航天华拓
Aerospace Innotech

Programming Manual

SAAT-800 Series Reader

Version: V2.0.0

Dear Customers:

Thanks for your trust and support! We will be happy to provide you with comprehensive service and technical support.

This manual will instruct you how to call API function package in CD-ROM when you using the SAAT-800 series reader and enable your software integration communicating with the reader, configuring reader parameters, controlling the reader to operate on tag and acquire tag returned operation information.

If you experience problems during operation, please contact our technical support department.

Your comments & suggestions are warmly welcome during your operation on our products. We will be always at your side.

This manual is suitable for the following readers:

SAAT-800 Series Reader

When this manual is written it is supposed that readers have basic knowledge over RFID and computer. The technical words, such as RFID, Radio Frequency & Ethernet etc. in this manual, are not specifically introduced. Please refer to other reference books or consult our technical department for support.

The following marks would appear in our manual, the meaning are as below:

Warning

If it disobey the restricted operations or using environments, it might be harm to human health or damage equipments.

Advice

Follow the advised instructions, the results might be better.

Table of Contents

Part I
Overview

1. API Function Package Introduction

The RFID devices provided by SZAAT would accompany with API function package. When user is developing RFID integrated application system, through calling this API function package, to control SAAT series reader and read/write operation on RFID tags.

To call API function package, user doesn't have to know the underlying communication format between computer and reader, neither have to know the interface protocol or the process of reading/writing operation. The user only needs to focus on the functions, to simplify the difficulty of using RFID reader during the process of system integration.

2. API Function Description

- Through API interface function, the following operations are available:
- Establish connection with reader based on serial, USB or Ethernet port.
- Configure and query reader system, RF and communication parameters.
- Achieve the basic operations on tag, such as read, write, lock, etc.
- Achieve the detection and control operations of reader IO input/output status.

Part II
Call of API

1. Environment Requirements

The following configurations are needed when user calls for API package:

Computer configuration: PIII 600MHz, 128M Memory, Windows XP/2003 Operation System, etc.

2. File Included

API function package including the following files:

- RFIDAPI.dll
- RFIDAPI.lib
- RFIDAPIEXPORT.h

3. API Call Methods

Add the files which Section 2.2 mentions to the project, and add the following sentence to source code: #include "RFIDAPIEXPORT.h", and then call the functions in it directly.

Please refer to the specific functions and features described in Part 4.

Part III
Special Explanation

1. About Tag

1.1 EPC Tag

EPC Class1 Gen2 is UHF frequency band RFID tag standard defined by EPC Global, which is currently known as EPC tag. The EPC tag is defined as ISO18000-6C standard after it is adapted by International Standardization Organization, the content of the two standards are basically the same. In this file and API function package, EPC tag is unified as ISO18000-6C or 6C tag, and will no longer be highlighted.

1.2 ISO8000-6B Tag

Currently, the ISO18000-6B tags are mostly from NXP, and general configuration is 224-byte data bank.

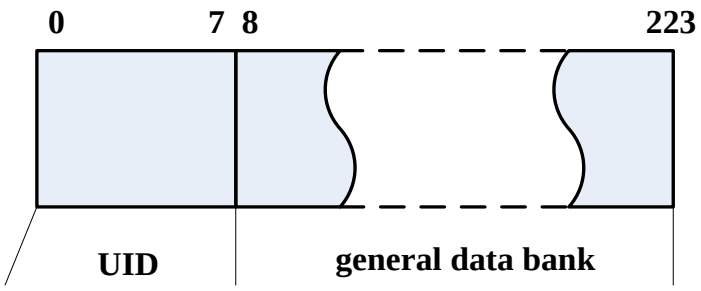

Figure.3-1 ISO18000-6B tag data bank configuration

The first 8bytes (address 0~7) is UID code, which is unique and locked in factory, only executing read operation.

8～223 bytes is general data bank, which can execute the operations such as write, read, lock and lock status querying.

When operating on ISO18000-6B general data bank, user must be aware that data address should not exceed the maximum bytes, which means parameter "start address" + "data length" not greater than 223.

Before operate on tag data bank, user should better confirm with tag provider for specific parameters.

1.3 ISO18000-6C Tag

There are several ISO18000-6C chip suppliers, and their data bank configurations are different due to applications and cost.

Popular configuration of data bank:

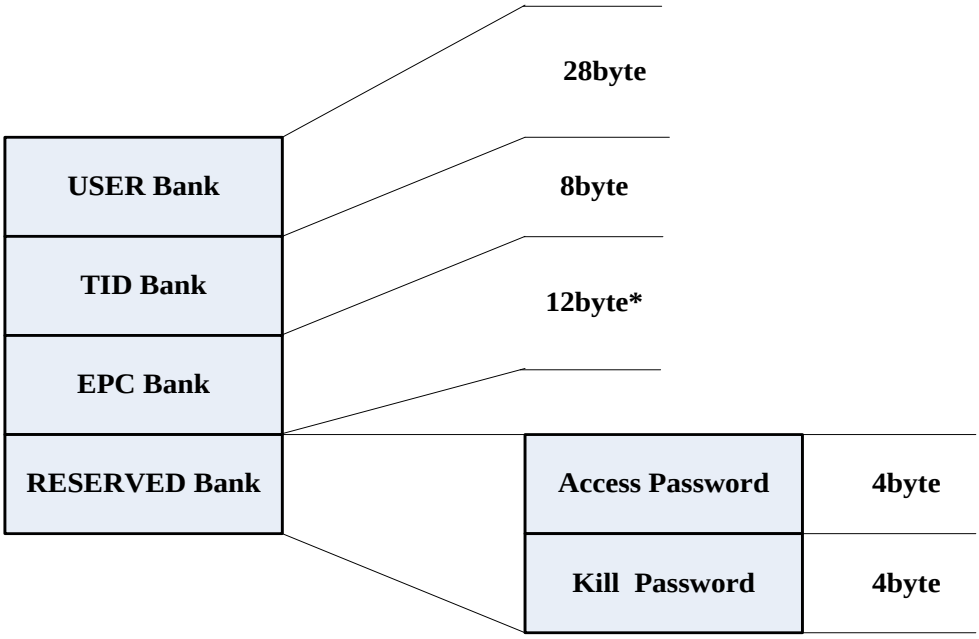

Figure.3-2 ISO18000-6B tag data bank configuration

TID Bank of ISO18000-6C tag is similar with UID Code of ISO18000-6B tag, and is required to be unique. While ISO18000-6C tag from some of these factories has no TID Bank, or with TID Bank but without unique TID code. There's only unified code from factory, some tags even has 4bytes TID Bank only.

EPC Code of ISO18000-6C tag is e-code, which is 0~64bytes according to protocol definition. Currently, tags from most factory is 12bytes (the actual length is greater than 12byte, but user usually doesn't involve in the rest content)

The length of USER Bank (user data area) is different due to different factory or model, some even without USER bank. Most tag factories use 28byte data bank.

RESERVED Bank including two areas: access password area and deactivating password area, 4byte each.

When operating on ISO18000-6C data banks, user must be aware that data address should not exceed the maximum bytes, which means parameter "start address" + "data length" not greater than 223.

Before operate on tag data bank, user should better confirm with tag provider for specific parameters.

2. About Reader

2.1 Antenna Port

The fixed reader usually has several antenna ports (F805 with 4 antenna ports). Based on this, user can equip with multi-antenna to achieve one reader operates on multi-spots, or operates on the same spot from different angle to ensure tag identifying accuracy. The function of antenna port is the same, working with time rolling method. In tag operation process, user has to point out which one to use. When the "port number" value is "0x01～0x04", the corresponding reader antenna port is 1#~4#. If it's "0x00", this requires all the usable ports of reader to query in roll.

2.2 Multi Reader

- User application software can simultaneously control multiple readers simultaneously, for the reader's recognition through the following ways:
- Serial or USB connection is achieved through the serial port number. When initial the serial or USB port, initialization function would return an initialization handle, and every operation function provides a handle as a mark for identifying different readers.
- If using multi-bus mode like RS-485, the initialization handle is the same, but the data bus address frame difference.
- If using Ethernet connection, you have to initial an Ethernet with software for different IP address and returning to each handle, and then call a different handle to a different IP.

Part IV
API Function List

- API function package including five functions: Communication Control Function, System Parameter Configuration Function, Reader Operation Function, Tag Operation Function and API Configuration Function.
- Communication control function includes establishing intercommunication connection between computer and reader, and the relative functions in communication process control.
- System parameter configuration function is used to query the reader configuration parameters, including communication parameters, the RF parameters.
- Reader operation function is to control the reader implementing the manipulation applications as switching on/off carrier, IO operation.
- Tag operation function is the specific operation function as reading, writing, locking a tag. The following document will describe ISO18000-6B and ISO18000-6C separately.
- API configuration function is used to configuring relative information for querying API function package.

1. Communication Control Function

No.	Name of Function	Description
1	SAAT_TCPInit	This function is to initial configuration parameter of computer Ethernet.
2	SAAT_COMInit	This function is to initial computer serial port and parameter.
3	SAAT_USBInit	This function is to initial computer USB port and parameter.
4	SAAT_Open	Create connection based on one of the serial port, USB port or Ethernet.
5	SAAT_Close	Close the established connection.
7	SAAT_Reconnect	Reconnect reader
8	SAAT_HeartSend	Check the connection status of communication links

2. System Parameter Configuration Function

No.	Name of Function	Description
1	SAAT_SysInfSet	To configure system parameter of reader.
2	SAAT_SysInfQuery	To query system parameter of reader.
3	SAAT_WorkModeSet	To set work mode of reader.
4	SAAT_ParmOp	To operate the input/output of reader parameter list.
5	SAAT_RFParaSet	To set reader RF parameter.
6	SAAT_RFParaQuery	To query reader RF parameter.
7	SAAT_CommunicatParaSet	To set reader communication parameter.
8	SAAT_CommunicatParaQuery	To query reader communication parameter.
9	SAAT_NetParaSet	To set reader network parameter.
10	SAAT_NetParaQuery	To query reader network parameter.
11	SAAT_TagOpParaSet	To set tag operation parameter
12	SAAT_TagOpParaQuery	To query tag operation parameter
13	SAAT_ExtendBroadParaSet	To set reader expansion board parameter.
14	SAAT_ExtendBroadParaQuery	To query reader expansion board parameter.
15	SAAT_TotalAntennaParmQuery	To query all RF ports parameters
16	SAAT_AntennaParmQuery	To query Specified RF port Parameter
17	SAAT_AntennaParmSet	To set RF port parameters

18	SAAT_SetAntennaPortEnable	To set port enable
19	SAAT_SetAntennaPower	To set port power
20	SAAT_SetAntennaTime	To set port read time

3. Reader Operation Function

No.	Function Name	Description
1	SAAT_PowerOff	To stop reading
2	SAAT_CarrierWaveOp	To control the carrier operation of reader.
3	SAAT_IOOperate	To control the IO output operation of reader.
4	SAAT_IOStateQuery	To query the IO input status of reader.
5	SAAT_Reboot	To reboot reader
6	SAAT_Reading_IOConfig	To config buzzer status
7	SAAT_Reading_IOQuery	To query buzzer status
8	SAAT_IOPulseWidthSet	To config the pulse width of output IO
9	SAAT_IOPulseWidthQuery	To query the pulse width of output IO

4. Tag Operation Function

No.	Function Name	Description
1	SAAT_6BTagSelect	Command the reader to select 6B tag.
2	SAAT_6BReadUIDCode	Command the reader to read UID code on 6B tag.
3	SAAT_6BReadUserData	Command the reader to reader data on 6B tag data bank.
4	SAAT_6BRevUIDMsg	To receive UID code of 6B tag, returning from reader.
5	SAAT_6BWriteUserData	Command the reader to write data on 6B tag data bank.
6	SAAT_6BTagLock	Command the reader to lock 6B tag data bank.
7	SAAT_6CTagSelect	Command the reader to select 6C tag.
8	SAAT_6CReadEPCCode	Command the reader to read EPC code on 6C tag.
9	SAAT_6CRevEPCMsg	To receive the EPC code returning from reader.
10	SAAT_6CReadTIDCode	Command the reader to read TID code of 6C tag
11	SAAT_6CRevTIDMsg	To receive TID code of 6C tag returning from reader.
12	SAAT_6CWriteEPCCode	Command the reader to rewrite EPC code of 6c tag.
13	SAAT_6CReadUserData	Command the reader to read user data bank of 6C tag.
14	SAAT_6CWriteUserData	Command the reader to write data to 6C tag user data bank.
15	SAAT_6CWriteBankData	Command the reader to operate 6C tag Bank.
16	SAAT_6CClearBankData	Command the reader to block-erase 6C tag Bank.
17	SAAT_6CAccessPWDSet	Command the reader to rewrite access password of 6C tag.
18	SAAT_6CDestroyPWDSet	Command the reader to rewrite destruction password of 6C tag.
19	SAAT_6CTagLock	Command the reader to lock 6C tag.
20	SAAT_6CTagKill	Command the reader to deactivate 6C tag.
21	SAAT_6CEASFlagSet	Command the reader to set EAS bits of 6C tag.
22	SAAT_6CEASMonitorEnable	Command the reader to monitor EAS.

5. API Configuration Function

No.	Function Name	Description
1	SAAT_Copyright	Information querying function of API software.
2	SAAT_SetLanguageType	Set function of API language type.
3	SAAT_GetErrorMessage	Get err information
4	SAAT_GetErrorCode	Get err code

Part V
API Function Description

1. Communication Control Function

1.1 TCP Parameter Initialization

Description	By an argument, this function is used to configure the computer's Ethernet applications, initialization of a SOCKET, to prepare a connection based on the Ethernet interface.	
Prototype	bool SAAT_TCPInit (void** pHandle,char *pHostName, int nsocketPort)	
Parameter	pHandle	the preservation of opening ports handle
	pHostName	reader IP address
	nsocketPort	reader SOCKET port, default is 7086
Return	true	Operation Success;
	false	Operation Failed.

1.2 COM Port Parameter Initialization

Description	By an argument, this function is to initial the specified computer COM port, preparing for the connection based on serial port.	
Prototype	Bool SAAT_COMInit(void** pHandle, unsigned char nBusAddr, char *pComNum, int nBaud)	
Parameter	pHandle	the preservation of opening ports handle
	nBusAddr	the Bus address, default is 0
	pComNum	COM1－COM4 communication mode is only valid in the COM
	nBaud	serial communication speed, default is 19200
Return	true	Operation Success;
	false	Operation Failed.

1.3 USB Parameter Initialization

Description	Reader USB interface, the current application is a virtual serial port, so the use of USB interface also need to initialize the COM port. Through the argument this function initializes the computer COM port, to prepare USB-based connection.	
Prototype	Bool SAAT_USBInit(void** pHandle,unsigned char nBusAddr,char * pUSBNum,int nBaud)	
Parameter	pHandle	the preservation of opening ports handle
	nBusAddr	the Bus address, default is 0
	pUSBNum	COM1 － COM4 (COM1-COM4 is virtual serial port, USB switching communication protocol) is only valid to USB communication mode.
	nBaud	USB port communication speed, default is 152000
Return	true	Operation Success;
	false	Operation Failed.

1.4 UDP Parameter Initialization

Description	By an argument, this function is used to configure the computer's Ethernet applications, initialization of a SOCKET, to prepare a connection based on the Ethernet interface.	
Prototype	bool SAAT_UDPInit (void** pHandle,char *pHostName, int nsocketPort)	
Parameter	pHandle	the preservation of opening ports handle
	pHostName	reader IP address
	nsocketPort	reader udp port, default is 7088
Return	true	Operation Success;
	false	Operation Failed.

1.5 Establish Connection

Description	Create connection based on one of the following interfaces, as serial port, USB port, Ethernet port or any other ports.	
Prototype	bool SAAT_Open(void* pHandle)	
Parameter	pHandle	the preservation of opening ports handle
Return	true	Operation Success;
	false	Operation Failed.

1.6 Close the Connection

Description	Close the connection has been established.	
Prototype	bool SAAT_Close(void *pHandle)	
Parameter	pHandle	the preservation of opening ports handle
Return	true	Operation Success;
	false	Operation Failed.

1.7 Re-connect reader

Description	Reconnect reader	
Prototype	bool SAAT_Reconnect(void *pHandle)	
Parameter	pHandle	the preservation of opening ports handle
Return	true	Operation Success;
	false	Operation Failed.

1.8 Check the connection

Description	Check the connection status of communication links	
Prototype	bool SAAT_HeartSend (void *pHandle)	
Parameter	pHandle	the preservation of opening ports handle
Return	true	Operation Success;
	False	Operation Failed.

2. Configuration of System Information

2.1 System Information Setting

Description: The function is used to set reader system configuration parameters, such as the name of readers, the passwords.

Prototype: bool SAAT_SysInfSet (void * pHandle, unsigned char nType, unsigned char* pParm, unsigned char nLen)

Parameter:

pHandle: the preservation of opening ports handle

configuration type of system information

nType	Definition	Length
0	Name of Reader	8bytes
10	Access Password of Reader System	6bytes

pParm: system parameter

nLen: length of system parameter

Return:

true: Operation Success;

false: Operation Failed.

2.2 Query of System Information

Description: This function is used to query the reader system information and other parameters, see the following query type as the "information type" defined.

Prototype: bool SAAT_SysInfQuery (void* pHandle ,unsigned char nType, unsigned char *pPara, unsigned char *pLen)

Parameter:

pHandle: the preservation of opening ports handle

nType: the information type for query(as following)

pParm: point to the pointer of memory which receives reader parameter data

nLen: pLen the length of pointed memory

Return:

true: Operation Success;

false: Operation Failed.

Explanation:

nType(information type) definition description:

nType	**Definition**	**Length**
0x00	Name of Reader	8bytes
0x01	Model of Reader	6bytes
0x02	Factory Serial No. of Reader	8bytes
0x03	Software Version of Reader Processor	4bytes
0x04	Software Version of Reader Decoding Unit	4bytes
0x05	Hardware Version No. of Baseband Circuit	4bytes
0x06	Hardware Version No. Of RF Circuit	4bytes

2.3 Configuration of Working Mode

Description: This function is used to configure the working mode of the reader.

Prototype: bool SAAT_WorkModeSet (void* pHandle ,unsigned nType)

Parameter:

pHandle: the opening port handle

nType: the working mode to configure

Return	true	Operation Success;
	false	Operation Failed.

Explanation:

nType(information type) definition description:

Working Mode	Definition	Description
0x00	Master-slave mode	Under the command of host computer, reader executes reading tag operation;
0x01	Preset Mode	The Reader executes tag operation based on preset command;
0x02	Trigger Mode	The reader triggers tag operation command according to IO input;
0x03	Timing Mode	According to the system clock timing, reader executes tag operation.

2.4 Operation on System Parameter List

Description	This function is used to operate parameter list of reader (for querying or configuring large quantity of reader parameter), including input/output whole or some part of the system parameter list.	
Prototype	bool SAAT_ParmOp (void* pHandle ,unsigned char nType, unsigned char nStartAddrr, unsigned char nLen, unsigned char *pData, unsigned char *pDataLen)	
Parameter	pHandle	the opening port handle
	nType	working mode of operating parameter list(as explanation)
	nStartAddrr	to specify the start-address of operating reader parameter list
	nLen	to specify the length of reader system parameter list
	pData	incoming data, or outgoing data
	pDataLen	length of pData
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

nType parameter list working mode:

0x00	restore the system parameter list to factory settings
0x01	afferent system parameter list
0x02	efferent system parameter list

Note: when working mode is 0x00, both start-address and operation lengths are meaningless.

2.5 Configuration of Carrier

Description	This function is used for configuring parameter of reader RF carrier.	
Prototype	bool SAAT_RFParaSet (void* pHandle ,unsigned char nType, int nParaLen,unsigned char* pPara)	
Parameter	pHandle	the opening port handle
	nType	parameter type of the set parameter list(as explanation)
	nParaLen	length of pPara
	pPara	the set frequency table(as explanation)
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

"Parameter Type" defined as following, is a carrier parameter type ready to configure:

0x00 Frequency Hopping Table

"Frequency table" is the values of frequency hopping table reader using, a combination of frequency points 0~15(CE and CN standard); a combination of frequency points 0~49(FCC standard).

2.6 Query Carrier Parameter

Description	This function is used for querying parameter settings of reader carrier.	
Prototype	Bool SAAT_RFParaQuery (void* pHandle , unsigned char nType, unsigned char* pPara, unsigned char *pLen)	
Parameter	pHandle	the opening port handle

	nType	query the type of carrier parameter, definition as 5.2.5
	pPara	returning parameter of frequency table
	pLen	incoming: length of pPara; Outgoing: length of frequency table
Return	true	Operation Success;
	false	Operation Failed.

2.7 Configuration of Communication Parameter

Description	This function is used to Bus communication parameters such as serial ports.	
Prototype	bool SAAT_CommunicatParaSet (void* pHandle ,unsigned char nType, unsigned char* pPara, unsigned char nLen)	
Parameter	pHandle	the opening port handle
	nType	configuration parameter type(as explanation)
	pPara	reader serial Bus parameter
	nLen	length of pPara
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

nType configuration parameter type:

0x00	serial Bus device address
0x01	RS-232 Bus speed
0x02	RS-485 Bus speed

Definition of serial Bus device address: when a reader is using multi-node Bus (such as RS-485 Bus), the reader node number is device address. The length of address is 1bytes, value 1~254, 0x00 and 0xff is broadcast address.

RS-232 Bus speed: communication speed of reader RS-232 serial Bus, and data defined as following

0x00	4800bps
0x01	9600 bps
0x02	19200 bps
0x03	38400 bps
0x04	57600 bps
0x05	115200 bps

RS-485 Bus speed: communication speed of RS-485 serial Bus, definition the same as RS-232 serial Bus

2.8 Query Communication Parameter

Description	This function is used for querying communication parameter of reader serial Bus.	
Prototype	bool SAAT_CommunicatParaQuery (void* pHandle ,int nType, unsigned char* pPara,unsigned char *pLen)	
Parameter	pHandle	the opening port handle
	nType	configuration parameter type(as explanation)
	pPara	query reader bus parameter
	pLen	incoming: length of pPara; outgoing: length of reader bus parameter
Return	true	Operation Success;
	false	Operation Failed.

Explanation: definition of nType is the same as 5.2.7.

2.9 Configuration of Network Parameter

Description	This function is used to configure Ethernet parameter, such as reader IP address.	
Prototype	bool SAAT_NetParaSet (void* pHandle ,unsigned char nType, unsigned char* pPara, unsigned char nLen)	
Parameter	pHandle	the opening port handle
	nType	configuration parameter type(as explanation)
	pPara	configuration Ethernet parameter data(as explanation)
	nLen	length of pPara

Return	true	Operation Success;
	false	Operation Failed.

Explanation:
nType is the parameter type:

0x01	MAC address
0x02	IP address
0x03	TCP SOCKET number
0x04	UDP SOCKET number

***IP configuration** Ethernet IP parameters, totally 12bytes in turn including 4bytes IP address, 4bytes mask, 4bytes default gateway. It's not compatible with IPV6 format currently.
***SOCKET number** when host computer and reader communicate via Ethernet, using the socket. This function is used for the reader to identify application program of host computer, the default socket number is 7086、7088. When the socket port is blocked by firewall or occupied by other applications in an Ethernet network system, user is available or legal to use this function to modify port number. Recommended to use with caution!

2.10 Query Network Parameter

Description	This function is used to query Ethernet parameter, such as reader IP address.	
Prototype	bool SAAT_NetParaQuery(void* pHandle ,int nType, unsigned char* pPara,unsigned char *pLen)	
Parameter	pHandle	the opening port handle
	nType	query parameter type(as explanation)
	pPara	the returning network parameter data
	pLen	incoming: length of pPara; outgoing: actual returning length
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
nType is parameter type, definition as 5.2.9

0x01	MAC address
0x02	IP address
0x03	TCP SOCKET port number
0X04	UDP SOCKET port number

2.11 Configuration of Tag Operation Parameter

Description	This function is used to configure the parameters when a reader is operating to read/write tags.	
Prototype	bool SAAT_TagOpParaSet (void* pHandle ,unsigned char nType, unsigned char *pPara,unsigned char nLen)	
Parameter	pHandle	the opening port handle
	nType	configuration parameter type(as explanation)
	pPara	configuration tag operation parameter
	nLen	incoming: length of pPara; outgoing: parameter length of tag operation
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
nType is configuration parameter type, including:

Type	Explanation	Returning information length
0x10	Q value of the default	1bytes
0x11	select parameter(BANK/address/mask/data)	Nbytes
0x13	default length of EPC code(the following is parameter configuration of IS018000-6B tag)	1bytes
0x21	select parameter(type/address/mask/data)	Nbytes

2.12 Query Tag Operation Parameter

Description	This function is used to query the operation parameters when reader is operating on tags.	
Prototype	bool SAAT_TagOpParaQuery (void* pHandle ,unsigned char nType, unsigned char* pPara, unsigned char *pLen)	
Parameter	pHandle	the opening port handle
	nType	configuration parameter type(as explanation)
	pPara	pointing to the memory pointer which saving configuration parameters
	Len	incoming: length of pPara; outgoing: parameter length of tag operation
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
nType is parameter type, definition as ***part V 2.11***

2.13 Configuration of Expansion Board Parameter

Description	This function is used to configure expansion board parameters.	
Prototype	bool SAAT_ExtendBroadParaSet (void* pHandle ,unsigned char nType, unsigned char pSendChunnel)	
Parameter	pHandle	the opening port handle
	nType	configuration tag operation parameter type
	pSendChunnel	configuration tag operation parameter
Return	true	Operation Success;
	false	Operation Failed.

0	IO output operation
1	IO input operation

2.14 Query Expansion Board Parameter

Description	This function is used to query expansion board parameters.	
Prototype	bool SAAT_ExtendBroadParaQuery (void* pHandle ,unsigned char nType, char* pPara, unsigned char* pLen)	
Parameter	pHandle	the opening port handle
	nType	configuration tag operation parameter type
	pPara	query parameters
	pLen	incoming: length of pPara; outgoing: length of query parameters
Return	true	Operation Success;
	false	Operation Failed.

0	IO output operation
1	IO input operation

2.15 Query RF Port Parameter

Description	This function is used to query RF port parameter of reader.	
Prototype	Bool SAAT_TotalAntennaParmQuery(void* pHandle,unsigned char *szAntennaPara,unsigned char *pLen)	
Parameter	pHandle	the opening port handle
	szAntennaPara	all RF port parameters(as explanation)
	pLen	incoming: length of szAntennaPara; outgoing: all RF port parameters length
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
all RF port parameters including

Port Enable	4bytes	Antenna port enable
Power Output	4bytes	Antenna port output power
Cycling Query	4bytes	Antenna port cycling query

"Port Enable" configuration parameter length is 4bytes, in turn for antenna 1~4 port configuration, 0x00 is "forbid",

0x01 is "allow".

"Power Output" configuration parameter length is 4bytes, in turn for antenna 1~4 port power output parameter (0x00~0x0A), and data defined as following

0x00	20.0(dBm)
0x01	21.0(dBm)
0x02	22.0(dBm)
0x03	23.0(dBm)
0x04	24.0(dBm)
0x05	25.0(dBm)
0x06	26.0(dBm)
0x07	27.0(dBm)
0x08	28.0(dBm)
0x09	29.0(dBm)
0x0A	30.0(dBm)

"Cycling Query" configuration parameter length is 4bytes, in turn for antenna 1~4 port cycling query parameter, which is the working time for the specific port, working time=time parameter*100ms.

2.16 Query Specified RF port Parameter

Description	This function is used to query RF parameter of reader parameter table.	
Prototype	bool SAAT_AntennaParmQuery (void* pHandle, unsigned char nAntenna, unsigned char * pAntennaEnable, unsigned char *pAntennaPower, unsigned char *pAntennaQueryTime)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number to configure
	nAntennaEnable	antenna port enable, 0x00 is "forbid", 0x01 is "allow"
	nAntennaPower	power output, in unit of dBm
	nAntennaQueryTime	in cycling query mode, the working time of this port(ms)
Return	true	Operation Success;
	false	Operation Failed.

2.17 Configuration of RF port parameter

Description	This function is used to configure RF port parameter of reader parameter table.	
Prototype	Prototype: bool SAAT_AntennaParmSet(void* pHandle ,unsigned char *pPara,unsigned char nLen)	
Parameter	pHandle	the opening port handle
	pPara	configure RF port parameter of reader parameter table(as explanation)
	nLen	parameter length, usually is 12bytes
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

all RF port parameters including

Port Enable	4bytes	Antenna port enable
Power Output	4bytes	Antenna port output power
Cycling Query	4bytes	Antenna port cycling query

"Port Enable" configuration parameter length is 4bytes, in turn for antenna 1~4 port configuration, 0x00 is "forbid", 0x01 is "allow".

"Power Output" configuration parameter length is 4bytes, in turn for antenna 1~4 port power output parameter (0x00~0x0A), and data defined as following

0x00	20.0(dBm)

0x01	21.0(dBm)
0x02	22.0(dBm)
0x03	23.0(dBm)
0x04	24.0(dBm)
0x05	25.0(dBm)
0x06	26.0(dBm)
0x07	27.0(dBm)
0x08	28.0(dBm)
0x09	29.0(dBm)
0x0A	30.0(dBm)

"Cycling Query" configuration parameter length is 4bytes, in turn for antenna 1~4 port cycling query parameter, which is the working time for the specific port, working time=time parameter*20ms.

2.18 Specified RF port Enable Configuration

Description	This function is used to configure reader specified RF port enable.	
Prototype	bool SAAT_SetAntennaPortEnable (void* pHandle, unsigned char nAntenna, unsigned char nEnable)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number for configuring
	nEnable	enable configuration
		0x00 forbid 0x01 allow
Return	true	Operation Success;
	false	Operation Failed.

2.19 Specified RF port Power Configuration

Description	This function is used to configure reader specified RF port power.	
Prototype	bool SAAT_SetAntennaPower (void* pHandle,unsigned char nAntenna,unsigned char nPower)	
Parameter	pHandle	the opening port handle
	nAntenna	RF port number 1~4
	nPower	Configuration power(0x00~0x0A)
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
nPower defined as following

0x00	20.0(dBm)
0x01	21.0(dBm)
0x02	22.0(dBm)
0x03	23.0(dBm)
0x04	24.0(dBm)
0x05	25.0(dBm)
0x06	26.0(dBm)
0x07	27.0(dBm)
0x08	28.0(dBm)
0x09	29.0(dBm)
0x0A	30.0(dBm)

2.20 Specified RF port Cycling Query Configuration.

Description	This function is used to configure each RF port polling time, according to the importance of each port and reading requirements to set time, thus to optimize system.	
Prototype	bool SAAT_SetAntennaTime (void* pHandle,unsigned char nAntenna,unsigned char nTime)	
Parameter	pHandle	the opening port handle
	nAntenna	RF port number 1~4
	nTime	set polling time, working time=polling time*100ms
Return	true	Operation Success;
	false	Operation Failed.

3. Reader Operation Functions

3.1 Stop Reading

Description	This function is used to inform the reader stop the current executing instructions related to operation on tags.	
Prototype	bool SAAT_PowerOff(void *pHandle);	
Parameter	pHandle	the opening port handle
Return	true	Operation Success;
	false	Operation Failed.

3.2 Carrier Operation

Description	This function is used to control reader carrier, opening the carrier and output or stop outputting by specified RF port. Knowing from principle of radio frequency identification, when reading tags the reader is required to have continuous carrier output, and providing energy for tag operation. This function is used to control open/stop of carrier. But in actual application, reader would auto-open carrier in executing tag operation and would auto-stop after tag operation finished. Thus users usually don't need to use it.	
Prototype	bool SAAT_CarrierWaveOp(void* pHandle ,unsigned char nType, unsigned char nPort)	
Parameter	pHandle	the opening port handle
	nType	carrier operation mode
		0x00 open 0x01 stop
	nPort	the open carrier port 1~4
Return	true	Operation Success;
	false	Operation Failed.

3.3 IO Output Operation

Description	This function is used to control reader operation on multi IO output port.	
Prototype	bool SAAT_IOOperate(void* pHandle,unsigned char nPort,unsigned char nState)	
Parameter	pHandle	the opening port handle
	nPort	IO output port number 1~4

		0x00	all ports
		0x01	port 1
		0x02	port 2
		0x03	port 3
		0x04	port 4
	nState	IO output power level state(as explanation)	
Return	true	Operation Success;	
	false	Operation Failed.	

Explanation:
nState output state:

0x00	specify IO switch to low power level state
0x01	specify IO switch to high power level state
0x02	specify IO output 10ms positive pulse signal
0x03	specify IO output 10ms negative pulse signal

3.4 IO Input Query Command

Description	This function is used to query reader IO input port status.	
Prototype	bool SAAT_IOStateQuery(void* pHandle,unsigned char *pState)	
Parameter	pHandle	the opening port handle
	pState	power level state pointer of input port
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
port status is indicated with bytes:

0x00	specify IO switch to low power level state
0x01	specify IO switch to high power level state
0x02	specify IO output 10ms positive pulse signal
0x03	specify IO output 10ms negative pulse signal

Bit	Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Definition	Reserve	Reserve	Reserve	Reserve	Reserve	Reserve	Port 2	Port 1

Port Status: 0 low power level; 1 high power level

3.5 Reboot reader

Description	This function is used to reboot reader	
Prototype	bool SAAT_Reboot(void* pHandle,unsigned char nMode)	
Parameter	pHandle	the opening port handle
	nMode	0x00 general 0x01 reboot
Return	true	Operation Success;
	false	Operation Failed.

3.6 Config reading indicate

Description	Config reading indicate	
Prototype	bool SAAT_Reading_IOConfig (void* pHandle,unsigned char nConfigBit)	
Parameter	pHandle	the opening port handle
	nConfigBit	Bit7 Bit6 Bit5 Bit4 Bit3 Bit2 Bit1 Bit0 R R R IO4 IO3 IO2 IO1 buzzer 0 disable 1 enable (R: Reserve)
Return	true	Operation Success;
	false	Operation Failed.

3.7 Query reading indicate

Description	Query reading indicate	
Prototype	bool SAAT_Reading_IOQuery (void* pHandle,unsigned char* pConfigBit)	
Parameter	pHandle	the opening port handle
	nConfigBit	Bit7 Bit6 Bit5 Bit4 Bit3 Bit2 Bit1 Bit0 R R R IO4 IO3 IO2 IO1 buzzer 0 disable 1 enable (R: Reserve)
Return	true	Operation Success;
	false	Operation Failed.

3.8 Config reading indicate pluse width of output IO

Description	Config reading indicate pluse width of output IO	
Prototype	bool SAAT_IOPulseWidthSet (void* pHandle,unsigned char nIOPort, unsigned char nWidth)	
Parameter	pHandle	the opening port handle
	nIOPort	Config IO port(0-4):port 0 is buzzer, port 1\port 2\port3\port4 is IO port
	nWidth	Pulse width (unit100ms)
Return	true	Operation Success;
	false	Operation Failed.

3.9 Query reading indicate pluse width of output IO

Description	Query reading indicate pluse width of output IO	
Prototype	bool SAAT_IOPulseWidthQuery (void* pHandle,unsigned char nIOPort, unsigned char* pWidth)	
Parameter	pHandle	the opening port handle
	nIOPort	Config IO port(0-4):port 0 is buzzer, port 1\port 2\port3\port4 is IO port
	nWidth	Pulse width (unit100ms)
Return	true	Operation Success;
	false	Operation Failed.

4. ISO18000-6B Tag Operations

4.1 Tag Selection Parameter Configuration

Description	This function is used to configure selective parameters when reader operates on ISO18000-6B tags. Reader would use the selected parameter to operate on ISO18000-6B tag next time, and realizing selective operation on multi ISO18000-6B tags of some type or one tag(UID code or several bytes in data, equal, not equal, greater than/ less than the set data).	
Prototype	bool SAAT_6BTagSelect (void* pHandle, unsigned char nType, unsigned char nStartAddr, unsigned char nDataBite, unsigned char * Data)	
Parameter	pHandle	the opening port handle
	nType	match type(as explanation)
	nStartAddr	start address of match data
	nDataBite	match mask
	Data	8bytes for matching data
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

definition of match type

0x00	SELECT	Equal
0x01	SELECT	Not equal
0x02	SELECT	Greater than
0x03	SELECT	Less than
0x04	UNSELECT	Equal
0x05	UNSELECT	Not equal
0x06	UNSELECT	Greater than
0x07	UNSELECT	Less than

Match the data start address: Indicating the start address to match the data, the maximum of match data should not exceed tag data bank. Currently, general data bank of ISO18000-6B tag is 224bytes, so the maximum is 216bytes.

Before using this function, user should better ask tag provider fot the length of tag data bank.

Match the data mask bit: Indicating from start address, which bytes are matching with matched data.

Match Data: Give the data which is to match tag data bank data, totally 8bytes.

Select command used to configure the reader back to the operating instructions specified only for a class of or one tag; the reader would cache the received parameter after receiving the selection instruction. On executing the next ISO18000-6B tag operation command, adding the parameter to the SELECT command (before sending tag operation command, every reader will send SELECT command), and enabling the command valid only to match tag.

If the reader doesn't receive tag selection command before executing tag operation command, the sending of command would use the default selected parameter.

Once the selection parameter configuration command is successful, the reader's next operation on ISO18000-6B tag will follow the configured parameter. After the operation, the selection parameter would restore to default, which means this command is a one-time operation, only effective for next reading.

If you need to use some selection parameter for long-term, you can execute "Tag Operation Parameter Configuration" command to modify the default tag selection command.

Selection command doesn't provide query function, when host computer is not sure about current parameters, and sending this command for re-configuration.

4.2 Read Tag UID Code

Description	This function is used to send "read ISO18000-6B tag UID code" command to reader.

Prototype	bool SAAT_6BReadUIDCode (void *pHandle,unsigned char nAntenna,unsigned char nType)	
Parameter	pHandle	the opening port handle
	nAntenna	operating antenna number(0-cycling query antenna, 1---antenna port 1, 2--antenna port 2, 3---antenna port 3, 4---antenna port 4)
	nType	operation mode
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

nType operation mode

0x00	Single reading mode, the reader would read only once and send all the tags' UID code been read, returning to host computer in turn, and later reader would auto-stop reading.
0x01	Cycling reading mode, reader returns all the UID code collected to host computer, and will auto-carry on this reading operation, till the host computer issue "stop operation" and then the reading operation will halt.

4.3 Receiving Tag UID Code

Description	This function starts receiving operation to receive tag UID code returning from reader.	
Prototype	int SAAT_6BRevUIDMsg (void *pHandle, unsigned char* nAntenna, unsigned char* pUIDData, unsigned char* nUIDLen)	
Parameter	pHandle	the opening port handle
	nAntenna	antenna number which reads UID code
	pUIDData	the received UID code data
	nUIDLen	incoming: length of pUIDData; outgoing: length of UID data(8bytes)
Return	true	Operation Success;
	false	Operation Failed.

4.4 Read Tag Data

Description	This function is used to reader data in tag data bank.	
Prototype	bool SAAT_6BReadUserData (void *pHandle ,unsigned char nAntenna,unsigned char nType,unsigned char * pTagID, unsigned char nStartAddr,unsigned char nReadLen, unsigned char *pdata, unsigned char dataLen)	
Parameter	pHandle	the opening port handle
	nAntenna	antenna number in read operation
	nType	operating mode, set as 0
	pTagID	the target tag 8bytesUID code to read
	nStartAddr	first target address of data bank to read
	nReadLen	the target data length
	pdata	the buffer pointer which stores the collected data
	dataLen	length of pdata
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

- This command requires reader to read within specified tag data bank some specified data content and returns to host computer.
- This is a single tag operation command, and must fill in specified tag UID code in "tag UID" parameter.
- Read tag "data start address", which range is from 0x00 to the last bytes address of data area, "data length" range is from 1 to tag total length of data area. It is required to meet "data area start address" + "data area length" not exceeding tag maximum address.

***ISO18000-6B** protocol defines reader in executing data area reading operation, the maximum is 8bytes each time, and to read more bytes, multiple operations is needed. This API can automatically check and cut off long data, so users don't need to write many times. It is recommended to write 64bytes every time, as prolonged operation increases the probability of interference.

*** The general ISO18000-6B tag** data bank is 224bytes, and the field maximum value is 22. Before using this function, user should better query tag provider for its data bank length parameter.

4.5 Write Tag Data

Description	This function is used to write data into tag data bank.	
Prototype	bool SAAT_6BWriteUserData (void* pHandle, unsigned char nAntenna, unsigned char nType, unsigned char *pTagID, unsigned char nStartAddrr, unsigned char *pValue, unsigned char *nLen)	
Parameter	pHandle	the opening port handle
	nAntenna	antenna number in write operation
	nType	operating mode, set as 0
	pTagID	the target tag 8bytesUID code to write
	nStartAddr	the first address to write in data bank(greater than 0x08)
	pValue	the target data to write
	nLen	incoming: length of pValue; outgoing: actual wrote data length
Return	true	Operation Success;
	false	Operation Failed.

***ISO18000-6B** protocol defines first 8bytes of tag as UID, usually has been locked in the factory, non-writable.

***ISO18000-6B** protocol defines when a reader executing data bank writes operation only 4bytes writable each time. If you want to write more bytes, the reader needs to execute several times of writing operation. But users don't need to read/write more times, as this API can automatically check and cut off long data, so users don't need to write many times. It is recommended to write 64bytes every time, as prolonged operation increases the probability of interference.

4.6 Lock Tag Data Area

Description	This function is used to execute tag data in non-reversible lock operation. The locked data bank data can not be rewritten or unlocked.	
Prototype	bool SAAT_6BTagLock (void* pHandle, unsigned char nAntenna, unsigned char nType, unsigned char *pTagID, unsigned char nStartAddrr, unsigned char nLen)	
Parameter	pHandle	the opening port handle
	nAntenna	antenna number which executing lock operation
	nType	lock operation mode, 0x00 is locked status
	pTagID	8bytes UID code which is to write in tag
	nStartAddrr	first address of data bank to be locked, 8~224bytes
	nLen	Data length of data bank to be locked, "length + start address" should not exceed tag's maximum length.
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
This command controls reader to execute lock operation on data bank.

Currently, general data bank of ISO18000-6B tags is 224bytes, and the maximum field is 224. Before using this function, users are recommended to ask the provider for length of tag data bank.

4.7 Query Tag Data Area Lock Status

Description	This function is used to query tag data bank lock status.	
Prototype	bool SAAT_6BTagLockQuery (void* pHandle, unsigned char nAntenna, unsigned char * pTagID, unsigned char nStartAddrr, unsigned char nLen, unsigned char *pData, unsigned char nDataLen)	
Parameter	pHandle	the opening port handle
	nAntenna	antenna number on querying operation
	pTagID	8bytes tag ID
	nStartAddrr	first address of queried data bank, 8~224bytes
	nLen	data length of queried data bank, "length + start address" should not exceed tag's maximum length

Return	true	Operation Success;
	false	Operation Failed.

5. ISO18000-6C Tag Operation

5.1 Tag Selection Parameters Configuration

Description	This function is used to configure the selection parameters when reader operates on the ISO18000-6C Tag. The reader will use the selection parameter at the next operation. This realizes making selectivity operation on multi ISO18000-6C Tags with similar properties (EPC code, TID code or some bytes data in user data field).	
Prototype	bool SAAT_6CTagSelect (void *pHandle, unsigned char nBank ,unsigned char nStartAddr,unsigned char MaskBit, unsigned char *Data ,unsigned char Datalength,unsigned char nSessionZone, unsigned char nActiveFlag, unsigned char nCutFlag)	
Parameter	pHandle	the opening port handle
	nBank	specify matching data field Bank(as explanation)
	nStartAddr	the start address of the specified matching data field Bank
	MaskBit	specify matching data bit number
	Data	specify matching data
	Datalength	data length
	nSessionZone	specify matching data session field(as explanation)
	nActiveFlag	specify matching data activity flag(as explanation)
	nCutFlag:	specify matching data cut off flag(as explanation)
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

Bank matching data field: indicating the data field need match

0x01	EPC data code
0x02	TID data code
0x03	USER data code

The first address of the matching data: It indicates the start address of the matching data. Tags from different manufacturers have different data are size, so does the start address range, but the max-size could not exceed the maximum matching data bank of tag. For specific data, please check with the data user manual. If you are not sure of the maximum data bank, try to read tag operation Band before any other operation. And through this way confirms the maximum data bank of tag. This field uses EVB data format, length of data is changeable. Please see the appendix A for definition of EVB format.

Match the Number of Bit: Indicates the bits of data for matching, taking Bit as a unit. Be aware that start address of matched data + matching bit number should not exceed the maximum value of data bank.

Match Data: Indicates the data for matching, and matched data is counted by bits while command is transmitted in bytes. When the match data is not full bytes, reader program will automatically cut the extra bits according to match bit number.

This function is used to set the reader tag operating parameters.

Selection command is only valid to next one operation. If you want it to be valid to all command, you can use "tag operation configuration command" to set the match rule as default.

Example:

Match Bit number = 13, then the match data should be

B7	B6	B5	B4	B3	B2	B1	B0	B7	B6	B5	B4	B3	B2	B1	B0
D	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0

D: Bit Data 0: Additional Bits

5.2 Read Tag EPC Code

Description	This function is used to instruct the reader executing "reader EPC code command".	
Prototype	bool SAAT_6CReadEPCCode (void *pHandle,unsigned char nAntenna, unsigned char nType, unsigned char nTagCount)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna number which is operating (0 is antenna rolling read, 1--antenna 1, … , 4--antenna 4)
	nType	operating mode 00: Reader returns EPC code of the same tag to host computer only once, and requires the host computer giving "return data confirmation" after receiving the EPC code. 01: Reader returns all the EPC code it collected to host computer, and doesn't require the host computer returning "return data confirmation". Till host computer sends "Amp off" command, the reader stops reading. This operating mode is commonly used.
	nTagCount	This function will also estimate the tag number, estimating the maximum tag number read simultaneously under working environment. It is designed to read 250pcs simultaneously, while the actual read tags exceed 500pcs.
Return	true	Operation Success;
	false	Operation Failed.

5.3 Receiving EPC Code

Description	This function is used to receive the EPC code reader returns, probably multiple EPC code is returned.	
Prototype	int SAAT_6CRevEPCMsg (void *pHandle, unsigned char* nAntenna, unsigned char* pEPCData, unsigned char* nEPCLen)	
Parameter	pHandle	the opening port handle
	nAntenna	antenna number which reads EPC code
	pEPCData	returning EPC code data
	nEPCLen	incoming: length of pEPCData; outgoing: length of EPC data, usually is 12
Return	1	Operation Success;
	0	Operation Failed.
	2	Heart beat package reader returns.

5.4 Reading Tag TID Code Command

Description	This function is used to instruct reader reading "tag TID code".	
Prototype	bool SAAT_6CReadTIDCode(void *pHandle, unsigned char nAntenna, unsigned char nType, unsigned char nTagCount)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna number which is operating (0 is antenna rolling read, 1--antenna 1, …, 4--antenna 4)
	nType	operating mode 00: Reader returns TID code of the same tag to host computer only once, and requires the host computer giving "return data confirmation" after receiving the EPC code. 01: Reader returns all the TID code it collected to host computer, and doesn't require the host computer returning "return data confirmation". Till host computer sends "Amp off" command, the reader stops reading. This operating mode is commonly used.
	nTagCount	This function will also estimate the tag number, estimating the maximum tag number read simultaneously under working environment. It is designed to read 250pcs simultaneously, while the actual read tags exceed 500pcs.
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

EPC global sets default bit of TID Bank of EPC Class1 Gen2 tags as 4 double-bits (words), which is 64bits. As the TID is getting marginalization in EPC applications, some factories even cancelled TID Bank. Reader underlying protocol will support different length of TID code.

5.5 Receive tag TID code order

Description	This function is used to receive the TID code sent by the ISO18000-6C Tag.	
Prototype	int SAAT_6CRevTIDMsg (void *pHandle, unsigned char* nAntenna, unsigned char* pTIDData, unsigned char* nTIDLen)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	pTIDData	TID data return by the reader
	nTIDLen	when input this is the pTIDData length and when output this is the length of TID data
Return	1	Operation Success;
	0	Operation Failed.
	2	The reader returns new TID.

5.6 Write EPC code

Description	This function is used to write EPC code	
Prototype	bool SAAT_6CWriteEPCCode(void* pHandle, unsigned char nAntenna, unsigned char nType, unsigned char *pAccessPWD, unsigned char *pWriteData, unsigned char nLen)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	nType	the operation type, the value is 0x00
	pAccessPWD	the password for visiting Tag, default value is 4 bytes 0
	pWriteData	the EPC code data needed to write in the Tag
	nLen	the length of the pWriteData
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
This instruction is used to write EPC code in the Tag.

EPC code length: Length of EPC code for setting Tag and the basic unit is double bytes(words).According to the EPC protocol the value range is 0x00~32H(when setting 00 the EPC Tag will be initialized to empty Tag).EPC global division rule that the EPC code default length is 6 double bytes.

EPC protocol requires the EPC data must be written by words unit and must be written continuously. To simplify the write operation, the reader is designed to only accept read and write EPC data use the starting position as the first address. And it has nonsupport to modify one middle data in the EPC code

There has different EPC data field size in Tags come from different manufacturers. So if the "EPC data length" is more the maximum EPC data filed length the operation will be fail and return the corresponding error code.

Before write the Tag the antenna port must be explicit.

5.7 Read user data Bank

Description	This function is used to get to User Bank data in ISO18000-6C Tag.	
Prototype	bool SAAT_6CReadUserData(void* pHandle, unsigned char nAntenna, unsigned int StartAddr, unsigned int nToReadLen, unsigned int nWaitTime ,unsigned char * UserData,unsigned char* pDataLen)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	StartAddr	the start address for data reading

	nLength	This is length of the reading data in the target tag, double-byte write and read. If length is 1 then read 2 bytes data or if length is 2 then read 4 bytes data and so on
	nWaitTime	the overtime for data reading(Calculated by ms unit, for 1 s the value is 1000)
	UserData	the data get from User Bank
	pDataLen	when input this the length of the UserData and when output this is the length of the reading data field
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

This function is used to configure the relevant parameter about the reader Tag operations.

There has different User Bank size in Tags come from different manufacturers. So before read please check the data sheet to make sure the total length (User data Bank first address +User data length) is no more than the User data Bank length. Or the operation will be fail and return the corresponding error code.

Here is no user data Bank in some tag, so when execute the operation the Tag will return error code.

"Maximum Tag data field length" is 64,when read more than 64 bytes data the upper computer should take multi read operations(This is limited by the FPGA decode field size).

5.8 Write user data Bank

Description	This function is used to write data in user data Bank of ISO18000-6C Tag.	
Prototype	bool SAAT_6CWriteUserData (void* pHandle, unsigned char nAntenna, unsigned char nType, unsigned char *pAccessPWD, unsigned int nStartAddr, unsigned int nWaitTime, unsigned char *pWriteData, unsigned int *pToWriteLen)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	nType	the operation type, the value is 0
	pAccessPWD	the password for visiting Tag, default value is 4 bytes 0
	nStartAddr	the start address for data writing
	inWaitTime	the overtime for data writing(Calculated by ms unit, for 1 s the value is 1000)
	pWriteData	the data for writing in data Bank of the Tag
	pToWriteLen	When input this is the length of the user data Bank and when output this is the length data actual writing in the data. Word (double byte) is the unit.
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

There has different User Bank size in Tags come from different manufacturers. So before write please check the data sheet to make sure the total length (User data Bank first address +User data length) is no more than the User data Bank length. Or the operation will be fail and return the corresponding error code.

There is no user data Bank in some tag, so when execute the operation the Tag will return error code.

The start address takes the EVB format. So the data length is variation. For the concrete EVB format please check in Appendix A.

"Write the tag data Bank" means the largest size can write in the data Bank. As the read operation, the reader at a time can write a word (double byte) data and the Tag need 20ms to write the data. When write too much data the probability for PF communication interfering between Tag and reader will increase. It is suggested that upper computer can maximum write 8 words at a time to the reader and divide data in several writing times when the data is larger than the limit.

Please ask supplier to get User data Bank length etc information.

5.9 Block write Bank data

Description	This function is used to block write Bank data of ISO18000-6C Tag.
Prototype	bool SAAT_6CWriteBankData (void* pHandle, unsigned char nAntenna, unsigned char nType, unsigned char *pAccessPWD, unsigned char nBank,

	unsigned char *pWriteData, unsigned char nLen)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	nType	the operation type, default value is 0
	pAccessPWD	the password for visiting Tag, default value is 4 bytes 0
	nBank	Bank for Tag data writing
	pWriteData	the user data for writing in the Tag
	nLen	the length of the writing data
Return	true	Operation Success;
	false	Operation Failed.

5.10 Block erase Bank Data

Description	This function is used to block erase Bank data of ISO18000-6C Tag.	
Prototype	bool SAAT_6CClearBankData (void* pHandle, unsigned char nAntenna, unsigned char nType, unsigned char *pAccessPWD, unsigned char nBank, unsigned char nStartAddr, unsigned char nLen)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	nType	the operation type, default value is 0
	pAccessPWD	the access password for visiting Tag, default value is0x00
	nBank	bank for Tag data writing
	nStartAddr	the start address of the Tag data for erasing
	nLen	the length of Tag data for erasing
Return	true	Operation Success;
	false	Operation Failed.

5.11 Configure Visiting password

Description	This function is used to set visiting password of ISO18000-6C tag.	
Prototype	bool SAAT_6CAccessPWDSet (void *pHandle, unsigned char nAntenna, unsigned char nType, unsigned char *pOrgPWD, unsigned char *pNewPWD)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	nType	in this operation the value is 0
	pOrgPWD	the primary 4 bytes visiting password, the default value is 4 bytes 0
	pNewPWD	the new 4 bytes visiting password
Return	true	Operation Success;
	false	Operation Failed.

Explanation:
The default visiting password is 4 bytes 0

If the Bank the password in is not locked, zero passwords can also execute the write operation (Get details in EPC protocol).

5.12 Change destruction password

Description	This function is used to Change the destruction password of ISO18000-6C Tag.	
Prototype	bool SAAT_6CDestroyPWDSet (void *pHandle, unsigned char nAntenna, unsigned char *pAccessPWD, unsigned char nType, unsigned char *pDestroyPWD)	
Parameter	pHandle	the opening port handle

	nAntenna	the antenna port number when operating tags(1-4)
	pAccessPWD	password for Tag visiting,4Bits
	nType	in this operation the value is 0
	pDestroyPWD	4 Bytes password for Tag destruction
Return	true	Operation Success;
	false	Operation Failed.

Explanation：

Visiting password: This password is for visiting the Tag. If the Tag destruction password is locked, the right visiting password is need to change the destruction password.

New destruction password: This password is a new one for destruction and the default value is "00000000".

EPC Protocol need nonzero password for the destruct operation.

If the Bank the password in is not locked, zero passwords can also execute the write operation (Get details in EPC protocol).

5.13 Tag lock status setting

Description	This function is used to set the tag lock status.	
Prototype	bool SAAT_6CTagLock (void *pHandle, unsigned char nAntenna, unsigned char *pAccessPWD, unsigned char nType, unsigned char nBank)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	pAccessPWD	password for tag visiting(4Bits)
	nType	the lock type, see the definition in the explanation.
	nBank	the executing Bank，see the definition in the explanation.
Return	true	Operation Success;
	false	Operation Failed.

Explanation：

Bank type：ISO18000-6C Every Bank of the tag can have lock operation independently and be configured is different lock status. It need to point the target Bank in the lock operation.

Code	definition
0x00	the whole Bank
0x01	TID Bank
0x02	EPC Bank
0x03	USER Bank
0x04	password for visiting
0x05	password for destruction

Lock operation type：ISO18000-6C protocol defines the following 4 lock operation.

Code	Definition
0x00	Lock
0x01	Deblocking
0x02	Permanent Lock
0x03	Permanent Open

The TID field in the EPC tag has been written with TID code and is configured irreversible permanent status. So there is no effect on TID field when execute "Unlock all the Blank" operation.

5.14 Tag Destruction

Description	This function is used to conduct Kill operation on tags	
Prototype	bool SAAT_6CKillTag (void *pHandle, unsigned char nAntenna,unsigned char *pDestroyPWD, unsigned char *pEPC, int nEPCLen)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)

	pDestroyPWD	password for tag destruction(4Bits）
	pEpc	EPC code for the target tag
	nEPCLen	EPC code length，default :12
Return	true	Operation Success;
	false	Operation Failed.

Explanation:

This deduction operation is irreversible. To avoid the mis-operation to other tags, an attached EPC code field should be added to the instruction. Only after reader getting the matching EPC code from the tag then the operation can be executed. If there is no added field destruction order will be executed without identification.

EPC code length has difference according to different signal tag. Default value is 12 bits.

The default password is "00000000", but according to the protocol the password must be changed before destruction operation.

5.15 EAS Bit zone setting

Description	This function is used to set the EAS bit zone.	
Prototype	bool SAAT_6CEASFlagSet(void *pHandle, unsigned char nAntenna, unsigned char nType, unsigned char* pAccessPwd, int nEASFlag)	
Parameter	pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	nType	ESA Setting operation mode
	pAccessPwd	tag visiting password
	nEASFlag	EAS Bit zone setting 0 cancel EAS Bit zone 1 set EAS Bit zone
Return	true	Operation Success;
	false	Operation Failed.

Explanation：

This function needs tags with this application, at present some tags of NXP support this function. Please ask supplier to get details.

5.16 Start EAS monitoring

Description	This function is used to start or stop EAS monitoring function.	
Prototype	bool SAAT_6CEASMonitorEnable (void *pHandle, unsigned char nAntenna, unsigned char nSetEAS)	
Parameter	Incoming Parameters: pHandle	the opening port handle
	nAntenna	the antenna port number when operating tags(1-4)
	nSetEAS	EAS monitoring operation control 0x00 -> Read end the EAS monitoring operation 0x01 -> Read start the EAS monitoring operation
Return	true	Operation Success;
	false	Operation Failed.

6. API Setting

6.1 API Versions information inquiry

Description	This function is used to inquire the current API function Versions information.	
Prototype	bool SAAT_Copyright (void *pHandle, char* copyright)	
Parameter	pHandle	the opening port handle
	copyright	versions information
Return	true	Operation Success;
	false	Operation Failed.

6.2 Setting API trends library language type

Description	This function is used to set API trends library language type.	
Prototype	bool SAAT_SetLanguageType(void *pHandle, char* szType)	
Parameter	pHandle	the opening port handle
	szType	language form cn Simplified Chinese tw Chinese Traditional en English.
Return	true	Operation Success;
	false	Operation Failed.

6.3 Get Error Message

Description	This function is used to get returning error message from reader.	
Prototype	bool SAAT_GetErrorMessage(void *pHandle,char *szMsg, int nLen)	
Parameter	pHandle	the opening port handle
	szMsg	returning error message
	nLen	szMsg buffer length
Return	true	Operation Success;
	false	Operation Failed.

6.4 Get Error Code

Description	This function is used to get returning error code from reader.	
Prototype	bool SAAT_GetErrorCode (void *pHandle,int *pCode);	
Parameter	pHandle	the opening port handle
	pCode	returning error code
Return	true	Operation Success;
	false	Operation Failed.

Part VI
Appendix A: Error code list

When executing order from upper computer failed, the reader will send back the error code and point out the operation failure reason.

Error code field is dividing as following:

0x10-0x1F	Reader system error code
0x20-0x2F	Reader operation error code
0x60-0x6F	Reader data transmission error code
0xA0-0xAF	ISO18000-6B Tag operation error code
0xB0-0xBF	ISO18000-6CTag operation error code

Error code is definitely defined as below:

Table A-1 Reader system error code list

Error code	Error code definition
0x10	Wrong software version in Reader decoding unit
0x11	Wrong hardware in Reader base band board
0x12	Wrong hardware in Reader RF board
0x13	Self-checking error in reader system parameter list
0x14	Self-checking error in Reader base band board
0x15	Self-checking error in Reader RF board
0x16	Ethernet Detection error in Reader
0x17	RTC Detection error in Reader
0x18	External Rom detection in Reader
0x19	Retain
0x1A	Retain
0x1B	Retain
0x1C	Retain
0x1D	Retain
0x1E	Retain
0x1F	Unknown system error code

Table A-2 Reader operation error code list

Error code	Error code definition
0x20	Operation password error
0x21	Using wrong antenna port number
0x22	Reader currently in testing mode
0x23	Get data in Reader internal storage device is failure
0x24	The Reader is Currently operating the tag

0x25	Run-time error, such as illegal operation in program run
0x26	Operation rights is blocked（Someone else has connected and used the Reader ）
0x27	Retain
0x28	Retain
0x29	Retain
0x2A	Retain
0x2B	Retain
0x2C	Retain
0x2D	Retain
0x2E	Retain
0x2F	Unknown Reader operation error

Table A-3 Data transmission error code list

Error code	Error code definition
0x60	Imperfect frames in Received instruction or data
0x61	CRC error in Received instruction or data
0x62	The current Reader cannot support this type the instruction
0x63	The current Reader cannot support the tag protocol
0x64	Parameter error in instruction
0x65	Instruction frames error（lack of domain or structure error in domain)
0x66	Cannot support this kind of instruction
0x67	Read receive too much instruction, and cannot handle temporarily
0x68	Retain
0x69	Retain
0x6A	Retain
0x6B	Retain
0x6C	Retain
0x6D	Retain
0x6E	Retain
0x6F	Unknown data transmission error

Table A-4 ISO18000-6B Tag operation error code list

Error code	Error code definition
0xA0	No response or no tag within RF range
0xA1	Tag doesn't support the command
0xA2	The command is not identified by tag, or address error
0xA3	This option is not supported
0xA4	Unknown error returned by tag
0xA5	Tag can not access a particular bank

0xA6	Specified bytes of tag is locked, and can not be re-locked
0xA7	Specified bytes of tag is locked, and can not be re-written
0xA8	Tag executing bytes of data writing failed
0xA9	Tag executing bytes locking failed
0xAA	Tag operation is cut down by host computer(such as carrier)
0xAB	Tag operation is not fulfilled due to interfering or others reasons
0xAC	Tag returns CRC error
0xAD	Retain
0xAE	Retain
0xAF	Unknown ISO18000-6B type tag operation error

Table A-5 ISO18000-6C Tag operation error code list

Error code	Error code definition
0xB0	No response or no tag within RF range
0xB1	The target storage bank of tag doesn't exist
0xB2	Tag return information: address of tag operation overflowing
0xB3	Tag return information: operation storage bank is locked
0xB4	Tag access password error
0xB5	Tag deactivate password error
0XB6	Tag operation is cut down by host computer(such as carrier)
0xB7	The tag, which is for reading/writing/locking data bank, has not been initialized.
0xB8	Uninitialized tag is found
0xB9	Tag return information: unknown error
0xBA	Tag return information: underpowered
0xBB	Retain
0xBC	Retain
0xBD	Retain
0xBE	Retain
0xBF	Unknown ISO18000-6C type tag operation error