development document

Content

1 Application development....1
1.1 The command format specification....1
1.1.1 Serial communication protocol structure....1
1.1.2 TCP/IP communication protocol structure....1
1.1.3 Data packet format definition....1
1.1.4 Instruction set definition....3
1.2 Basic command....5
1.2.2 EPC C1G2 Tag read and writes commands....18
1.3 SDK command function....24
1.3.1 Compose of SDK....24
1.3.2 Response Code of API....24
1.3.3 Define Data Type....25
1.3.4 Command and Control....32
1.3.5 Read-Write ISO18000-6C tag function....44

1 Application development

1.1 The command format specification

Communication protocol refers to the communication protocol of the PC machine through the RS-232 communication interface.

RS-485 communication interface in the data link layer to support the RS-232 protocol, but there is a protocol extension.

The data format of the byte oriented asynchronous communication protocol is adopted in the communication protocol. PC to the reader to the data frame for the command, reader data frame for the response is returned to the PC. In response to instructions or data frames are variable length bytes, the packet and checkout method and backward error.

The command or response data frame is 199 bytes

1.1.1 Serial communication protocol structure

1.1 RS232 parameter settings

Data sending and receiving of the signal is received by the physical layer, and the physical layer should be in accordance with the requirement of RS-232. Specific design requirements are as follows：

1 start bit, 8 bit data bit, 1 stop bit, no parity check；

The baud rate design for 9600bps and 19200bps, 38400bps, 57600bps, 115200bps optional.

1.1.2 TCP/IP communication protocol structure

In accordance with international standards。

1.1.3 Data packet format definition

1.1.3.1 Instruction format definition

The command frame is a data frame for the host operating reader, as shown in the following table：

Head	Len	Addr	Cmd	Data[0]	…	Data[n]	Check
0xA0	1 Byte	1 Byte	1 Byte	Byte 1		Byte n	1 Byte

Head Is a frame header flag, defined as 0xA0。

Addr Is a reader address, the general address from 0 to 254255 (0xFF) for public address. The reader only receives the instruction of the address and the address of the address.。

Len Packet length, the number of bytes in the Length domain。

Cmd Command code。

Data Is a parameter in the command frame。

Check It is the verification and the specified calibration range is the last byte of the frame header from the header to the parameter domain. The reader receives the instruction frame after the need to calculate the checksum to check。

1.1.3.2 Response frame format definition

The response frame is a data frame for the reader to return to the host, and the response frame contains the data that the reader needs to collect, and the format definition is shown in the following table：

Head	Len	Addr	Cmd	Status	Response	…	Response	Check
0x0A	1Byte	1Byte	1Byte	1Byte	Byte1		Byte n	1Byte

Head Is the response frame head, fixed to 0x0A。

Addr Is the reader's address。

Len Is the packet length, the number of bytes in the Length domain。

Cmd is executed instruction code reader。

Status The result of an action performed by the instruction is executed, and the 0 represents the success of the implementation, and the other is said to be wrong

False information。

Response Is the return data in the response frame。

Check It is the verification and the domain, which provides the verification range from the packet type to the last byte of the parameter domain. The PC receives the instruction frame after the need to calculate the checksum to check。

1.1.4 Instruction set definition

Instruction set list

NO.	Instruction code	Name	Explain
General operation instruction			
1	00H	Get_Version	Get hardware and software version
2	01H	Set_BaudRate	Set the serial communication baud rate
3	02H	Get_BaudRate	Get the serial communication baud rate
4	03H	Set_Address	Set reader address
5	04H	Get_Address	Access reader address
6	05H	Set_Net_Address	Set the reader network address
7	06H	Get_Net_Address	Access reader network address
8	07H	Set_Date_Time	Set reader clock
9	08H	Get_Date_Time	Get reader clock
10	09H	Reboot	Reset reader
11	0AH	Set_Relay	Control relay
12	0BH	Set_GPIO_Output	Set GPIO output port state
13	0CH	Get_GPIO_Output	GPIO output port level
14	0DH	Get_GPIO_Input	GPIO input port level
15	0EH	Set_Buzzer	Set buzzer
16	0FH	Set_Antenna_Parameter	Set antenna parameters
17	10H	Get_Antenna_Parameter	Gain antenna

			parameters
18	11H	Set_Antenna_Status	Set antenna state
19	12H	Get_Antenna_Status	Get antenna state
20	13H	Set_Frequency_Band	Set reader RF band
21	14H	Get_Frequency_Band	Get reader RF band
22	15H	Set_Max_Tag	Set up the maximum number of inventory tags
23	16H	Get_Max_Tag	Get the maximum number of inventory tags
24	17H	Set_Continuous_Mode	Set up the reader continuous working mode
25	18H	Get_Continuous_Mode	Continuous working mode for reader
26	19H	Get_Buzzer_Status	Get buzzer state
EPC C1G2 read and write instructions			
1	30H	EPC_Select_Criteria	C1G2 EPC tag selection criteria
2	31H	EPC_Direct_Inventory	C1G2 EPC Tags
3	32H	EPC_Read_Data	Read C1G2 EPC tag data
4	33H	EPC_Write_Data	Write C1G2 EPC tag data
5	34H	EPC_Lock_Memory	Lock C1G2 EPC tag data area
6	35H	EPC_Kill_Tag	Destroy C1G2 EPC Tags
7	36H	EPC_Set_Alarm	Set EAS alarm
8	37H	EPC_Detect_Alarm	EAS alarm
9	38H	EPC_MemBlock_Lock	Lock user area data block read

10	39H	Start_Continuous_Inventory	Start the reader for continuous working mode
11	3AH	Stop_Continuous_Inventory	Stop reader continuous working mode

1.2 Basic command

1.2.1.1 Get reader software and hardware version

Get_Version

Function: Get the hardware and software version number of the reader.

Instruction code: 00H.

Instruction parameters: N/A

Instruction packet: "AOH 03H address 00H CheckSum"

Return data: if the instruction is executed correctly, the data portion of the packet is returned as a version number of 4 bytes:

Byte0 - hardware major version, Byte1 - hardware minor version (using the hardware version number Instead of reader model)

Byte2 - software major version, Byte3 - software minor version

For example: if the reader model is Reader2004/2104/2114, the software version number is V1.3, then the packet is returned.：

0AH 08H address 00H 00H 14H 04H 01H 03H CheckSum

0AH 08H address 00H 00H 15H 04H 01H 03H CheckSum

0AH 08H address 00H 00H 15H 0EH 01H 03H CheckSum

Execution error：“0AH 04H address 00H status CheckSum”, statusNot equal to 0，Len byte info。

1.2.1.2 Set the serial communication baud rate

Set_BaudRate

Function: set the baud rate of serial work.

The initial communication rate of the reader is 115200bps. When the reader receives the instructions, according to the instructions to set the parameters of reader serial baud rate. Regardless of whether or not to close the reader power supply, the working rate will be maintained until the next reset.

Instruction code: 01H.

Instruction parameters: 1 bytes of BPS, value: 00H~04H, respectively, representing:

00H - 115200bps

01H - 57600bps

02H - 38400bps

03H - 19200bps

04H - 9600bps

Instruction packet et: "04H address 01H BPS CheckSum A0H"

Return data: If instructions executed correctly, Return Data part of the package (Return Data) for the state

“0AH 04H address 01H 00 CheckSum”

Execution error: "04H address 01H status CheckSum 0AH", status is not equal to 0, Len byte info。

1.2.1.3 Get the serial communication baud rate

Get_BaudRate

Function: get the reader serial baud.

Instruction code: 02H.

Instruction packet age: “A0H 03H address 02H CheckSum”

Return Data: if instructions executed correctly, Return Data part of the package (Return Data) are reader current baud rate BPS.

“0AH 05H address 02H 00 BPS CheckSum”

1 byte BPS, respectively:

04H - 9600bps

03H - 19200bps

02H - 38400bps

01H - 57600bps

Other - 115200bps

Execution error: “0AH 04H address 02H status CheckSum”, status is not equal to 0, Len byte info。

1.2.1.4 Set the address of the reader Set_Address

Function: set the address of the reader.

Instruction code: 03H.

Instruction parameters: 1 byte new address address_New. Address_Old is the old address.

Instruction packet et : " A0H 04H 255 (or address_Old) address_New CheckSum 03H"

Return data: If instructions executed correctly, return the package data section for the state .

"0AH 04H address_New 03H 00 CheckSum "

Execution error: "0AH 04H 255 (or address_Old) status CheckSum 0AH", status is not equal to 0, Len byte info。

1.2.1.5 Get the address Get_Address

Function: get the address of the reader.

Instruction code: 04H.

Instruction parameters:N/A

Instruction packet et: " A0H 03H 255 04H CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is returned to the reader's address address.

"0AH 05H 255 04H 00 address CheckSum "

Execution error: "0AH 04H 255 04H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.6 Set the reader network address Set_Net_Address

Function: set the network address of the reader. You must reset the reader to

use the new network address.

Instruction code: 05H.

Instruction parameter 1:4 byte the readr IP address Reader_IP_addr. 2 byte reader port number Reader_Port.

Instruction parameter 2:4 IP address Host_IP_addr. 2 byte host port number Host_Port.

Instruction parameter 3:4 byte reader subnet mask Reader_Mask. 4 byte reader gateway. 6 byte reader MAC address Reader_MAC.

Instruction packet et: “A0H 1DH address 05H Reader_IP_addr Reader_Port Host_IP_addr Host_Port Reader_Mask Reader_ Gateway Reader_MAC CheckSum”

For example: the reader IP addr is 192.168.1.12, reader port 5000, host IP addr is 192.168.1.103, host port for 3500, reader mask 255.255.255.0, reader for gateway 192.168.1.1, reader MAC 0X00,0x14,0x97,0x0F, 0x1D, 0xE3. Then the Instruction packet et is:

A0H 1DH address 05H C0H A8H 01H 0CH 13H 88H C0H A8H 01H 67H 0DH ACH FFH FFH FFH 00H C0H A8H 01H 01H 00H 14H 97H 0FH 1DH E3H CheckSum

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

“0AH 04H address 05H 00 CheckSum”

Execution error: " AH 04H address 05H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.7 Access reader network address Get_Net_Address

Function: get the network address of the reader.

Instruction code: 06H.

Instruction parameters:N/A

Instruction packet et: " A0H 03H address 06H CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is returned to the reader's network address.

"0AH 1EH address 06H 00 Reader_IP_addr Reader_Port Host_IP_addr Host_Port Reader_Mask Reader_ Gateway Reader_MAC CheckSum "

Reader_IP_addr represents the IP address of the 4 byte reader.

Reader_Port indicates the port number of 2 byte reader ports.

Host_IP_addr represents the 4 byte host IP address.

Host_Port represents the 2 byte host port number.

Reader_Mask represents a 4 byte reader subnet mask.

Gateway Reader_ represents a 4 byte reader gateway.

Reader_MAC represents the MAC address of the 6 byte reader.

Execution error: "04H address 06H status CheckSum 0AH", status is not equal to 0, Len byte info。

1.2.1.8 Set the reader clock Set_Date_Time

Function: set the reader time.

Instruction code: 07H.

Instruction parameters: 6 bytes, yy/, hh/, mm/, ff/,, dd/, ss.

Instruction packet : " A0H 09H address 07H yy mm dd hh ff ss CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 07H 00 CheckSum "

Execution error: "0AH 04H address 07H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.9 Get reader clock Get_Date_Time

Function: get the reader time.

Instruction code: 08H.

Instruction parameters: N/A.

Instruction packet : "03H address 08H CheckSum A0H"

Return data: if the instruction is executed correctly, the data portion of the packet is time.

"0AH address 08H YY 00 MM DD HH FF SS CheckSum 0AH"

Yy/ hh/ mm/ in dd/, ff/ seconds SS seconds.

Execution error: "04H address 08H status CheckSum 0AH", status is not equal to 0, Len byte info。

1.2.1.10 Reset reader Reboot

Instruction code: 09H.

Instruction parameters:N/A

Instruction packet : " A0H 03H address 09H CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 09H 00 CheckSum "

Execution error: "0AH 04H address 09H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.11 Control relay Set_Relay

Function: control relay.

Instruction code: 0AH.

Instruction parameters: 1 byte JDQ, 1 - closed, 0 - breaking.

Instruction packet : " A0H 04H address 0AH JDQ CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 0AH 00 CheckSum "

Execution error: "0AH 04H address 0AH status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.12 Set GPIO output port level state Set_GPIO_Output

Function: set GPIO output port level state.

Instruction code: 0BH.

Instruction parameters: 1 byte data OUT. Bit0 to set the output port 1, Bit1 to set the output port 2, Bit2 to set the output port 3, Bit3 to set the output port 4. Bit4 set output port 5, Bit5 set output port 6 and Bit6 to set the output port 7. Bit7 set output port 8. Bit* = 0, set the output port to be low. Bit* = 1, set the output port to be high.

Instruction packet : " A0H 04H address 0BH OUT CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 0BH 00 CheckSum "

Execution error: "0AH 04H address 0BH status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.13 GPIO output port level Get_GPIO_Output

Function: GPIO output port level state.

Instruction code: 0CH.

Instruction parameters:N/A

Instruction packet : " A0H 03H address 0CH CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is set to the level of OUT, which is defined as the setting output level instruction.

"0AH 05H address 0CH 00 OUT CheckSum

Execution error: "0AH 04H address 0CH status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.14 GPIO input port level Get_GPIO_Input

Function: GPIO input port level state.

Instruction code: 0DH.

Instruction parameters:N/A

Instruction packet : " A0H 03H address 0DH CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in a level state IN.

"0AH 05H address 0DH 00 IN CheckSum "

1 byte data IN. Bit0 gets the input port 1 state, access to the input port 2 state Bit1, access to the input port 3 state Bit2, access to the input port 4 state Bit3, access to the input port state Bit4, Bit5 gets the input port 6 state, Bit6 gets the input port 7 state, Bit7 won 8 state of the input port. Bit* = 0, the input port is low. Bit* = 1, the input port is high.

Execution error: "04H address 0DH status CheckSum 0AH", status is not equal to 0, Len byte info。

1.2.1.15 Set buzzer Set_Buzzer

Function: control buzzer sound and not ring.

Instruction code: 0EH.

Instruction parameters: 1 byte Buzzer, 1 --ring 0—no ring

Instruction packet : " A0H 04H address 0EH Buzzer CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 0EH 00 CheckSum "

Execution error: "0AH 04H address 0EH status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.16 Set antenna parameters Set_Antenna_Parameter

Function: set the working parameters of the antenna.

Instruction code: 0FH.

MS) for parameter instruction: 1 byte antenna number ant (antenna of the antenna 1 - 4), 2 byte RF power power (unit: DBM), two bytes of the working time of the antenna WTime (unit, antenna 1 bytes count the number of WCount.

Instruction packet : " A0H 09H address 0FH Ant Power WTime WCount CheckSum ".

For example: antenna 1, the RF power is 150dBm, the antenna working time is 1500ms, the antenna count number 0. Then the Instruction packet is:

A0H 09H address 0FH 01H 00H 96H 05H DCH 00H CheckSum.

Return data:

Execute correctly: "0AH 04H address 0FH 00H CheckSum ".

Execution error: "0AH 04H+Len address 0FH status info CheckSum ", status is not equal to 0. Len byte info。

1.2.1.17 Get antenna parameters Get_Antenna_Parameter

Function: get the working parameters of the antenna.

Instruction code: 10H.

Instruction parameters: 1 byte antenna number Ant (antenna 1 - 4).

Instruction packet : " A0H 04H address 10H Ant CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is provided with an antenna operating parameter, defined as a set of

antenna operating parameters.

"0AH 0AH address 10H 00H Ant Power WTime WCount CheckSum "

1 byte antenna number Ant (antenna of the antenna 1 - 4), 2 byte RF power power (unit: DBM), two bytes of the working time of the antenna WTime (unit, antenna 1 bytes count the number of WCount.

For example: the return of the parameters is: 1, the RF power is 15dBm, the antenna working time is 1500ms, the count is 1. Then the Instruction packet et is:

0AH 0AH address 10H 00H 01H 00H 96H 05H DCH 01H CheckSum

Execution error: "0AH 04H address 10H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.18 Set antenna state Set_Antenna_Status

Function: set the working status of the RF antenna.

Instruction code: 11H.

Instruction parameter: 1 byte antenna number Ant (antenna 1, antenna 4), 1 byte antenna state stat (0 not enable, 1 enable).

Instruction packet : "0 A0H 05H address 11H Ant stat CheckSum "

For example: antenna 1, so that can. Then the Instruction packet et is:

A0H 05H address 11H 01H 01H CheckSum

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 11H 00 CheckSum "

Execution error: "0AH 04H address 11H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.19 Get antenna status Get_Antenna_Status

Function: working status of RF antenna.

Instruction code: 12H.

Instruction parameters: 1 byte antenna number Ant (antenna 1 - 4).

Instruction packet : " A0H 04H address 12H Ant CheckSum "

Return data: if the instruction is executed correctly, the data portion of the

packet is provided with an antenna operating parameter, defined as the antenna state instruction set.

"0AH 06H address 12H 00 Ant stat CheckSum "

For example: the return parameter is: antenna 2, so that it can. Then return the package is:

0AH 06H address 12H 00H 02H 01H CheckSum

Execution error: "0AH 04H address 12H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.20 Set the reader RF band Set_Frequency_Band

Function: set the RF frequency band of the reader.

Instruction code: 13H.

Instruction parameters: 1 byte minimum frequency Min_Fre, 1 byte maximum frequency point Max_Fre.

Instruction packet : " A0H 05H address 13H Min_Fre Max_Fre CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 13H 00 CheckSum "

Execution error: "0AH 04H address 13H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.21 Access reader RF band Get_Frequency_Band

Function: to get the RF frequency band of the reader.

Instruction code: 14H.

Instruction parameters:N/A

Instruction packet : " A0H 03H address 14H CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is set to the frequency band, which is defined as the set of frequency bands.

"0AH 06H address 14H 00 Min_Fre Max_Fre CheckSum "

Execution error: "0AH 04H address 14H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.22 Set up a maximum count of Set_Max_Tag

Function: set up the number of tags.

Instruction code: 15H.

Instruction parameter: 1 bytes of NTag, NTag is 0, which means that the maximum number of tags is not limited.

Instruction packet : " A0H 04H address 15H NTag CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 15H 00 CheckSum "

Execution error: "0AH 04H address 15H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.23 Get the maximum count of Get_Max_Tag

Function: get the maximum number of tags.

Instruction code: 16H.

Instruction parameters:N/A

Instruction packet : " A0H 03H address 16H CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is defined as the antenna operating parameter, and the maximum number of tags are defined.

"0AH 05H address 16H 00 Ntag CheckSum "

Execution error: "0AH 04H address 16H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.24 Setting the reader continuous mode

Set_Continuous_Mode

Function: Set the reader continuous mode, the effect after the next power.

Script: 17H.

Command Parameter 1: 1 byte reader mode Continuous (0- discontinuous mode, 1-continuous mode, 2-trigger mode).

Command Parameter 2: 2-byte continuous mode counting interval time.

Command Parameter 3: 1 byte antenna Ant, antenna control bit 0, bit 1

controls the antenna 2, bit 2 controls the antenna 4, bit 3 controls the antenna 4, and so on; bit = 0, the antenna is not enabled, bit = 1, the antenna is enabled.

Command Parameter 5: 1 byte Out_port, tag data output interface, 0 = RS232,1 = RS485,2 = UDP, 3 = TCP.

Command Parameter 6: 1 byte Selected, = 1, select all the tags; = 2, select all tags except for specified tags; = 3, check the specified tag.

Command Parameter 7: 1 byte Session.

Command Parameter 8: 1 byte Target.

Command Parameter 9: 1 byte tag storage selection Men, for conditional inventory. 0 = password district, 1 = EPC area, 2 = TID district, 3 = user area.

Command Parameter 10: 2-byte starting address Str_addr (Unit: bits).

Command Parameter 11: 1 byte data length Length (unit: bits).

12:32 byte instruction parameters characteristic word Mask.

13:16 byte instruction parameters Rev, retained.

Instruction packet: " A0H 3EH address 17H Continuous time Ant Out_port Selected Session Target Men Str_addr Length Mask Rev CheckSum "

Returns data: If the command correctly, the data portion of the packet is returned to the state.

"0AH 04H address 17H 00 CheckSum "

Execution Error: "0AH 04H address 17H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.25 Get the reader continuous mode

Get_Continuous_Mode

Function: Get the reader continuous mode.

Script: 18H.

Parameters: None.

Instruction packet: " A0H 03H address 18H CheckSum "

Returns data: If the command correctly, the data portion of the packet is returned to 60 bytes reader continuously operating parameters.

"0AH 3FH address 18H 00 Continuous time Ant Out_port Selected Session Target Men Str_addr Length Mask Rev CheckSum "

Execution Error: "0AH 04H address 18H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.26 Get buzzer status Get_Buzzer_Status

Function: Get buzzer status (ring or not ring).

Script: 19H.

Instruction packet: " A0H 03H address 19H CheckSum "

Returns data: If the command correctly, the data portion of the packet is returned to the state of the beeper. 1 byte Buzzer, 1-- ring, zero - for no sound.

"0AH 05H address 19H 00 Buzzer CheckSum "

Execution Error: "0AH 04H address 19H status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.27 Set MAC address Set_Mac_Address

Function: set the MAC address of the reader.

Instruction code: 1AH.

Instruction parameter 1:6 byte reader MAC address Reader_MAC.

Instruction packet : " A0H 09H address 1AH Reader_MAC CheckSum "

For example: Reader_MAC is 0X00,0x14,0x97,0x0F, 0x1D, 0xE3. Then the Instruction packet et is:

A0H 09H address 1AH 00H 14H 97H 0FH 1DH E3H CheckSum

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 1AH 00 CheckSum "

Execution error: "0AH 04H address 1AH status CheckSum ", status is not equal to 0, Len byte info。

1.2.1.28 Get Reader MAC address Get_Mac_Address

Function: Get the reader MAC address.

Instruction code: 1BH.

Parameters: None.

Instruction packet et: " A0H 03H address 1BH CheckSum "

Returns data: If the command correctly, the data portion of the packet is returned to the reader MAC address.

"0AH 0AH address 1BH 00 Reader_MAC CheckSum "

Reader_MAC represents six byte MAC address of the reader.

Execution Error: "0AH 04H address 1BH status CheckSum ", status is not equal to 0, Len byte info。

1.2.2 EPC C1G2 Tag read and writes commands

1.2.2.1 Tag selection criteria EPC_Select_Criteria

Function: when the conditions set inventory tags. Only tags that meet these criteria can be identified.

Script: 30H.

Parameters: 1 byte storage area Bank. Two byte starting address Str_addr (unit: bit). 1 byte data length Length (unit: bit). 32 bytes characteristic word Mask. A byte Selected, = 1, select all the tags; = 2, select all tags except for specified tags; = 3, check the specified tag. 1 byte Session. 1 byte Target.

Instruction packet: "A0H 2AH address 30H Bank Str_addr Length Mask Selected Session Target CheckSum"

For example: Set EPC district, the starting address is 32, the data length is 24, characterized by word 11H 22H 33H, and select only the specified tag. The Instruction packet age for:

"A0H 2AH address 30H 01H 00H 20H 18H 11H 22H 33H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 00H 03H 00H 00H CheckSum"

Returns data: If the command correctly, the data portion of the packet is returned to the state.

"0AH 04H address 30H 00 CheckSum"

Execution Error: "0AH 04H address 30H status CheckSum", status is not equal to 0, Len byte info。

1.2.2.2 Inventory EPC C1G2 tag EPC_Direct_Inventory

Features: inventory tags. Only meeting the selection criteria in order to be recognized tag.

Script: 31H.

Parameters: None.

Instruction packet: "A0H 03H address 31H CheckSum"

Returns data: If the command correctly, the data portion of the packet is returned by EPC identification tag. Inventory to identify the number of tags, there are many packets. Each packet contains only one tag EPC.

"0AH 1AH + L address 31H 00H ant invinfo L EPC CheckSum".

1 byte ant: inventory number to the tag antenna (antenna 1- antenna 4).

20 bytes invinfo: inventory information.

A byte L: EPC length, unit: bytes.

Performed correctly: "0AH 04H address 31H 00H CheckSum".

Execution Error: "0AH 04H + Len address 31H status info CheckSum", status is not equal to 0. Len bytes info。

Table Definition 1.20 of bytes invinfo

First to 4 bytes	Fifth byte	Sixth byte	Seventh to eighth bytes	Ninth to tenth bytes	Eleventh to twentieth bytes
Retain	BroadbandRSSI	wide bandRSSI	Retain	RSSI	Retain

Among them RSSI(dBm)=(0x10000-0xRSSI)/10

1.2.2.3 EPC C1G2 tag data read EPC_Read_Data

Function: Reads the specified area specified tag data.

Script: 32H.

Condition Parameter 1: 1 byte storage area Bank. Two byte starting address Str_addr (unit: bit). 1 byte data length Length (unit: bit). 32 bytes characteristic word Mask. A byte Selected, = 1, select all the tags; = 2, select all tags except for specified tags; = 3, check the specified tag. 1 byte Session. 1 byte Target.

Password Parameter 2: 4 bytes Password.

3.1 bytes read parameter storage area Mem, 1 byte starting address Mem_addr (unit: word), one byte of data length Mem_len (unit: word, requires less than or equal to 90 and not 0).

Instruction packet et: "A0H 31H address 32H Bank Str_addr Length Mask Selected Session Target Password Mem Mem_addr Mem_len CheckSum"

For example: Password is 11223344, EPC read area starting at address 2, the data length of the content data 6, the Instruction packet:

"A0H 31H address 32H Bank Str_addr Length Mask Selected Session Target 11H 22H 33H 44H 01H 02H 06H CheckSum"

1 on condition of parameters, see '2.2.1 tag selection criteria' instructions.

Returns data: If the command correctly, the data portion of the packet is returned to Mem_len word tag data.

"0AH 04H + Mem_len * 2 address 32H 00 Data CheckSum"

Execution Error: "0AH 04H address 32H status CheckSum", status is not equal to 0, Len byte info。

1.2.2.4 Write EPC C1G2 tag data EPC_Write_Data

Function: write data to the specified area designation tag.

Script: 33H.

Condition Parameter 1: 1 byte storage area Bank. Two byte starting address Str_addr (unit: bit). 1 byte data length Length (unit: bit). 32 bytes characteristic word Mask. A byte Selected, = 1, select all the tags; = 2, select all tags except for specified tags; = 3, check the specified tag. 1 byte Session. 1 byte Target.

Password Parameter 2: 4 bytes Password.

Write Parameters 3.1 byte storage area Mem, 1 byte starting address Mem_addr (unit: word), one byte of data length Mem_len (unit: word, requires less than 74 and not equal to 0). Data to be written Data.

Instruction packet: "A0H 31H + Mem_len * 2 address 33H Bank Str_addr Length Mask Selected Session Target Password Mem Mem_addr Mem_len Data CheckSum"

Returns data: If the command correctly, the data portion of the packet is returned to the state.

"0AH 04H address 33H 00 CheckSum"

Execution Error: "0AH 04H address 33H status CheckSum", status is not equal to 0, Len byte info。

1.2.2.5 Lock EPC C1G2 tag data area EPC_Lock_Memory

Function: Locks the specified area designation tag.

Script: 34H.

Condition Parameter 1: 1 byte storage area Bank. Two byte starting address

Str_addr (unit: bit). 1 byte data length Length (unit: bit). 32 bytes characteristic word Mask. A byte Selected, = 1, select all the tags; = 2, select all tags except for specified tags; = 3, check the specified tag. 1 byte Session. 1 byte Target.

Password Parameter 2: 4 bytes Password.

Lock Parameter 3: 1 byte AREA, 0 - kill card password field, 1 - Access passwords area, 2 - EPC area, 3 - TID District 4 - User Area. 1 byte Lock, 0- any read-write (write), l always writable (write), 2-password can read and write (write), 3 never be read (write).

Instruction packet: "A0H 30H address 34H Bank Str_addr Length Mask Selected Session Target Password AREA Lock CheckSum"

Returns data: If the command correctly, the data portion of the packet is returned to the state.

"0AH 04H address 34H 00 CheckSum"

Execution Error: "0AH 04H address 34H status CheckSum", status is not equal to 0, Len byte info。

1.2.2.6 Destroy C1G2 EPC tags EPC_Kill_Tag

Function: destroy the specified tag.

Instruction code: 35H.

Conditional parameter 1:1 byte storage area Bank. 2 byte starting address Str_addr (unit: bit). 1 byte data length Length (unit: bit). 32 byte character Mask. 1 bytes of Selected, =1, select all tags; =2, select all tags except the specified tag; =3, select the specified tag. 1 byte Session. 1 byte Target.

Password parameter 2:4 Password.

Password parameter 3:4 Kill Password.

Instruction package: A0H 32H address 35H Bank Str_addr Length Mask Selected Session Target Password KillPassword CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 35H 00 CheckSum "

Execution error: "0AH 04H address 35H status CheckSum ", status is not equal to 0, Len byte info。

1.2.2.7 Set EAS alarm EPC_Set_Alarm

Function: set the EAS state for the specified tag.

Instruction code: 36H.

Conditional parameter 1:1 byte storage area Bank. 2 byte starting address Str_addr (unit: bit). 1 byte data length Length (unit: bit). 32 byte character Mask. 1 bytes of Selected, =1, select all tags; =2, select all tags except the specified tag; =3, select the specified tag. 1 byte Session. 1 byte Target.

Password parameter 2:4 Password.

The state parameters of 3:1 bytes (Eas 0 not alarm, alarm 1).

Instruction package:A0H 2FH address 36H Bank Str_addr Length Mask Selected Session Target Password Eas CheckSum "

Return data: if the instruction is executed correctly, the data portion of the packet is in the state.

"0AH 04H address 36H 00 CheckSum "

Execution error: "0AH 04H address 36H status CheckSum ", status is not equal to 0, Len byte info。

1.2.2.8 EAS alarm EPC_Detect_Alarm

Function: detect the EAS state of the tag.

Instruction code: 37H.

Parameters: No.

Instruction packet : " A0H 03H address 37H CheckSum "

Return data: if the tag is detected, the packet is returned to the packet:

"0AH 04H address 37H 00 CheckSum "

No alarm or execution error: "0AH 04H address 37H status CheckSum ", status is not equal to 0, Len byte info。

1.2.2.9 Lock the user area block read EPC_MemBlock_Lock

Function: Locks the specified area designation tag user area.

Script: 38H.

Condition Parameter 1: 1 byte storage area Bank. Two byte starting address Str_addr (unit: bit). 1 byte data length Length (unit: bit). 32 bytes characteristic word Mask. A byte Selected, = 1, select all the tags; = 2, select all tags except for specified tags; = 3, check the specified tag. 1 byte Session. 1 byte Target.

Password Parameter 2: 4 bytes Password.

Block lock Parameter 3: 1 byte Lock, below.

Data Block	Lock	Read-protect	Unread-protection
1	Bit7	=1	=0
2	Bit6	=1	=0
3	Bit5	=1	=0
4	Bit4	=1	=0
5	Bit3	=1	=0
6	Bit2	=1	=0
7	Bit1	=1	=0
8	Bit0	=1	=0

Instruction packet: "A0H 2FH address 38H Bank Str_addr Length Mask Selected Session Target Password Lock CheckSum"

Returns data: If the command correctly, the data portion of the packet is returned to the state.

"0AH 04H address 38H 00 CheckSum"

Execution Error: "0AH 04H address 38H status CheckSum", status is not equal to 0, Len byte info。

1.2.2.10 Start the reader continuous mode

Start_Continuous_Inventory

Function: Start the reader continuous mode.

Script: 39H.

Instruction packet et: "A0H 03H address 39H CheckSum"

Returns data: If the command correctly, the first return packets:

"0AH 04H address 39H 00 CheckSum".

Then there will be a series of return package, part of the identification tag is EPC data. Identify how many tags, there are many a return package, each return package is only a tag EPC.

"0AH 06H + L address 39H 00 ant L EPC CheckSum"

Wherein, ant represents inventory tag number antenna (antenna 1- antenna 4). L represents the length of EPC, unit: bytes.

Execution Error: "0AH 04H address 39H status CheckSum", status is not equal

to 0, Len bytes of info.

1.2.2.11 Stop the reader continuous mode

Stop_Continuous_Inventory

Function: Stop the reader continuous mode.

Script: 3AH.

Fixed parameters: 14H 0CH 0BH 01H 04H 11H 0EH 2DH

Instruction packet et: "A0H 0BH address 3AH 14H 0CH 0BH 01H 04H 11H 0EH 2DH CheckSum"

Returns data:

Start continuous mode,

If the command is correctly executed, the return package to: "0AH 04H address 3AH 00 CheckSum"

If the instruction execution error, no data is returned.

Continuous mode is not activated,

If the command is correctly executed, the return package to: "0AH 04H address 3AH 00 CheckSum"

If the instruction execution error, it returns an error status:

"0AH 04H address 3AH status CheckSum", status is not equal to 0, Len byte info。

1.3 SDK command function

1.3.1 Compose of SDK

SL144 provides application software development kit SDK, the development kit is mainly consists of the following documents:

（1）YKR2004DLL.dll-- Dynamic Link Library

（2）YKR2004LIBORDLL.h-- API function declaration file

1.3.2 Response Code of API

API function return code RFID_LIBRARY_API has the following information:

RFID_STATUS_OK=0 operation success

RESULT_UNKNOWN_ERROR=1, unknown error

RESULT_GENERAL_ERROR=2, general error

RESULT_INEXISTENT_CMD=3, there is no instruction

RESULT_ILLEGAL_CMD_PARAMS=4, the command parameter is not valid.

RESULT_FAIL_TO_RECEIVE_DATA=5, not received data

RESULT_NO_TAG=6, not recognized by the tag

RESULT_INCORRECT_PACKET_CONTENT=7, return packet data error

RESULT_RFID_INVALID_PARAMS=8, the reader parameter is invalid

RESULT_RFID_RADIO_BUSY=9, reader busy

RESULT_RFID_MAC_ERROR=10, reader MAC error

1.3.3 Define Data Type

```
typedef struct
{
    unsigned char connectflag;//0- Not connected；- Connected
    unsigned char connectindex;// Connection index
    unsigned char NETorUART;//0- Network connection；- Serial port connection

    // Network connection parameters
    unsigned char TCPorUDP;//0- TCP connection；- UDP connection
    char targetIP[32];// Target IP address
    unsigned short targetport;// Target port number
    char hostIP[32];// Local IP address
    unsigned short hostport;// Local port number

    // Serial connection parameters
    char serport[32];// Serial number
    unsigned int baudrate;// Baud rate
}CONNECTPARAMETER,*PCONNECTPARAMETER;

// Error code
```

```
enum ResultErrorCode
{
    RESULT_SUCCESS=0x00,
    RESULT_GENERAL_ERROR,
    RESULT_COM_ABNORMAL,
    RESULT_LOGIC_ABNORMAL,
    RESULT_INEXISTENT_CMD,
    RESULT_ILLEGAL_CMD_PARAMS,
    RESULT_OPERATION_FAILURE,
    RESULT_OPERATION_UNSUPPORTED,
    RESULT_NO_TAG,
    RESULT_WRONG_PASSWORD,
    RESULT_MEM_LOCATION_NOT_EXIST,
    RESULT_MEM_LOCATION_PROTECTED,
    RESULT_INSUFFICIENT_WRITE_POWER,
    RESULT_RFID_BUSY,
    RESULT_RFID_INVALID_PARAMS,
    RESULT_RFID_MAC_ERROR,

    COM_ABNORMAL=0x101,
    IS_BUSY,
    ILLEGAL_PARAMS,
    NOT_RECEIVE_DATA,
};

// Callback function (- correct, error)
typedef int (__stdcall * RFID_PACKET_CALLBACK_FUNCTION)(unsigned char
length,const unsigned char* pdata,void* context);

typedef struct
{
    unsigned char readerip[4];//Reader IP
```

```
    unsigned short readerport;//Reader port
    unsigned char hostip[4];//Host IP
    unsigned short hostport;//Host port
    unsigned char readermask[4];//Reader mask
    unsigned char gateway[4];//Reader gateway
}IP_ADDRESS;

typedef struct
{
    unsigned char Year;// Year
    unsigned char Month;// month
    unsigned char Day;//day
    unsigned char Hour;//hour
    unsigned char Minute;//minute
    unsigned char Second;//second
}Reader_DATE;

typedef struct
{
    unsigned char WorkMode;// Working mode
    unsigned long OperationInterval;// Operation time interval
    unsigned char Ant;// Working antenna
    unsigned char OutInterface;// Output interface
    unsigned char SelBank;
    unsigned short SelOffset;
    unsigned char SelCount;
    unsigned char SelMask[32];
    unsigned char SelTarget;
    unsigned char SelAction;
    unsigned char SelTruncate;
    unsigned char SelCtl;// Select control
    unsigned char QrySelected;
```

```
    unsigned char QrySession;
    unsigned char QryTarget;
    unsigned char OutMethod;// Output mode
    unsigned char OutInterval;// Output time interval
    unsigned char OutFormat;// Output format
    unsigned char TagRegion;// Tag storage area
    unsigned char TagAddress;// Tag starting address
    unsigned char TagLength;// Tag length
    unsigned char Rev[6];// Retain
}AUTOWORKPARAMETER;

typedef struct
{
    unsigned char antennaPort;
    unsigned short powerLevel;
    unsigned long dwellTime;
}RFID_ANTENNA_PORT_PARAMETER;

typedef struct
{
    unsigned char bank;
    unsigned short offset;
    unsigned char count;
    unsigned char mask[32];
}RFID_18K6C_SELECT_MASK;

typedef struct
{
    unsigned char target;
    unsigned char action;
    int enableTruncate;
}RFID_18K6C_SELECT_ACTION;
```

```
typedef struct
{
    RFID_18K6C_SELECT_MASK mask;
    RFID_18K6C_SELECT_ACTION action;
}RFID_18K6C_SELECT_CRITERION;

typedef struct
{
    unsigned char selected;
    unsigned char session;
    unsigned char target;
}RFID_18K6C_TAG_GROUP;

typedef struct
{
    unsigned char bank;
    unsigned char offset;
    unsigned char count;
}RFID_18K6C_READ_CMD_PARMS;

typedef struct
{
    RFID_18K6C_READ_CMD_PARMS readCmdParms;
    unsigned long accessPassword;
}CSTM_18K6C_READ_PARMS;

typedef struct
{
    unsigned char bank;
    unsigned char offset;
    unsigned char count;
```

```
    unsigned short* pData;
}RFID_18K6C_WRITE_SEQUENTIAL_CMD_PARMS;

typedef struct
{
    unsigned char writeType;
    union
    {
        RFID_18K6C_WRITE_SEQUENTIAL_CMD_PARMS sequential;
    }writeCmdParms;
    unsigned long accessPassword;
}CSTM_18K6C_WRITE_PARMS;

typedef struct
{
    unsigned char area;
    unsigned char action;
}RFID_18K6C_LOCK_CMD_PARMS;

typedef struct
{
    RFID_18K6C_LOCK_CMD_PARMS lockCmdParms;
    unsigned long accessPassword;
}CSTM_18K6C_LOCK_PARMS;

typedef struct
{
    unsigned long killPassword;
}RFID_18K6C_KILL_CMD_PARMS;

typedef struct
{
```

```
    RFID_18K6C_KILL_CMD_PARMS killCmdParms;
    unsigned long accessPassword;
}CSTM_18K6C_KILL_PARMS;

typedef struct
{
    unsigned char eas;
}RFID_18K6C_EAS_CMD_PARMS;

typedef struct
{
    RFID_18K6C_EAS_CMD_PARMS easCmdParms;
    unsigned long accessPassword;
}CSTM_18K6C_EAS_PARMS;

typedef struct
{
    unsigned char lock;
    unsigned long accessPassword;
}CSTM_18K6C_BLOCKLOCK_PARMS;

typedef struct
{
    unsigned short access_flg;
    unsigned short reg_addr;
    unsigned int reg_data;
}HOSTREGREQ;

typedef struct
{
    unsigned short rfu0;
    unsigned short reg_addr;
```

```
    unsigned int reg_data;
}HOSTREGRESP;

typedef struct
{
    unsigned char pkt_head;
    unsigned char pkt_len;
    unsigned char pkt_addr;
    unsigned char pkt_cmd;
    unsigned char pkt_status;
}CUSTOM_RFID_PACKET_COMMON;
```

1.3.4 Command and Control

(1) Connect reader

signed int RFID_Connect(PCONNECTPARAMETER conpara);

Function：

Establish a communication connection with the communication port connected to the reader

input parameters：

conpara：Communication port parameters

Return：

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(2) Disconnect the reader connection

signed int RFID_Disconnect(PCONNECTPARAMETER conpara);

Function：

Close connection with the reader, release the serial port resources.

input parameters：

conpara：Communication port parameters

(3) Get current reader error details

signed int Get_Current_Error_Detail(unsigned char* len,unsigned char* dat);

Function：

Get the current returned error code

Output parameters：

len：Length of the error message.

dat：Error code.

(4) Get software and hardware versions

signed int Get_Version(PCONNECTPARAMETER conpara,unsigned char address,unsigned short* phardver,unsigned short* psoftver);

Function：

Read reader hardware and software version numbers。

input port parameters：

conpara：Communication port parameters。

address：Reader Address

Output parameters:

phardver：Reader hardware version number

psoftver：Reader software version number

Return：

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(5) Set_BaudRate

signed int Set_BaudRate(PCONNECTPARAMETER conpara,unsigned char address,unsigned char bps);

Function：

Setting RS232 / RS485 port work baud rate。

input port parameters：

conpara：Communication port parameters。

address：Reader Address。

bps：Baud rate，9600，19200，38400，57600，115200。

Return：

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(6) Get the baud rate

signed int Get_BaudRate(PCONNECTPARAMETER conpara,unsigned char address,unsigned char* bps);

Function：

Get RS232 / RS485 port baud work。

input port parameters：

conpara：Communication port parameters。

address：Reader Address。

output port parameters：

bps：Baud rate，9600，19200，38400，57600，115200。

Return：

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(7) Set Reader Address

signed int Set_Address(PCONNECTPARAMETER conpara,unsigned char address,unsigned char newaddress);

Function：

Set reader RS485 port addresses, addresses from 0～254，255（0xFF）is the broadcast address.

Input parameter:

conpara: communication port parameters.

address: Reader address.

newaddress：New reader address

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(8) Get reader address

signed int Get_Address(PCONNECTPARAMETER conpara,unsigned char address,unsigned char* paddress);

Function：

Get the reader RS485 port addresses, addresses from 0～254，255（0xFF）is

the broadcast address.

Input parameter:

conpara: communication port parameters.

address: Reader address.

Output parameters:

paddress： Reader Address

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(9) Set network address

signed int Set_Net_Address(PCONNECTPARAMETER conpara,unsigned char address,const IP_ADDRESS* pipinfo);

Function：

Set reader network address, including IP and port.

Input parameter:

conpara: communication port parameters.

address: Reader address.

pipinfo： Reader Network Address

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(10) Get_Net_Address

signed int Get_Net_Address(PCONNECTPARAMETER conpara,unsigned char address,IP_ADDRESS* pipinfo);

Function：

Set reader network address, including IP and port.

Input parameter:

conpara: communication port parameters.

address: Reader address.

output parameters：

pipinfo： Reader Network Address

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(11) Set_Date_Time

signed int Set_Date_Time(PCONNECTPARAMETER conpara,unsigned char address,const Reader_DATE* ptime);

Function：

Setting the reader real-time clock.

Input parameter:

conpara: communication port parameters.

address: Reader address.

ptime: the current time.

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(12) Get_Date_Time

signed int Get_Date_Time(PCONNECTPARAMETER conpara,unsigned char address,Reader_DATE* ptime);

Function：

Get the reader time.

Input parameter:

conpara: communication port parameters.

address: Reader address.

Output parameters:

ptime: the current time.

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(13) Set_Relay

signed int Set_Relay(PCONNECTPARAMETER conpara,unsigned char address,unsigned char relay);

Function：

Control relay switch inside the reader.

Input parameter:

conpara: communication port parameters.

address: Reader address.

relay: 1- closed, 0 disconnected.

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(14) Set_GPIO_Output State level

signed int Set_GPIO_Output(PCONNECTPARAMETER conpara,unsigned char address,unsigned char output);

Function：

Control the reader port output level.

input parameter：

conpara：Communication parameters.

address：Reader Address.

output：Bit0—setoutput port1，Bit1—setoutput port2，Bit2—setoutput port3，Bit3—setoutput port4，Bit4—setoutput port5，Bit5—setoutput port6，Bit6—setoutput port7，Bit7—setoutput port8。BitX＝0，set the output port is low. BitX = 1, setting the output port is high.

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(15) Get_GPIO_Output State level

signed int Get_GPIO_Output(PCONNECTPARAMETER conpara,unsigned char address,unsigned char* poutput);

Function：

Control the reader port output level.

input parameter：

conpara：Communication parameters.

address：Reader Address.

output parameters：

output：Bit0—output port1，Bit1—output port2，Bit2—output port3，Bit3—output port4，Bit4—output port5，Bit5—output port6，Bit6—output port7，Bit7—output port8。BitX＝0， so input port is the low level. BitX＝1， so input port is the high level.

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(16) Get_GPIO_Input

signed int Get_GPIO_Input(PCONNECTPARAMETER conpara,unsigned char address,unsigned char* pinput);

Function：

Get the reader port input level state.

input parameter：

conpara：Communication parameters.

address：Reader Address.

output parameters：

pinput：Bit0—input port1，Bit1—input port2，Bit2—input port3，Bit3—input port4， Bit4—input port5， Bit5—input port6， Bit6—input port7， Bit7—input port8。BitX＝0，so input port is the low level. BitX＝1，so input port is the high level.

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(17) Set_Buzzer

signed int Set_Buzzer(PCONNECTPARAMETER conpara,unsigned char address,unsigned char buzzer);

Function：

Set the reader buzzer.

input parameter：

conpara：Communication parameters.

address：Reader Address.

buzzer1 - ring, 0 - not ring.

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(18) Get_Buzzer_Status

signed int Get_Buzzer_Status(PCONNECTPARAMETER conpara,unsigned char address,unsigned char* pbuzzer);

Function：

Get the reader buzzer.

input parameter：

conpara： Communication parameters.

address： Reader Address.

output parameters：

pbuzzer： 1 - ring, 0 - not ring.

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(19) Set_Antenna_Parameter

signed int Set_Antenna_Parameter(PCONNECTPARAMETER conpara,unsigned char address,const RFID_ANTENNA_PORT_PARAMETER* pantparams);

Function：

Set the reader antenna working parameters.

input parameter：

conpara： Communication parameters.

address： Reader Address.

pantparams： The number of antenna, the antenna rf power and working time.

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(20) Get Antenna parameter

signed int Get_Antenna_Parameter(PCONNECTPARAMETER conpara,unsigned char address,unsigned char ant,RFID_ANTENNA_PORT_PARAMETER*

pantparams);

Function：

Gets reader antenna working parameter

input parameter：

conpara：Communication parameters.

address：Reader Address.

ant：antenna number.

output parameters：

pantparams：The antenna of the radio frequency power and working time.

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(21) Set antenna status

signed int Set_Antenna_Status(PCONNECTPARAMETER conpara,unsigned char address,unsigned char ant,unsigned char antstatus);

Function：

Set the reader antenna working status.

input parameters：

conpara：Communication input parameters

address：Reader Address.

ant：antenna number

antstatus：0- unable，1- enable

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(22) Get antenna status

signed int Get_Antenna_Status(PCONNECTPARAMETER conpara,unsigned char address,unsigned char ant,unsigned char* pantstatus);

Function：

Set the reader antenna working status.

input parameters：

conpara：Communication input parameters

address：Reader Address.

ant：antenna number

output parameters：

antstatus：0- unable，1- enable

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(23) Set the radio frequency spectrum

signed int Set_Frequency_Band(PCONNECTPARAMETER conpara,unsigned char address,unsigned char minfreq,unsigned char maxfreq);

Function：

Set the reader rf frequency

input parameters：

conpara：Communication parameters.

address：Reader Address.

minfreq：working minimum frequency

maxfreq：working maximum frequency

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(24) Get radio frequency spectrum

signed int Get_Frequency_Band(PCONNECTPARAMETER conpara,unsigned char address,unsigned char* pminfreq,unsigned char* pmaxfreq);

Function：

Get reader Rf frequency

input parameters：

conpara：Communication parameters.

address：Reader Address

output parameters:

minfreq：working minimum frequency

maxfreq：working maximum frequency

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(25) Set max tag number

signed int Set_Max_Tag(PCONNECTPARAMETER conpara,unsigned char address,unsigned char maxtagcnt);

Function：

Set the reader most identification tag number， reader According to the parameters set multiple tags algorithm.

input parameters：

conpara：Communication parameters.

address：Reader Address

maxtagcnt：reader Most identification tag number.

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(26) Get Most identification tag number

signed int Get_Max_Tag(PCONNECTPARAMETER conpara,unsigned char address,unsigned char* pmaxtagcnt);

Function：

set reader Most identification tag number, reader According to the parameters set multiple tags algorithm

Input parameters：

conpara：Communication input parameters

address: Reader Address

output parameters

pmaxtagcnt：reader most identification tag number。

Return：

If the function return value is RESULT_SUCCESS, the operation is successful, otherwise fail.

(27) Set MAC address

signed int Set_Mac_Address(PCONNECTPARAMETER conpara,unsigned char

address,const unsigned char* pmac);

Function：

Setting the reader network MAC address

Input parameters：

conpara：Communication input parameters

address：Reader Address

pmac：Reader MAC address

Return：

If the function return value is RESULT_SUCCESS , the operation is successful, otherwise fail.

(28) Get MAC address

signed int Get_Mac_Address(PCONNECTPARAMETER conpara,unsigned char address,unsigned char* pmac);

Function：

Get the reader network MAC address

Input parameters：

conpara：Communication input parameters

address: Reader address

output parameters：

pmac：readerMAC address

Return：

If the function return value is RESULT_SUCCESS operation is successful, otherwise the operation fails

(29) Stop continuous working mode

signed int Stop_Continuous_Mode(PCONNECTPARAMETER conpara,unsigned char address);

Function：

Stop the reader continuous working mode turning into command mode

input parameters：

conpara：Communication input parameters。

address：Reader address

Return：

If the function return value is RESULT_SUCCESS, operation is successful, otherwise the operation failed.

(30) Reboot reader

signed int Reboot(PCONNECTPARAMETER conpara,unsigned char address);

Function：

Reset reader

Input parameters：

conpara：COM input parameters

address：reader address

return：

If the function return value is RESULT_SUCCESS，operation is successful, otherwise the operation failed.

1.3.5 Read-Write ISO18000-6C tag function

(1) Sets the specified tag inventory status

signed int EPC_Select_Criteria(PCONNECTPARAMETER conpara,unsigned char address,const RFID_18K6C_SELECT_CRITERION* pCriterion,unsigned char autosel);

Function：

Command the reader to the specified tag, set to a specific status。

Input parameters：

conpara：Communication input parameters。

address：Reader address。

pCriterion：Selection Criteria。

autosel：Whether the selection criteria, 0 is not used, 1 use。

Return：

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(2) Set inventory statuss

signed int EPC_Query_Group(PCONNECTPARAMETER conpara,unsigned char

address,const RFID_18K6C_TAG_GROUP* pGroup);

Function：

Set reader inventory statuss, in line with the statuss of the tag inventory in order to be inventory.Input parameters：

conpara：Communication input parameters.

address：Reader address.

pGroup：Inventory statuss。

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(3) EPC C1G2 tag inventory

signed int EPC_Direct_Inventory(PCONNECTPARAMETER conpara,unsigned char address,RFID_PACKET_CALLBACK_FUNCTION pCallback,void* context);

Function：

Command reader inventory tag.

Input parameters:

conpara：Communication input parameters.

address：Reader address.

pCallback：The callback function receives the tag data.

context：Retention parameters.

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(4) EPC C1G2 tag reader

signed int EPC_Direct_Read(PCONNECTPARAMETER conpara,unsigned char address,const RFID_18K6C_SELECT_CRITERION* pCriterion,const RFID_18K6C_TAG_GROUP* pGroup,const CSTM_18K6C_READ_PARMS* pParms,unsigned char* pData);

Function：

Command reader reads the tag data specified area.

Input parameters:

conpara：Communication input parameters.address：Reader Address。

pCriterion：Reserved.。

pGroup：Reserved.

pParms：Specify the data storage area, the start address and data length.

output parameters

pData：Read data.

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(5) EPC C1G2 tag write

signed int EPC_Direct_Write(PCONNECTPARAMETER conpara,unsigned char address,const RFID_18K6C_SELECT_CRITERION* pCriterion,const RFID_18K6C_TAG_GROUP* pGroup,const CSTM_18K6C_WRITE_PARMS* pParms);

Function：

Command to write data to the tag reader specified area.

Input parameters:

conpara: communication input parameters.

address: Reader address.

pCriterion: Reserved.

pGroup: Reserved.

pParms: Specifies the data storage area, the start address, data length, and data.

Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(6) EPC C1G2 tag lock

signed int EPC_Lock_Memory(PCONNECTPARAMETER conpara,unsigned char address,const RFID_18K6C_SELECT_CRITERION* pCriterion,const RFID_18K6C_TAG_GROUP* pGroup,const CSTM_18K6C_LOCK_PARMS* pParms);

Function：

Command reader lock tag specified area.

Input parameters:

conpara: communication input parameters.
address: Reader address.
pCriterion: Reserved.
pGroup: Reserved.
pParms: Specifies the data storage area and how to lock.
Returns:
If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails.

(7) EPC C1G2 Kill_Tag

signed int EPC_Kill_Tag(PCONNECTPARAMETER conpara,unsigned char address,const RFID_18K6C_SELECT_CRITERION* pCriterion,const RFID_18K6C_TAG_GROUP* pGroup,const CSTM_18K6C_KILL_PARMS* pParms);

Function：

Command reader to Kill the specified tag

Input parameters:
conpara：communication input parameters.
address: Reader address.
pCriterion：Reserve

pGroup：Reserve

pParms：kill password and access password

Returns：

If the function return value is RESULT_SUCCESS，Operation is successful,or the operation fails.

(8) Set EAS status

signed int EPC_Set_Alarm(PCONNECTPARAMETER conpara,unsigned char address,const RFID_18K6C_SELECT_CRITERION* pCriterion,const RFID_18K6C_TAG_GROUP* pGroup,const CSTM_18K6C_EAS_PARMS* pParms);

Function：

Command reader sets EAS tag status. Only suitable for NXP company tag

Input parameters:
conpara: communication input parameters.
address: Reader address.
pCriterion: Reserve.
pGroup: Reserve.
pParms: access password and EAS status, 1 alarm, 0 no alarm.
Returns:

If the function return value is RESULT_SUCCESS , said the operation was successful, otherwise the operation fails

(9) Detection EAS status

signed int EPC_Detect_Alarm(PCONNECTPARAMETER conpara,unsigned char address);

Function：

Command reader detects tags EAS status.

Input parameters:

conpara: communication input parameters.

address: Reader address.

Returns:

If the function return value is RESULT_SUCCESS , then there is an alarm, otherwise no alarm.