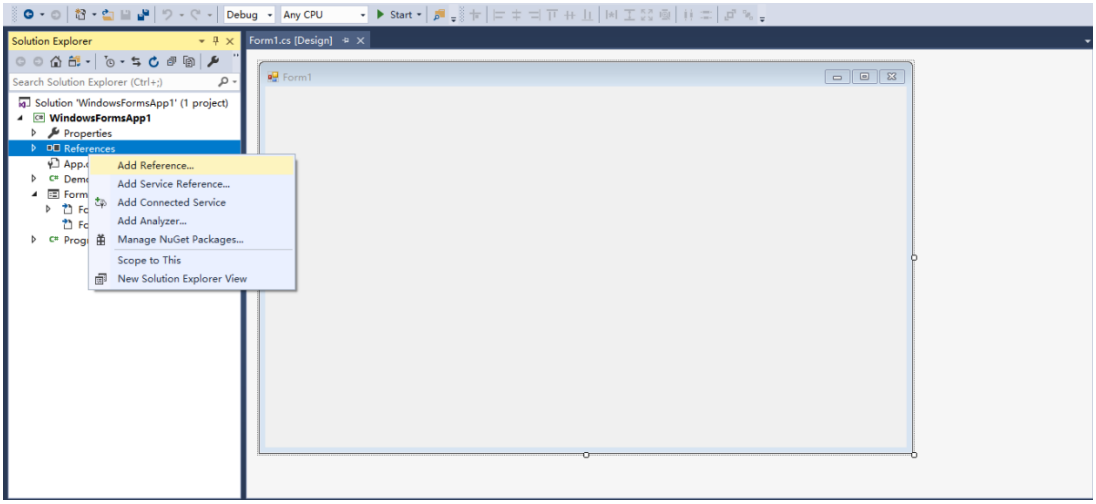
1.1 Add SDK

1. New project —— a Windows forms application.

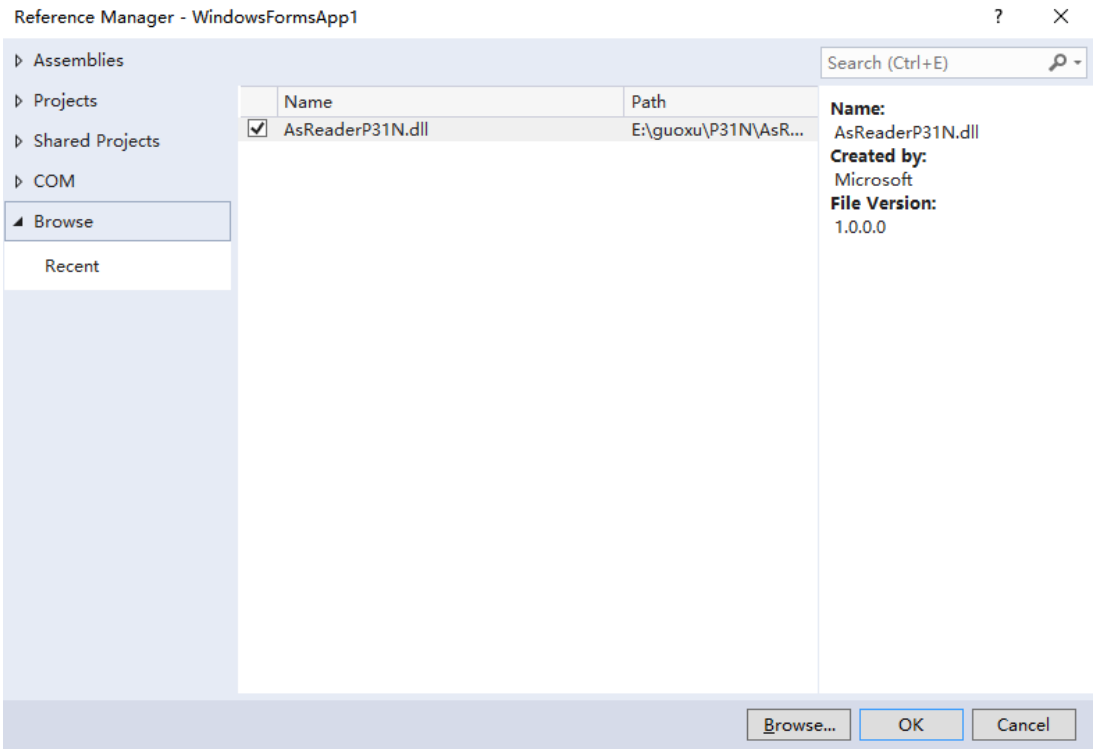
Copy AsReaderP31N.dll into the project folder.

2. Add a reference to the project ——AsReaderP31N.dll.

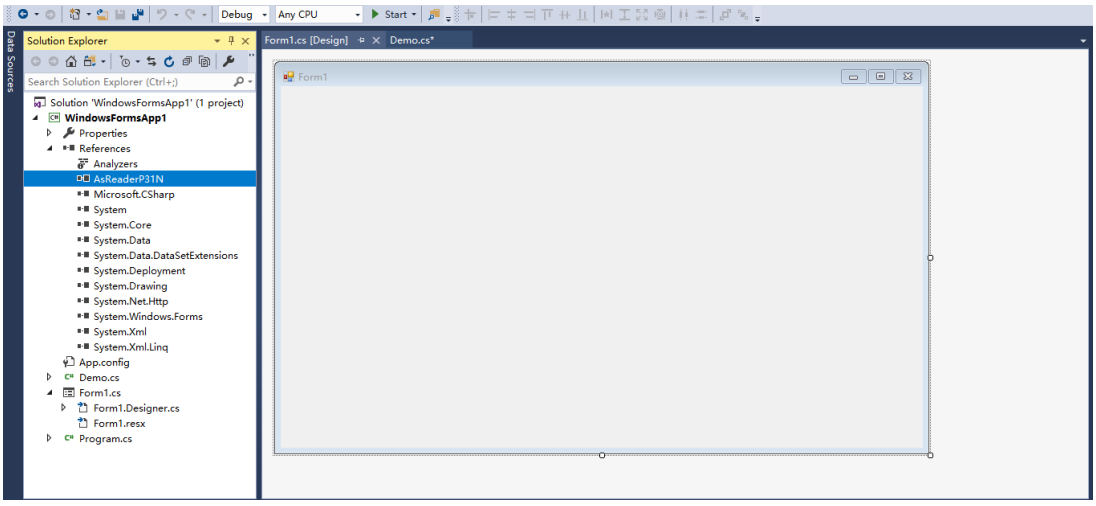
Right-click "reference" to add a reference.



3. Click “Browse”, check “AsreaderP31N.dll”, and click “OK”.



4. The page looks like the following figure after completing the addition.

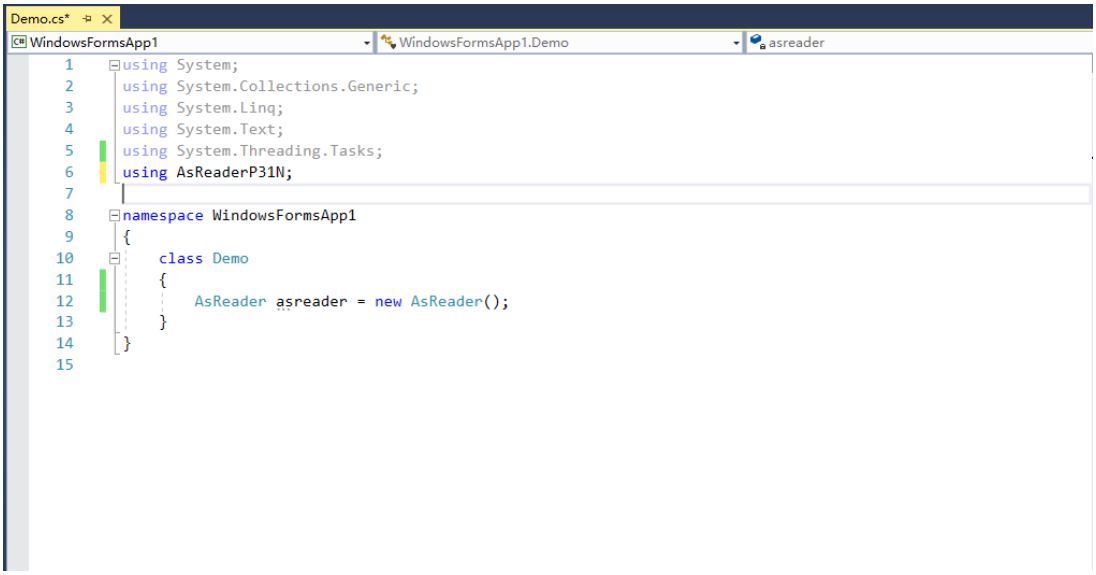


1.2 SDK Import

1. Reference namespace

```
using AsReaderP31N;
```

2. Instantiate objects.



3. Run the ConnectWithVCP function (see section 2.1.1) to Connect to the AsReader.

Note: If successful, return 0; If it fails, return 1.

```
asreader.ConnectWithVCP("COM1");
```

2 API Description

2.1 AsReader Class

The AsReader Class provides API interfaces such as RFID inventory, read tags, write tags, lock tags, and so on.

2.1.1 ConnectWithVCP

Function	UInt32 ConnectWithVCP(string comPort)		
Parameter	**IN/OUT**	**Type**	**Description**
comPort	IN	string	serial number
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Open the COM to established the connection between the host and device, then initializing all the device and registering all the callback handler.			
For example: ConnectWithVCP(“COM1”);			

2.1.2 ConnectWithTCP

Function	UInt32 ConnectWithTCP(string ipAddress,string tcpPort)		
Parameter	**IN/OUT**	**Type**	**Description**
ipAddress	IN	string	IP Address of AsReader
tcpPort	IN	string	Port number of AsReader
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Open the TCP/IP to established the connection between the host and device, then initializing all the device and registering all the callback handler.			
For example: ConnectWithTCP(“192.168.1.100”,“9600”); **Note:**			

TCP port number: 5000. UDP port number: 50001.

2.1.3 Disconnect

Function	UInt32 Disconnect()		
Parameter	**IN/OUT**	**Type**	**Description**
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Disconnect the terminal from the AsReader device and reset the AsReader device.			
For example: Disconnect the terminal from the AsReader device. Disconnect();			

2.1.4 StartInventory

Function	UInt32 StartInventory(bool an1,bool an2,bool an3,bool an4, bool an5,bool an6,bool an7,bool an8)		
Parameter	**IN/OUT**	**Type**	**Description**
an1	IN	bool	True: open antenna port; False: close the antenna port.
an2	IN	bool	True: open antenna port; False: close the antenna port.
an3	IN	bool	True: open antenna port; False: close the antenna port.
an4	IN	bool	True: open antenna port; False: close the antenna port.
an5	IN	bool	True: open antenna port; False: close the antenna port.
an6	IN	bool	True: open antenna port; False: close the antenna port.
an7	IN	bool	True: open antenna port; False: close the antenna port.
an8	IN	bool	True: open antenna port; False: close the antenna port.
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: The device opens the specified antenna ports for tag inventory.			
For example: Open antenna1 to inventory tags. StartInventory(true,false,false,false,false,false,false);			

2.1.5 StartInventory

Function	UInt32 StartInventory(InventoryType inventoryType)		
Parameter	**IN/OUT**	**Type**	**Description**

inventoryType	IN	enum	Open the corresponding tag inventory mode. Refer to 2.2.1.1.
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Start to inventory tags.			
For example: Only inventory the EPC bank data of tags. StartInventory(ONLY_PC_EPC);			

2.1.6 StopInventory

Function	UInt32 StopInventory()		
Parameter	**IN/OUT**	**Type**	**Description**
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Stop inventorying tags.			
For example: Stop inventorying tags. StopInventory();			

2.1.7 SetSelectMask

Function	UInt32 SetSelectMask(MemBankType memBank,TargetType target,ActionType action unit startAddressWord,byte[] selectMask)		
Parameter	**IN/OUT**	**Type**	**Description**
memBank	IN	enum	The memory bank of the tag to be selection-mask compared. Refer to 2.2.1.6 .
target	IN	enum	The session for the Selection Mask to be applied. Refer to 2.2.1.4.
action	IN	enum	Action after tags be flagged. Refer to 2.2.1.5.
startAddressWord	IN	uint	Offset of the selected tag bank, which is the starting address. (unit: word)
selectMask	IN	byte	Value of Selection Mask. [0-16]. (unit: word)
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Set the select parameter, so that in the case of counting multiple tags, the select parameter can be used for inventorying/reading/writing/locking and other operations on specific tags.			

For example: Select the label that meets the following criteria: Bank: EPC; Session: SESSION_S0; Action: ACTION_ASLINVA_DSLINVB; Offset: 2; Mask length: 6; Mask contents: byte[]selectMask={0x12,0x34,0x56,0x78,0x12,0x34,0x56x,0x78,0x12,0x34,0x56,0x78}; SetSelectMask(Mem_EPC,SESSION_S0,ACTION_ASLINVA_DSLINVB,0x02,selectMask);

2.1.8 WriteMemory

Function	UInt32 WriteMemory(MemBankType memBank,uint startAddressWord, uint accessPassword, byte[] writeData,byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
memBank	IN	enum	Sets the tag memory bank to be written to. Refer to 2.2.1.6.
startAddressWord	IN	uint	Sets start address of data. (unit: word)
accessPassword	IN	uint	The access password for the target tag. (0 if no password)
writeData	IN	byte	The value to be written.
epcData	IN	byte	EPC number of the target tag.
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Target a tag with epcData and writes data to the target bank of the tag. The length of data to be written should not exceed 32 Words(64 Bytes).			
For example: Write tags. Target bank: EPC; Offset: 2; Access password: 0x12345678; Value: byte[] writedata = {0x12,0x34}; EPC number of target tag: byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34, 0x56x,0x78,0x12,0x34,0x56,0x78}; WriteMemory(Mem_EPC,0x02,0x12345678,writeData,epcData);			

2.1.9 ReadMemory

Function	UInt32 ReadMemory(MemBankType memBank,uint startAddressWord,

	uint lengthWord, uint accessPassword,byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
memBank	IN	enum	The tag memory bank to be read. Refer to 2.2.1.6.
startAddressWord	IN	uint	Offset of target tag bank. (unit: word)
lengthWord	IN	uint	Length of the data to be read. (unit: word)
accessPassword	IN	uint	The access password for the target tag. (0 if no password)
epcData	IN	byte	EPC number of the target tag.
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Target a tag with epcData and reads data of the target bank. The length of data to be read should not exceed 32 Words(64 Bytes).			
For example: Read tags. Target bank: EPC; Offset: 2; Length to read: 2; Access password: 0x12345678; EPC number of target tag: byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34, 0x56x,0x78,0x12,0x34,0x56,0x78}; ReadMemory(Mem_EPC,0x02,0x02,0x12345678,epcData);			

2.1.10 Kill

Function	UInt32 Kill(uint KillPassword,byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
KillPassword	IN	uint	Kill password of the target tag.
epcData	IN	byte	EPC number of the target tag.
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Before killing the tag, the kill password needs to be written to the RESERVED bank, starting with offset 00 and writing two words. **Note:** Kill operation is irreversible once executed.			
For example: Kill password is 0x12345678. EPC number of target tag: byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34, 0x56x,0x78,0x12,0x34,0x56,0x78}; Kill(0x12345678,epcData);			

2.1.11 LockMemory

Function	Uint32 LockMemory(TagMask tagMask,TagAction tagAction,unit accessPassword,byte[] epcData)		
Parameter	**IN/OUT**	**Type**	**Description**
tagMask	IN	TagMask	Mask for the Lock operation. Refer to Appendix III.
tagAction	IN	TagAction	Action for the Lock operation. Refer to Appendix III.
accessPassword	IN	uint	The access password for the target tag. (0 if no password)
epcData	IN	byte	EPC number of the target tag.
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.

Description:

Used to lock (Lock) /permanently lock (PermaLock)/ unlock (Unlock)or permanently unlock (PermaUnlock)the target bank of the tag.

Before locking the tag, the access password needs to be written to the RESERVED bank, starting with offset 0X20 and writing two words.

The high 4 bits of the Lock operation parameter target are reserved bits. The remaining 20 bits are the Payload of the Lock operation, including the Mask and the Action, 10 bits each from high to low. Only Actions with Mask bit equals 1 are valid.

Each data area has 2bits for Action.00 01 10 11 indicate unlock, permanently unlock, lock, and permanently lock.

The meanings of each bit of Mask and Action are shown in the following table:

Lock-Command Payload

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Kill Mask		Access Mask		EPC Mask		TID Mask		File_0 Mask		Kill Action		Access Action		EPC Action		TID Action		File_0 Action	

Masks and Associated Action Fields

	Kill pwd		Access pwd		EPC memory		TID memory		File_0 memory	
	19	18	17	16	15	14	13	12	11	10
Mask	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write
	9	8	7	6	5	4	3	2	1	0
Action	pwd read/ write	perma lock	pwd read/ write	perma lock	pwd write	perma lock	pwd write	perma lock	pwd write	perma lock

For example: Lock the EPC bank of the tag.

Access password of the target tag is 12345678.

EPC number of target tag: byte[] epcData= {0x12,0x34,0x56,0x78,0x12,0x34, 0x56x,0x78,0x12,0x34,0x56,0x78};

LockMemory(0x008020,0x12345678,epcData);

2.1.12 SetRegion

Function	UInt32 SetRegion(RegionType region)		
Parameter	**IN/OUT**	**Type**	**Description**
region	IN	enum	Country or region currently located. Refer to 2.2.1.2
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: To set the country or region currently located.			
For example: Set the country or region currently located to be REGION_CHINA1. SetRegion(REGION_CHINA1);			

2.1.13 GetRegion

Function	UInt32 GetRegion(ref Region region)		
Parameter	**IN/OUT**	**Type**	**Description**
region	OUT	enum	Country or region currently located. Refer to 2.2.1.2
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: To get the country or region currently located.			
For example: To get the country or region currently located. GetRegion(Value);			

2.1.14 GetTxPower

Function	UInt32 GetTxPower(ref uint power)		
Parameter	**IN/OUT**	**Type**	**Description**
power	OUT	uint	Value of TX power. [13-24]
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Used to get the TX power for the AsReader device.			
For example: To get the TX power for the AsReader device. GetTxPower(power);			

2.1.15 SetTxPower

Function	UInt32 SetTxPower(uint txPower)		
Parameter	**IN/OUT**	**Type**	**Description**
txPower	IN	uint	Value of TX power. [13-24]
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Used to set the TX power for the AsReader device.			
For example: Set TX power to 24dBm. SetTxPower(24);			

2.1.16 SetTxPower

Function	UInt32 SetTxPower(uint txPower_an1, uint txPower_an2, uint txPower_an3, uint txPower_an4, uint txPower_an5, uint txPower_an6, uint txPower_an7, uint txPower_an8)		
Parameter	**IN/OUT**	**Type**	**Description**
txPower_an1	IN	uint	TX power of antenna 1. [13-24]
txPower_an2	IN	uint	TX power of antenna 2. [13-24]
txPower_an3	IN	uint	TX power of antenna 3. [13-24]
txPower_an4	IN	uint	TX power of antenna 4. [13-24]
txPower_an5	IN	uint	TX power of antenna 5. [13-24]
txPower_an6	IN	uint	TX power of antenna 6. [13-24]
txPower_an7	IN	uint	TX power of antenna 7. [13-24]
txPower_an8	IN	uint	TX power of antenna 8. [13-24]
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: To set the current TX power of the antennas.			
For example: Set the TX power of antenna1 to 24dBm. SetTxPower(24,0,0,0,0,0,0,0);			

2.1.17 GetTxPower

Function	UInt32 GetTxPower(ref uint power_an1, ref uint power_an2, ref uint power_an3, ref uint power_an4, ref uint power_an5, ref uint power_an6, ref uint power_an7, ref uint power_an8)		
Parameter	**IN/OUT**	**Type**	**Description**

power_an1	OUT	uint	TX power of antenna 1. [13-24]
power_an2	OUT	uint	TX power of antenna 2. [13-24]
power_an3	OUT	uint	TX power of antenna 3. [13-24]
power_an4	OUT	uint	TX power of antenna 4. [13-24]
power_an5	OUT	uint	TX power of antenna 5. [13-24]
power_an6	OUT	uint	TX power of antenna 6. [13-24]
power_an7	OUT	uint	TX power of antenna 7. [13-24]
power_an8	OUT	uint	TX power of antenna 8. [13-24]
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: To get the current TX power of the antennas.			
For example: Get the current TX power of the antennas. GetTxPower(power_an1,power_an2,power_an3,power_an4,power_an5,power_an6, power_an7,power_an8);			

2.1.18 SetSession

Function	UInt32 SetSession(SessionType session)		
Parameter	**IN/OUT**	**Type**	**Description**
session	IN	enum	RFID Session. Refer to 2.2.1.3
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Used to set RFID Session.			
For example: Set RFID Session to S0. SetSession(SESSION_S0);			

2.1.19 GetSession

Function	UInt32 GetSession(ref SessionType session)		
Parameter	**IN/OUT**	**Type**	**Description**
session	OUT	enum	RFID Session. Refer to 2.2.1.3
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Used to get the current RFID Session.			
For example: Get the current RFID Session. GetSession(session);			

2.1.20 SetChannel

Function	UInt32 SetChannel(ChannelType channel)		
Parameter	**IN/OUT**	**Type**	**Description**
channel	IN	enum	RFID channel. Refer to 2.2.1.7 The range of channel number depends on regional settings.
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Used to set the RFID channel of the device.			
For example: Set the RFID channel to CHANNEL_24. SetChannel(CHANNEL_24);			

2.1.21 GetChannel

Function	UInt32 GetChannel(ref ChannelType channel)		
Parameter	**IN/OUT**	**Type**	**Description**
channel	OUT	enum	RFID channel. Refer to 2.2.1.7 The range of channel number depends on regional settings.
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Used to get the RFID channel of the device.			
For example: Get the RFID channel of the device. GetChannel(channel);			

2.1.22 SetGain

Function	UInt32 SetGain(GainType gain)		
Parameter	**IN/OUT**	**Type**	**Description**
gain	IN	enum	PA Gain mode. Refer to 2.2.1.9
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Set the PA Gain mode.			
For example: Set the PA Gain mode to HIGH_GAIN. SetGain(HIGH_GAIN);			

2.1.23 GetGain

Function	UInt32 GetGain(ref GainType gain)		
Parameter	**IN/OUT**	**Type**	**Description**
gain	OUT	enum	PA Gain mode. Refer to 2.2.1.9
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Get the PA Gain mode.			
For example: Get the PA Gain mode. GetGain(gain);			

2.1.24 SetQuery

Function	UInt32 SetQuery(QueryType dr, QueryType m,QueryType tRext, QueryType sel, QueryType session, QueryType target, QType q)		
Parameter	**IN/OUT**	**Type**	**Description**
dr	IN	enum	DR (TRcal divide ratio) sets the T=>R link frequency. Refer to 2.2.1.10
m	IN	enum	Encoding mode. Refer to 2.2.1.11
tRext	IN	enum	TRext chooses whether a Tag prepends the T=>R preamble with a pilot tone. Refer to 2.2.1.12
sel	IN	enum	Sel chooses which Tags respond to the Query. Refer to 2.2.1.13
session	IN	enum	Session chooses a session for the inventory rounds. Refer to 2.2.1.3
target	IN	enum	Target selects whether Tags whose inventoried flag is A or B participate in the inventory rounds. Tags may change their inventoried flag from A to B (or vice versa) as a result of being singulated. Refer to 2.2.1.9
q	IN	enum	Q sets the number of slots in the round. Slot counts=2^q. Refer to 2.2.1.14
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Sets all parameter values in the tag query operation.			
For example: Set the dived ratio= DR_8, encoding mode= M1, no pilot tone. Sel= SEL_ALL, Session= SESSION_S0, Session flag= TARGET_A, Slots=16. SetQuery(DR_8,M1,NO_Pilot_Tone,SEL_ALL,SESSION_S0,TARGET_A,Q4);			

2.1.25 GetQuery

Function	UInt32 GetQuery(ref QueryType dr, ref QueryType m, ref QueryType tRext, ref QueryType sel, ref QueryType session, ref QueryType target, ref QType q)		
Parameter	**IN/OUT**	**Type**	**Description**
dr	IN	enum	DR (TRcal divide ratio) sets the T=>R link frequency. Refer to 2.2.1.10
m	IN	enum	Encoding mode. Refer to 2.2.1.11
tRext	IN	enum	TRext chooses whether a Tag prepends the T=>R preamble with a pilot tone. Refer to 2.2.1.12
sel	IN	enum	Sel chooses which Tags respond to the Query. Refer to 2.2.1.13
session	IN	enum	Session chooses a session for the inventory rounds. Refer to 2.2.1.3
target	IN	enum	Target selects whether Tags whose inventoried flag is A or B participate in the inventory rounds. Tags may change their inventoried flag from A to B (or vice versa) as a result of being singulated. Refer to 2.2.1.9
q	IN	enum	Q sets the number of slots in the round. Slot counts=2^q. Refer to 2.2.1.14
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Gets all parameter values in the tag query operation.			
For example: Gets all parameter values in the tag query operation. GetQuery(dr,m,tRext,sel,session,target,q);			

2.1.26 SetAntiCollisionMode

Function	UInt32 SetAntiCollisionMode(AntiCollisionMode antiCollisionMode)		
Parameter	**IN/OUT**	**Type**	**Description**
antiCollisionMode	IN	enum	Anti-Collision Mode Refer to 2.2.1.15
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Sets which anti-collision algorithm to use.			

For example: Sets to use FixedQ. SetAntiCollisionMode(FixedQ);

2.1.27 GetAntiCollisionMode

Function	UInt32 GetAntiCollisionMode(ref AntiCollisionMode antiCollisionMode)		
Parameter	**IN/OUT**	**Type**	**Description**
antiCollisionMode	OUT	enum	Anti-Collision Mode Refer to 2.2.1.15
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Gets which anti-collision algorithm is using.			
For example: Gets which anti-collision algorithm is using. GetAntiCollisionMode(antiCollisionMode);			

2.1.28 SetCWOn

Function	UInt32 SetCWOn()		
Parameter	**IN/OUT**	**Type**	**Description**
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Open Carrier Wave (CW). Carrier Wave(CW): Carrier wave describes a reader transmitted signal at the configured channel or frequency without any forward link modulation.			
For example: Open Carrier Wave (CW). SetCWOn();			

2.1.29 SetCWOff

Function	UInt32 SetCWOff()		
Parameter	**IN/OUT**	**Type**	**Description**
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Close Carrier Wave (CW).			

Carrier Wave(CW): Carrier wave describes a reader transmitted signal at the configured channel or frequency without any forward link modulation.
For example: Close Carrier Wave (CW). SetCWOff();

2.1.30 SetReadTime

Function	UInt32 SetReadTime(uint time_an1,uint time_an2,uint time_an3,uint time_an4,uint time_an5,uint time_an6,uint time_an7,uint time_an8,)		
Parameter	**IN/OUT**	**Type**	**Description**
time_an1	IN	uint	Read time of antenna1 (10~40000, 1=1ms)
time_an2	IN	uint	Read time of antenna2 (10~40000, 1=1ms)
time_an3	IN	uint	Read time of antenna3 (10~40000, 1=1ms)
time_an4	IN	uint	Read time of antenna4 (10~40000, 1=1ms)
time_an5	IN	uint	Read time of antenna5 (10~40000, 1=1ms)
time_an6	IN	uint	Read time of antenna6 (10~40000, 1=1ms)
time_an7	IN	uint	Read time of antenna7 (10~40000, 1=1ms)
time_an8	IN	uint	Read time of antenna8 (10~40000, 1=1ms)
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.
Description: Sets the read time of each antenna.			
For example: Set the read time of each antenna to 1000ms. SetReadTime(1000,1000,1000,1000,1000,1000,1000,1000);			

2.1.31 GetReadTime

Function	UInt32 GetReadTime(ref uint time_an1,ref uint time_an2,ref uint time_an3,ref uint time_an4,ref uint time_an5,ref uint time_an6,ref uint time_an7,ref uint time_an8)		
Parameter	**IN/OUT**	**Type**	**Description**
time_an1	IN	uint	Read time of antenna1 (10~40000, 1=1ms)
time_an2	IN	uint	Read time of antenna2 (10~40000, 1=1ms)
time_an3	IN	uint	Read time of antenna3 (10~40000, 1=1ms)
time_an4	IN	uint	Read time of antenna4 (10~40000, 1=1ms)
time_an5	IN	uint	Read time of antenna5 (10~40000, 1=1ms)
time_an6	IN	uint	Read time of antenna6 (10~40000, 1=1ms)
time_an7	IN	uint	Read time of antenna7 (10~40000, 1=1ms)
time_an8	IN	uint	Read time of antenna8 (10~40000, 1=1ms)
Return Value	OUT	UInt32	If successful, return 0; If it fails, return 1.

Description: Gets the read time of each antenna.
For example: Get the read time of each antenna. GetReadTime(time_an1,time_an2,time_an3,time_an4,time_an5,time_an6,time_an7,time_an8);

2.1.32 GetDeviceInformation

Function	void GetDeviceInformation(ref DeviceInformation deviceInformation)		
Parameter	**IN/OUT**	**Type**	**Description**
deviceInformation	OUT	DeviceInformation	Device information. Refer to Appendix II
Return Value	OUT	void	
Description: Gets the device information.			
For example: Gets the device information. DeviceInformation mode; GetDeviceInformation(mode);			

2.1.33 SetStaticIP

Function	bool SetStaticIP(string ip,string subnet,string gateway,string dnsServer)		
Parameter	**IN/OUT**	**Type**	**Description**
ip	IN	string	IP address
subnet	IN	string	Subnet mask
gateway	IN	string	Gateway
dnsServer	IN	string	DNS server
Return Value	OUT	bool	
Description: Sets a static IP address.			
For example: SetStaticIP(“192.168.3.1”,”255.255.255.0”,”192.168.3.1”,”192.168.3.1”);			

2.1.34 GetSdkVersion

Function	void GetSdkVersion(ref string sdkVersion)		
Parameter	**IN/OUT**	**Type**	**Description**
sdkVersion	OUT	string	SDK version
Return Value	OUT	Void	
Description:			

This function is used to get the SDK version.
For example: string sdkVersion; GetSdkVersion(sdkVersion);

2.1.35 SetDelegate

Function	void SetDelegate(CallBackReadTagDatareadTagData,CallBackErrorCode errorCode,CallBackSuccessCode successCode)		
Parameter	**IN/OUT**	**Type**	**Description**
readTagData	IN	CallBackReadTagData	The delegate function for data processing.
errorCode	IN	CallBackErrorCode	The delegate function used to execute the error output.
successCode	IN	CallBackSuccessCode	The delegate function used to execute the success output.
Return Value	OUT	void	
Description: Used to set the delegate functions.			
For example: CallBackReadTagData Rec = null; CallBackErrorCode Rec1 = null; CallBackSuccessCode Rec2 = null; Void test(InventoryResult ReadTagStruct); Void test1(uint error); Void test2(uint success); Rec = test; Rec1 = test1; Rec2 = test2 SetDelegate(Rec,Rec1,Rec2);			
Note: Defines the delegate function for data processing and the delegate function for error output. public delegate void CallBackReadTagData(InventoryResult tagcallbackdata); public delegate void CallBackErrorCode (uint error); public delegate void CallBackErrorCode (uint success); InventoryResult: refer to Appendix I. error: refer to 2.2.1.16. success: refer to 2.2.1.17.			

2.2 Types Class

The Types class is used to define the region type/RFID mode type/session type/search mode type/Selection Mask target session type/action type/memory bank type and key value type.

2.2.1 Enumeration Types

2.2.1.1 InventoryType

Query Parameter	Value
PC_EPC_RSSI	1
PC_EPC_TID	2
ONLY_PC_EPC	3

2.2.1.2 RegionType

Region	Value
REGION_US	0x21
REGION_US_Narrow	0x22
REGION_Europe	0x31
REGION_JAPAN	0x41
REGION_CHINA1	0x51
REGION_CHINA2	0x52
REGION_BRAZIL	0x61

2.2.1.3 SessionType

Session	Value
SESSION_S0	0x0
SESSION_S1	0x1
SESSION_S2	0x2
SESSION_S3	0x3

2.2.1.4 TargetType

The target session for the Selection Mask is applied.

Target	Value
SESSION_S0	0x0
SESSION_S1	0x1
SESSION_S2	0x2
SESSION_S3	0x3
SL_FLAG	0x4

2.2.1.5 ActionType

ActionType enumeration type that returns what happens to the Session and Session Flag of a tag that matches and does not match with a Selection Mask when using the Selection Mask function.

Action	Value
ACTION_ASLINVA_DSLINVB	0x0
ACTION_ASLINVA_NOTHING	0x1
ACTION_NOTHING_DSLINVB	0x2
ACTION_NSLINVS_NOTHING	0x3
ACTION_DSLINVB_ASLINVA	0x4
ACTION_DSLINVB_NOTHING	0x5
ACTION_NOTHING_ASLINVA	0x6
ACTION_NOTHING_NSLINVS	0x7

2.2.1.6 MemBankType

MemBankType enumeration type that returns memory bank of tag which mask data of Selection Mask will be compared to.

MemBank	Value
MEM_RESERVED	0x0
MEM_EPC	0x1
MEM_TID	0x2
MEM_USER	0x3

2.2.1.7 ChannelType

RF Channel	Value
CHANNEL_24	24

CHANNEL_25	25
CHANNEL_26	26
CHANNEL_27	27
CHANNEL_28	28
CHANNEL_29	29
CHANNEL_30	30
CHANNEL_31	31
CHANNEL_32	32

2.2.1.8 GainType

PA Gain	Value
HIGH_GAIN	0x00
LOW_GAIN	0x01

2.2.1.9 TargetABType

Session Flag	Value
TARGET_A	0x00
TARGET_B	0x01

2.2.1.10 DRType

TRcal Divide Ratio	Value
DR_8	0x00
DR_64_3	0x01

2.2.1.11 MType

M (cycles per symbol) sets the T=>R data rate and modulation format.

M	Value
M1	0x00
M2	0x01
M4	0x02
M8	0x03

2.2.1.12 TRextType

TRext chooses whether a Tag prepends the T=>R preamble with a pilot tone.

TRext	Value
NO_Pilot_Tone	0x00
Use_Pilot_Tone	0x01

2.2.1.13 SelType

Sel chooses which Tags respond to the Query.

Sel	Value
SEL_ALL	0x00
SEL_SL_N	0x02
SEL_SL	0x03

2.2.1.14 QType

Q sets the number of slots in the round.

Q	Value
Q0	0
Q1	1
Q2	2
Q3	3
Q4	4
Q5	5
Q6	6
Q7	7
Q8	8
Q9	9
Q10	10
Q11	11
Q12	12
Q13	13
Q14	14
Q15	15

2.2.1.15 AntiCollisionModeType

Anti-Collision Mode	Value
FixedQ	0x0
DynamicQ	0x01

2.2.1.16 ErrorCode

Error Code	Value
OTHER_ERROR	0x0
NOT_SUPPORTED	0x1
INSUFFICIENT_PRIVILEGES	0x2
MEMORY_OVERRUN	0x3
MEMORY_LOCKED	0x4
CRYPTO_SUITE_ERROR	0x5
COMMAND_NOT_ENCAPSULATED	0x6
RESPONSEBUFFER_OVERFLOW	0x7
SECURITY_TIMEOUT	0x8
INSUFFICIENT_POWER	0xB
NON_SPECIFIC_ERROR	0xF
SENSOR_SCHEDULING_CONFIGURATION	0x11
TAG_BUSY	0x12
MEASUREMENT_TYPE_NOT_SUPPORTED	0x13
NO_TAG_DETECTED	0x80
HANDLE_ACQUISITION_FAILURE	0x81
ACCESS_PASSWORD_FAILURE	0x82
KILL_PASSWORD_FAILURE	0x83
CRC_ERROR	0x90
RX_TIMEOUT	0x91
REGISTRY_UPDATE_FAILURE	0xA0
REGISTRY_ERASE_FAILURE	0xA1
REGISTRY_WRITE_FAILURE	0xA2
REGISTRY_NOT_EXIST	0xA3
UART_FAILURE	0xB0
SPI_FAILURE	0xB1
I2C_FAILURE	0xB2
GPIO_FAILURE	0xB3
NOT_SUPPORTED_COMMAND	0xE0
UNDEFINED_COMMAND	0xE1
INVALID_PARAMETER	0xE2
TOO_HIGH_PARAMETER	0xE3
TOO_LOW_PARAMETER	0xE4
FAILURE_AUTOMATIC_READ_OPERATION	0xE5
NOT_AUTOMATIC_READ_MODE	0xE6
FAILURE_TO_GET_LAST_RESPONSE	0xE7
FAILURE_TO_CONTROL_TEST	0xE8
FAILURE_TO_RESET_READER	0xE9

RFID_BLOCK_CONTROL_FAILURE	0xEA
PR9200_BUSY	0xEB
COMMAND_FAILURE	0xF0
VERIFY_FAILURE	0xF1
ABNORMAL	0xFC
ERROR_NONE	0xFF

2.2.1.17 SuccessCode

Success Code	**Value**
SET_READER_POWER_CONTROL	0x0
GET_READER_INFORMATION	0x03
GET_REGION	0x06
SET_REGION	0x07
SET_SYSTEM_RESET	0x08
GET_TYPE_C_AI_SELECT_PARAMETERS	0xB
SET_TYPE_C_AI_SELECT_PARAMETERS	0xC
GET_TYPE_C_AI_QUERY_RELATED_PARAMETERS	0xD
SET_TYPE_C_AI_QUERY_RELATED_PARAMETERS	0xE
GET_CURRENT_RF_CHANNEL	0x11
SET_CURRENT_RF_CHANNEL	0x12
GET_FH_AND_LBT_PARAMETERS	0x13
SET_FH_AND_LBT_PARAMETERS	0x14
GET_TX_POWER_LEVEL	0x15
SET_TX_POWER_LEVEL	0x16
RF_CW_SIGNAL_CONTROL	0x17
GET_MULTIPLE_POWER	0x18
SET_MULTIPLE_POWER	0x19
SET_ANTENNA	0x1B
GET_READ_TIME	0x1E
SET_READ_TIME	0x1F
READ_TYPE_C_UII	0x22
READ_TYPE_C_UII_RSSI	0x23
READ_TYPE_C_USER_DATA	0x24
READ_TYPE_C_UII_TID	0x25
START_AUTO_READ	0x27
STOP_AUTO_READ	0x28
READ_TYPE_C_TAG_DATA	0x29

READ_TYPE_C_TAG_DATA2	0x2A
GET_SESSION	0x2E
SET_SESSION	0x2F
GET_FREQUENCY_HOPPING_TABLE	0x30
SET_FREQUENCY_HOPPING_TABLE	0x31
GET_MODULATION	0x32
SET_MODULATION	0x33
GET_ANTICOLLISION_MODE	0x34
SET_ANTICOLLISION_MODE	0x35
START_AUTO_READ2	0x36
STOP_AUTO_READ2	0x37
START_AUTO_READ_RSSI	0x38
STOP_AUTO_READ_RSSI	0x39
START_AUTO_READ_EX2	0x3A
WRITE_TYPE_C_TAG_DATA	0x46
BLOCKWRITE_TYPE_C_TAG_DATA	0x47
BLOCKERASE_TYPE_C_TAG_DATA	0x48
NXP_READPROTECT	0x52
NXP_RESET_READPROTECT	0x53
NXP_CHANGE_EAS	0x54
NXP_EAS_ALARM	0x55
NXP_CALIBRATE	0x56
ISP_DATA	0x57
KILL_RECOM_TYPE_C_TAG	0x65
SET_GAIN	0x66
Get_GAIN	0x67
LOCK_TYPE_C_TAG	0x82
BLOCKPERMALOCK_TYPE_C_TAG	0x83
SET_MODEM_REGISTER	0xA6
SET_RF_REGISTER	0xA7
GET_MODEM_REGISTER	0xA8
GET_RF_REGISTER	0xA9
ISP_DOWNLOAD	0xB1
ISP_DUMP	0xB3
SET_ANTENNA_PATH	0xB4
SET_LEAKAGE_CAL_MODE	0xB5
GET_IAP_VERSION	0xB6
GET_TEMPERATURE	0xB7
GET_RSSI	0xC5
SCAN_RSSI	0xC6
UPDATE_REGISTRY	0xD2

ERASE_REGISTRY	0xD3
GET_REGISTRY_ITEM	0xD4
SET_REGISTRY_ITEM	0xD5
SET_OPTIMUM_FREQUENCY_HOPPING_TABLE	0xE4
GET_FREQUENCY_HOPPING_MODE	0xE5
SET_FREQUENCY_HOPPING_MODE	0xE6
GET_TX_LEAKAGE_RSSI_LEVEL_FOR_SMART_HOPPING_MODE	0xE7
SET_TX_LEAKAGE_RSSI_LEVEL_FOR_SMART_HOPPING_MODE	0xE8
START_READ_WITH_FAST_LEAKAGE_CAL	0xEC
REQUEST_FAST_LEAKAGE_CAL	0xED

Appendix I

InventoryResult

Items displayed when inventory tags
rssi
channel
phase
antenna
TagData tagData

TagData

Tag Data
pc
epc
tid
data

Appendix II

Device Information
macAddress
version
status
ipAddress
subnetMask
gateway
dns
dhcp
password

Appendix III

TagMask
userMemoryBit1
userMemoryBit2
tidMemoryBit1
tidMemoryBit2
epcMemoryBit1
epcMemoryBit2
accessMemoryBit1
accessMemoryBit2
killMemoryBit1
killMemoryBit2

TagAction
userMemoryBit1
userMemoryBit2
tidMemoryBit1
tidMemoryBit2
epcMemoryBit1
epcMemoryBit2
accessMemoryBit1
accessMemoryBit2
killMemoryBit1
killMemoryBit2