Hopeland

Focus on RFID core technology

Hopeland RFID READER - PC

Development Guide Python

Editor: Paul

Shenzhen Hopeland Technologies Co., Ltd

V 1.1

CONTENTS

1. Overview

1.1 Content Overview

In order to ease the secondary development of users, we have developed a function library that can run on the Python platform. The library is written in Python and the development environment is python 3.7.
This development guide introduces the corresponding technical indicators, application development instructions and precautions, application interface function descriptions, etc.

1.2 Applicable Models

This document lists all RFID devices' API, the following table lists supportable function of different models (please refer to specific notes of function module details)

Function module	Applicable models
Connection operation	All products
Device configuration	All products
RFID configuration	All products
GPO operation	All products
6C tag operation	All products

1.3 Copyright Statement

2. Object mode controls the reader

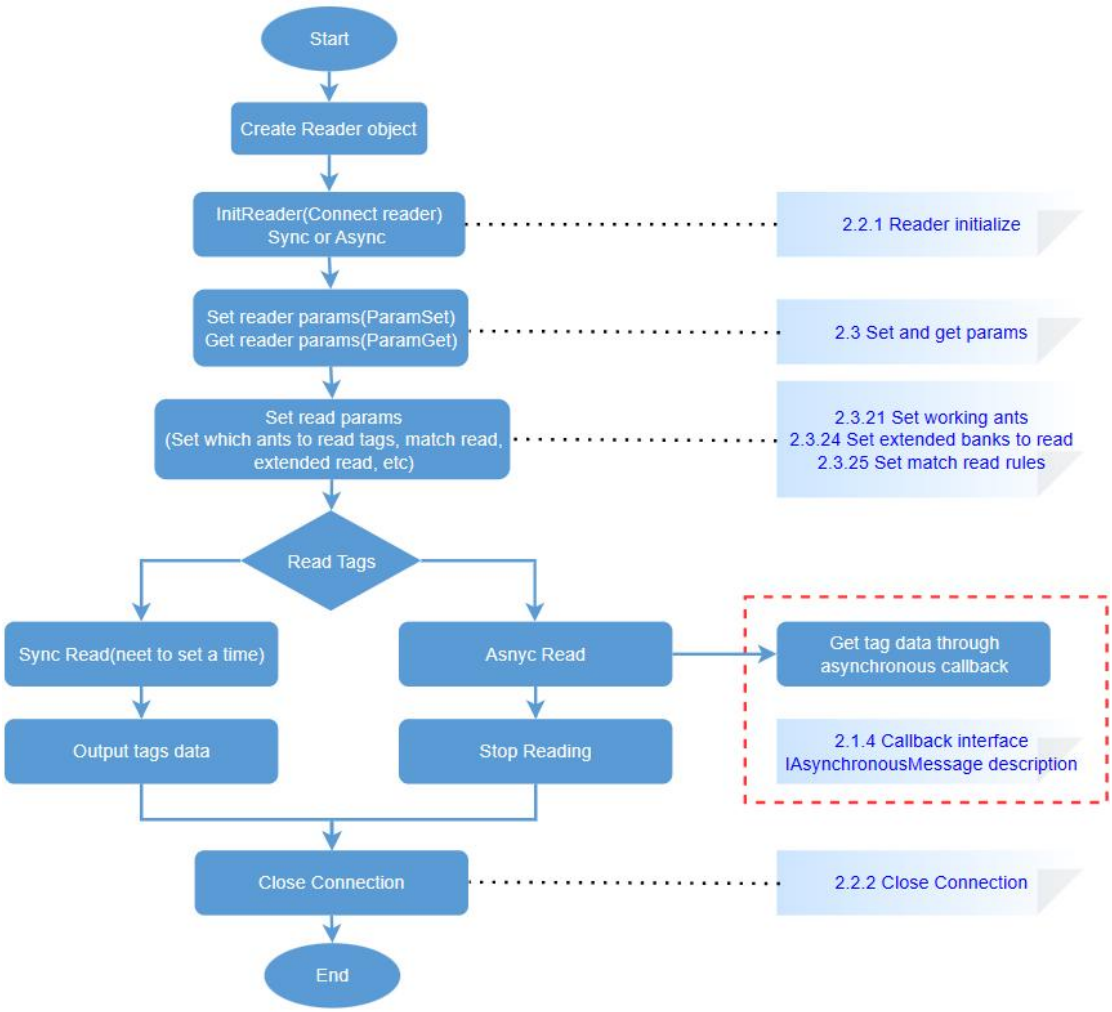
2.1 API function module description

Cd(change directory) to the directory where the whl file is located, run the following command to install the library file.
pip install RFIDReaderAPI-1.0-py3-none-any.whl.
The following dependent libraries also need to be installed.

pip install gmssl.
pip install hidapi.
pip install pyserial.
pip install pyusb

2.1.1 Development Process

The object-oriented development process is as follows:



2.1.2 Functional Modules

The main functional modules of Reader are basic functions and device configuration functions.
Basic functions include connecting, disconnecting readers, reading, writing, locking, destroying tags, restarting devices, factory resets, etc.
The read and write functions of the device configuration can refer to the EReaderEnum function enumeration.

R/W	Enumeration name	Description
Read only	**RO_ReaderInformation**	**Reader Information**
	RO_ReaderAntennaStandingWaveRatio	Antenna port VSWR
	RO_ReaderTemperature	**Reader Temperature**
	RO_ReaderGPIState	**Reader GPI State**
Reader R/W	**RW_ReaderSerialPortParam**	RS232 params
	RW_Reader485Param	RS485 params
	RW_ReaderNetwork	Network configuration (IP,Mask,Gateway,MAC address)
	RW_ReaderTime	reader time(UTC, NTP server)
	RW_ReaderWorkMode	TCP Server / Client mode
	RW_ReaderBuzzerSwitch	buzzer(control and switch)
	RW_ReaderStateLED	LED Status indicator
	RW_ReaderDataOutputFormat	Tag output format
	RW_ReaderCustomCode	Custom ID
RFID R/W	**RW_RFIDAntPower**	Antenna power (Ant No., power, enable)
	RW_RFIDRF	RF configuration (frequency band, frequency point)
	RW_RFIDEpcBasebandParam	EPC Baseband Params
	RW_RFIDTagUploadParam	Tag Upload Params
	RW_RFIDAutoIdleParam	Reader Auto Idle Mode
Write only	**WO_RFIDAntPlan**	Reader Working Antenna Configuration
	WO_RFIDReadExtended	Write (Read Extended Area)
	WO_RFIDReadSpecial	Write (Read Special Area)
	WO_RFIDReadTagFilter	Write (Read Filter Rules)

2.1.3 Auxiliary classes

Read extension class

Namespace	com.rfid.models
Constructors	ReadExtendedArea_Model (**EReadBank** bank, **Integer** readStart, **Integer** readLen,**String** passWord)
Params	Bank: The area to be read during inventory. ReadStart: Start address in bank. readLen: The number of words to read. PassWord: Access password

Tag filter class

Namespace	com.rfid.models

Constructors	TagFilter_Model(**EReadBank** Bank,**Integer** tagStart,**String** tagLen,**String** data)
Params	Bank: The area to be read during inventory. tagStart: Start address in bank, Match read unit is bit, match write unit is word tagLen: data length Data: tag data need to be matched

Antenna standing wave detection class

Namespace	com.rfid.models
Constructors	AntennaStandingWave(**EAntennaNo** antNum)
Params	antNum: Antenna No. forwardPower: Forward power detection value backwardPower: Backward power detection value returnLoss: Return loss standingWaveRatio: Standing wave ratio

EPC baseband params

Namespace	com.rfid.models
Constructors	EpcBaseband_Model(**EBasebandRate** eBasebandRate, **Integer** qValue, **Integer** session, **ESearchType** searchType)
Params	eBasebandRate: EPC baseband rate qValue: The starting Q value used by the reader. Session: session value searchType: Inventory flag parameters (FlagA only, FlagB only, FlagA and FlagB double-sided inventory)

EPC extended Baseband params

Namespace	com.rfid.models
Constructors	EPCExtendedParam_Model ()
Params	TAG extended parameter object: TagExtendParam properties are: IMJ_Tag_Focus: bool type, IMJ_Tag_Focus function switch IMJ_Fast_Id: bool type, IMJ_Fast_Id function switch NXP_Fast_ID: bool type, NXP_Fast_ID function switch DNQ extended parameter object: DQNExtendParam properties are: MaxQ: int type, decimal MinQ: int type, decimal Tmult: int type, decimal AutoQ: bool type, Dynamic start Q function switch ForceQ: bool type, Forced loop algorithm function switch AST extended parameter object: ASTExtendParam properties are: AntSwitchMode: antenna working switching mode, EASTSwitchMode enumeration (Switch_Immediately_Without_Tags, Running_Out_Of_Residence_Time).

	Retry: int type, decimal, number of retries (number of retries identified without tags, a reference option for antenna switching) ResidenceTime: int, decimal, maximum antenna residence time (x10ms) AST2 extended parameter object: ASTExtendParam2 properties are: WaitingTime:int type, decimal, antenna switching wait time (x10ms) AntStep:int type, decimal, antenna switching step value AntThreshold: int, decimal, antenna protection threshold (return loss dBm). Set it to 0 to disable protection. LBT extended parameter object: LBTExtendParam properties are: WorkMode: working mode, ELBTWorkMode enumeration (Disable, LBT_Listening_Only, Read_Tag_After_Listening, Read_Tag_After_Meeting_RSSI) MaxRSSI: RSSI maximum

Working antenna

Namespace	com.rfid.models
Constructors	ReaderAntPlan_Model (**Integer**[] antennas)
Params	Antennas: Integer array. An array of antennas used to configure the reader to read tags, which can be one antenna or multiple antennas.

Antenna configuration auxiliary class

Namespace	com.rfid.models
Constructors	ReaderWorkingAntSet_Model(**EAntennaNo** antennaNo, **Integer** power,**Boolean** enable)
Params	antennaNo: Antenna No. power: Power in dBm enable: Enable

Reader automatic idle mode class

Namespace	com.rfid.models
Constructors	ReaderAutoSleep_Model(**Boolean** autoIdleSwitch,**Integer** time)
Params	autoIdleSwitch: Automatic idle mode switch Time: Automatic idle time, Unit 10ms

Reader generic Boolean class

Namespace	com.rfid.models
Constructors	ReaderBoolean_Model (**Boolean** flag)
Params	Flag: Pass data of type Boolean.

Buzzer auxiliary class

Namespace	com.rfid.models
Constructors	ReaderBuzzer_Model (**EBuzzerControl** buzzerControl, **Boolean** buzzerSwitch,

	EBuzzerType buzzerType)
Params	buzzerControl: EBuzzerControl enumeration (Reader control: ReaderControl, that is, after reading a tag, the built-in buzzer of the reader will sound; upper computer control: PCControl, that is, through buzzerSwitch and buzzerType to control whether the buzzer sounds.) buzzerSwitch: True is on, false is off buzzerType: EBuzzerType enumeration (Once: ring once, Always: ring all the time)

Tag output format class

Namespace	com.rfid.models
Constructors	ReaderDataOutput_Model(**EOutputSwitch** outputSwitch, **EOutputFormat** outputFormat, **TagFilter_Model** tagFilter, **String** startData, **String** endData)
Params	outputSwitch: Special output format switch, EOutputSwitch enumeration (Close, Open, UDPOutput) outputFormat: Output format, EOutputFormat Enumeration (Hex, ASCII, Decimal) tagData: Tag upload class, configure the matching area, start byte, and character length. (Matching area: EPC, TID) startData: Start of Text endData: End of Text

Reader GPI/O info

Namespace	com.rfid.models
Constructors	ReaderGPIParam_Model(**EGPI** GPINum, **ETriggerStart** triggerStart, **ETriggerCode** triggerCode, **ETriggerStop** triggeerStop, **Integer** delayTime, **Boolean** isUpload, **String** customCMD)
Params	GPINum: GPI port number triggerStart: Trigger start condition,eTriggerStart enumeration(OFF, Low_level, High_level, Rising_edge, Falling_edge, Any_edge) triggerCode: Trigger execution command, eTriggerCode enumeration triggeerStop: Trigger stop condition, eTriggerStop enumeration(OFF, Low_level, High_level, Rising_edge, Falling_edge, Any_edge、Delay) delayTime: Stop delay time isUpload: Upload Flag customTriggerCode: Custom command

GPI status class

Namespace	com.rfid.models
Constructors	ReaderGPIState_Model(**HashMap<EGPI, EGPIState>** dicState)
Params	dicState: Contains the GPI port number and the corresponding status

Reader information class

Namespace	com.rfid.models
Constructors	ReaderInfo_Model(**String** softVersion, **String** name, **Long** powerTime, **String** basebandVersion, **String** readerSN, **Integer** minPower, **Integer** maxPower, **Integer** antCount, **List<ERF_Range>** RFList, **List<Integer>** protocolList)
Params	softVersion: Software version name: Reader name powerTime: The number of seconds elapsed by the reader from the moment of power up to the current moment basebandVersion: Baseband version readerSN: reader SN minPower: Minimum power maxPower: Maximum power antCount: Number of antenna ports RFList: Frequency band list protocolList: Protocol list

Hidden LED light auxiliary class

Namespace	com.rfid.models
Constructors	ReaderLED_Model (**Boolean** LEDState,**Integer** LEDTime)
Params	LEDState: Integrated reader hidden light strip switch. True is on LEDTime: Turn on time after reading a tag, in milliseconds.

Network configuration auxiliary class

Namespace	com.rfid.models
Constructors	ReaderNetWork_Model(**String** ip, **String** mask, **String** gateway, **String** dns) The constructor only configures the IPv4 address, subnet mask, gateway, and DNS, and configures the rest according to the situation.
Params	Mac: reader MAC address Ipv4Address: reader Ipv4 address Ipv4Mask: reader Ipv4 mask Ipv4GateWay: reader Ipv4 gateway Ipv4Dns: reader Ipv4 DNS DhcpSwitch: DHCP switch, True is on Ipv6Switch: Ipv6 switch Ipv6Address: reader Ipv6 address Ipv6Mask: reader Ipv6 mask Ipv6GateWay: reader Ipv6 gateway Ipv6Dns: reader Ipv6 DNS

Frequency band auxiliary class

Namespace	com.rfid.models
Constructors	ReaderRF_Model(**ERF_Range** readerWorkFrequency, **EWF_Mode** RFHoppingMode, **List<Integer>** readerWorkPoint)

Params	readerWorkFrequency: Working frequency band RFHoppingMode: Frequency hopping mode, EWF_Mode enumeration(Specified, Auto) readerWorkPoint: Specify the frequency points, such as "0", "3"

Serial port class

Namespace	com.rfid.models
Constructors	ReaderSerial_Model(**Integer** address, **EBaudrate** baudrate), for RS485 connection. ReaderSerial_Model(**EBaudrate** baudrate), for RS232 connection.
Params	Address: RS232/RS485 serial port no. Baudrate: Serial baud rate

Reader generic string class

Namespace	com.rfid.models
Constructors	ReaderString_Model (**String** data)
Params	data: Used to pass string data

Tag upload auxiliary class

Namespace	com.rfid.models
Constructors	ReaderTagUpdate_Model(**Integer** repeatTimeFilter, **Integer** rssiFilter, **Integer** dBmFilter)
Params	repeatTimeFilter: Duplicate tag upload filter time rssiFilter: RSSI filter dBmFilter: RSSI_dBm Threshold (optional parameters, not supported by some devices)

Reader time auxiliary class

Namespace	com.rfid.models
Constructors	ReaderTime_Model(**String** utc, **Boolean** ntp_Switch, **String** ip)
Params	utc: UTC Date and time, for example: "2023.10.10 10:00:00" ntp_Switch: NTP switch, True is on ip: NTP server IP address

Server/Client Mode Auxiliary Class

Namespace	com.rfid.models
Constructors	ReaderWorkMode_Model(**EWorkMode** workMode, **String** ip, **Integer** port)
Params	workMode: TCP working mode, EWorkMode enumeration(Server,Client) Ip: If the Client mode is selected, you need to configure an IP address. The IP address is the IP address of the Server to which the reader actively connects in Client mode. port: Specified port

2.1.4 Callback Interface IAsynchronousMessage

```
# Asynchronous callback information interface
class IAsynchronousMessage:
    __metaclass__ = ABCMeta # Specifies that this is an abstract class.

    '''
    Output debugging information
    @param msg Debugging information
    '''
    @abstractmethod
    def WriteDebugMsg(self,connID,msg):
        pass

    '''
   Output log information
    @param msg Log information
    '''
    @abstractmethod
    def WriteLog(self,connID, msg):
        pass

    '''
    Client connection callback in TCP server mode
    That is, the reader works in TCP Client mode to actively connect to the Server, and the program on the Server acts as TCP server to monitor the connection request actively sent from the reader.
    @param connID Connection identification
   '''
    @abstractmethod
    def PortConnecting(self,connID):
        pass

    '''
    Disconnected callback. When a device is disconnected, the API calls back the connection ID, indicating that the device with the current connection ID is disconnected.
    @param connID Connection identification
    '''
    @abstractmethod
    def PortClosing(self,connID):
        pass
```

```
        '''
        Output tag information callback
        @param tag Tag information
        '''
        @abstractmethod
        def OutputTags(self,tag):
            pass

        '''
        Notice of the end of reading tags
        '''
        @abstractmethod
        def OutputTagsOver(self,connID):
            pass

        '''
        GPI trigger message callback
        @param gpi_model GPI Information Class
        '''
        @abstractmethod
        def GPIControlMsg(self,connID,gpi_model):
            pass

        '''
        Output barcode scanning data callback for Bluetooth handheld
    device
        @param scandata Scanned barcode data
        '''
        @abstractmethod
        def OutputScanData(self,connID,scandata):
            pass
```

Call back method	Remark
WriteDebugMsg	Print API internal process debugging information.
WriteLog	API logging callback (currently not open).
PortConnecting	The reader works in TCP client mode and actively connects to the TCP server program. The connection ID returned by the TCP server program after successfully establishing a connection with the reader. After obtaining the connection ID from the callback, you can control the reader through the connection ID.
PortClosing	When the device is disconnected, the API will call back the connection ID, indicating that the device with the current connection ID is disconnected.
OutputTags	The tag data read by the reader is obtained through this callback interface. Note: When the API processes asynchronous callbacks for tag data, do not

	process complex logic in the callback to ensure that the cached data in the API is cleared in time.
OutputTagsOver	After the last tag is uploaded, a synchronization end signal is uploaded, indicating the end of the current read tag action.
GPIControlMsg	gpiModel:GPI infrared trigger message callback. This function will call back the GPI port number of the current event, as well as the level status information, and so on. The device needs to be configured with appropriate GPI parameters to trigger the GPI callback API. For example, the GPI parameters are set as follows. Trigger start condition: high level, Stop condition: low level, Then when the GPI state is switched from low level to high level, the trigger start information is sent out through this callback, and when the GPI state is switched from high level to low level, the trigger stop information is sent out through this callback.

Callback Data Tag_Model Field Introductions

Field	Remark
_ReaderName	Reader connection ID, which represents which reader read this tag data, example: "192.168.1.116:9090"
_ReaderSN	Reader SN, this field will only exist after calling the GetSN interface
_TagType	Tag type，”6c”,”6b”,”gb” 3 types.
_EPC	Tag EPC data, Hexadecimal string.
_PC	Tag PC value
_ANT_NUM	Antenna no., which represents which antenna read the tag data
_RSSI	RSSI value
_TID	Tag TID, Hexadecimal string.
_UserData	Tag user area data, Hexadecimal string.
_RsvdData	Tag password area data, includes access password and kill password, Hexadecimal string.
_ReadTime	The time the tag was read is displayed by the time inside the reader

2.2 Basic Functions

2.2.1 Reader Initialize (connect a reader)

There are two ways to initialize:
Asynchronous initialization: The data callback interface is opened, and the callback interface contains various callback interface methods, such as the tag

information callback interface. Asynchronous mode is recommended.

Synchronization initialization: the interface of data callback is enclosed in the Reader class. When calling the tag reading method, the tag reading time is required. After reading, the tag data read by the reader will be returned together. (Thread will be blocked during reading)

Asynchronous initialization

Namespace	com.rfid.Reader
Function	**initReader (string param, IAsynchronousMessage log)**
Params	param: Connection mode + connection parameter. Log: Data callback interface, from which all tag data will be called back. Serial connection, such as "Serial:"+"COM1:115200" or "RS232:"+"COM1: 115200" RS458 connection, such as ”RS485:” + ”1:COM1:115200” TCP connection, such as "TCP:" + ”192.168.1.116:9090” USB connection, such as "USB:" + connection parameter USB connection parameter can be obtained by the static method GetUsbHidDeviceList () function in the same namespace.
Return	True succeeded, False failed.
Remark	1. **log** Please refer to chapter 2.1.4 Callback Interface for the data callback interface.
Example code	log = Text() reader = Reader() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): print(**"Connection created successfully!"**) **else**: print(**"Failed to create connection!"**)

Synchronization initialization

Namespace	com.rfid.Reader
Function	**initReader (string param)**
Params	Connection mode + connection parameter. Serial connection, such as "Serial:"+"COM1: 115200" or "RS232:"+"COM1: 115200" RS458 connection, such as ”RS485:” + ”1:COM1:115200” TCP connection, such as "TCP:" + ”192.168.1.116:9090” USB connection, such as "USB:" + connection parameter USB connection parameter can be obtained by the static method GetUsbHidDeviceList () function in the same namespace.
Return	True succeeded, False failed.
Remark	1. **After reading the tag data, it is returned by the read method.**

Example code	reader = Reader() **if** reader.initReader(**"TCP:192.168.1.116:9090"**): print(**"Connection created successfully!"**) **else**: print(**"Failed to create connection!"**)

2.2.2 Close Connection

Namespace	com.rfid.Reader
Function	**void closeConnect()**
Params	None
Return	None
Remark	log = Text() reader = Reader() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): print(**"Connection created successfully!"**) reader.closeConnect() *# close the current connection*

2.2.3 Restart Reader

Namespace	com.rfid.Reader
Function	**void restartReader ()**
Params	None
Return	None
Remark	log = Text() reader = Reader() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): print(**"Connection created successfully!"**) reader.restartReader() *# restart the reader*

2.2.4 Reset Factory

Namespace	com.rfid.Reader
Function	**EReaderResult setReaderRestoreFactory()**
Params	None
Return	EReaderResult RT_ OK succeeded, otherwise failed
Remark	Restore the reader settings to the factory state and use this interface carefully (the MAC address and reader time will not change).
Example code	log = Text() reader = Reader() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): **if** reader.setReaderRestoreFactory() == EReaderResult.RT_OK:

	print("Factory settings restored successfully!") else: print("Failed to restore factory settings!") else: print("Failed to create connection!") reader.closeConnect()

2.2.5 Read Tags(asynchronous)

Namespace	com.rfid.Reader
Function	**EReaderResult inventory ()**
Params	inventory: Asynchronous mode can be called directly to start reading.
Return	EReaderResult RT_ OK succeeded, otherwise failed
Remark	Send asynchronous inventory command. If no configuration is configured, antenna 1 is used. No filtering and extended query is required. If you need to read TID or other tag memory banks, see extended read function EReaderEnum.WO_RFIDReadExtended in paramSet. In asynchronous mode, data is transmitted through a callback interface. See Callback interface OutputTags.

Example code	log = Text()
	reader = Reader()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	if reader.inventory() == EReaderResult.RT_OK:
	print("Start reading successfully!)
	else:
	print("Failed to start reading")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

2.2.6 Read Tags(synchronous)

Namespace	com.rfid.Reader
Function	**EReaderResult read(int milliseconds, Object val)**
Params	milliseconds: Read duration in milliseconds. Note: this method will block the thread for this duration. Val: Return value, return List < Tag_Model >.
Return	EReaderResult RT_ OK succeeded, otherwise failed
Remark	Send synchronous inventory command. If no configuration is configured, antenna 1 is used. No filtering and extended query is required. If you need to read TID or other tag memory banks, see extended read function EReaderEnum.WO_RFIDReadExtended in paramSet.

Example code	reader = Reader() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, None): tag_modelList = [] readerResult = reader.read(1500, tag_modelList) **for** item **in** tag_modelList: Reader.print_object(item) **else**: print(**"Failed to create connection!"**)

2.2.7 Stop Reading

Namespace	com.rfid.Reader
Function	**EReaderResult stop ()**
Params	None
Return	EReaderResult RT_ OK succeeded, otherwise failed
Remark	Stop the reader from reading tags
Example code	log = Text() reader = Reader() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): **if** reader.inventory() == EReaderResult.RT_OK: print(**"Start reading successfully!"**) time.sleep(1.5) *# Stop reading* reader.stop() **else**: print(**"Failed to start reading"**) **else**: print(**"Failed to create connection!"**)

2.2.8 Write a Tag

Namespace	com.rfid.Reader
Function	**EReaderResult writeTag(TagFilter_Model** setWritingRules**, TagFilter_Model matchFilter,String passWord)**
Params	setWritingRules: Write rules, including (write area, write starting address, write data) Write areas include Reserved, EPC, UserData Write start address: decimal, write length in word. EPC usually starts at 2, while others start at 0. matchFilter: Filter rules (matching area, starting index of matching data, matching data)

	Matching areas include EPC, TID, and UserData Matching start index: decimal, matching length in bit. EPC usually starts at 32, while others start at 0. passWord: access password
Return	EReaderResult RT_ OK succeeded, otherwise failed
Remark	Match the tag according to the filtering rule matchFilter, and then write the information in the writing rule setWritingRules to this tag.

Example code	log = Text()
	reader = Reader()
	if reader.initReader(**"TCP:192.168.1.116:9090"**, log):
	# Match the tag with EPC of 00C35800000000000000C359
	matchFilter = TagFilter_Model(EReadBank.EPC, 32, **"00C35800000000000000C359"**)
	# Write EPC value as 00C35800000000000000C358.
	setFilter = TagFilter_Model(EReadBank.EPC, 2, **"00C35800000000000000C358"**)
	# Execute tag writing with the password of 00000000.
	if reader.writeTag(setFilter, matchFilter, **"00000000"**) == EReaderResult.RT_OK:
	print(**"Write the tag successfully!"**)
	else:
	print(**"Failed to write the tag!"**)
	else:
	print(**"Failed to create connection!"**)
	
	Note: if you start writing data from EPC starting address 2, SDK will automatically rewrite the PC value according to the length of the written data.
	If you start writing data from EPC starting address 1, then SDK will think that the developer is rewriting the PC value, and SDK will not automatically rewrite the PC value.

2.2.9 Lock a Tag

Namespace	com.rfid.Reader
Function	**EReaderResult lockTag(ELockArea lockArea,ELockType lockType,TagFilter_Model filter,String passWord)**
Params	lockArea: Lock area enumeration (destroy password, access password, EPC, UserData) lockType: Lock type enumeration (unlock, lock, permanent unlock, permanent lock) Filter: Filter rules (matching area, starting index of matching data, matching data) Matching areas include EPC, TID, and UserData

	Matching start index: decimal, matching length in bit. EPC usually starts at 32, while others start at 0. passWord: access password
Return	EReaderResult RT_ OK succeeded, otherwise failed
Remark	Find the tag according to the filter rule, and then lock or unlock the lock area by type.

Example code	log = Text()
	reader = Reader()
	if reader.initReader(**"TCP:192.168.1.116:9090"**, log):
	# Match the tag with TID of E280110520005346C94402C1
	filter = TagFilter_Model(EReadBank.TID, 0, **"E280110520005346C94402C1"**)
	# Execute the lock tag, lock EPC, and the access password is 111111111. The access password of the tag is 00000000 by default. You need to change the access password of the tag to non-zero in advance before you can successfully operate the lock tag with the new access password.
	if reader.lockTag(ELockArea.epc, ELockType.Lock, filter, **"11111111"**) == EReaderResult.RT_OK:
	print(**"Lock the tag successfully!"**)
	else:
	print(**"Failed to lock the tag!"**)
	else:
	print(**"Failed to create connection!"**)

2.2.10 Kill a Tag

Namespace	com.rfid.Reader
Function	**EReaderResult destroyTag(TagFilter_Model filter,String passWord)**
Params	filter: Filter rules (matching area, starting index of matching data, matching data) Matching areas include EPC, TID, and UserData Matching start index: decimal, matching length in bit. EPC usually starts at 32, while others start at 0. passWord: kill password
Return	EReaderResult RT_ OK succeeded, otherwise failed
Remark	Find the tag according to the filter rule, and then kill the tag.

Example code	log = Text()
	reader = Reader()
	if reader.initReader(**"TCP:192.168.1.116:9090"**, log):
	# Match the tag with TID of E280110520005346C94402C1
	filter = TagFilter_Model(EReadBank.TID, 0, **"E280110520005346C94402C1"**)
	# Execute the kill tag, the kill password is 111111111. The kill password of the tag is 00000000 by default. You need to change the kill password of the tag to

	non-zero in advance before you can successfully operate the kill tag with the new kill password.
	`if reader.destroyTag(filter, "11111111") == EReaderResult.RT_OK:`
	`  print("Kill the tag successfully!")`
	`else:`
	`  print("Failed to kill the tag!")`
	`else:`
	`  print("Failed to create connection!")`

2.2.11 Get error information

Namespace	com.rfid.Reader
Function	**static String getDetailError(EReaderResult rr)**
Params	EreaderResult Result enumeration
Return	Structure description
Remark	Get detailed information based on structure enumeration
Example code	print(Reader.getDetailError(EReaderResult.RT_OK))

2.2.12 Set the error message language

Namespace	com.rfid.Reader
Function	**static void setLanguage(ELanguage language)**
Params	Elanguage enumeration(Chinese, English)
Return	None
Remark	Configure the language pack for error information query.
Example code	reader = Reader() reader.setLanguage(ELanguage.Chinese) print(Reader.getDetailError(EReaderResult.RT_OK))

2.3 Device configuration and query function

In the Reader class, there are two methods to query and set the reader.

2.3.1 Get Params

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Enumeration parameters that need to be queried Val: Query results are returned based on the actual situation
Return	EReaderResult Result enumeration
Remark	Query the configuration of Reader and RFID

2.3.2 Set Params

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Enumeration parameters that need to be configured Val: Object parameters that need to be configured
Return	EReaderResult Result enumeration
Remark	Configure Reader and RFID

2.3.3 Reader Info(read-only)

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum. RO_ReaderInformation Val: Return object ReaderInfo_Model
Return	EReaderResult Result enumeration Properties in ReaderInfo: softVersion: Software version name: Reader name powerTime: The number of seconds elapsed by the reader from the moment of power up to the current moment basebandVersion: Baseband version readerSN: reader SN minPower: Minimum power maxPower: Maximum power antCount: Number of antenna ports RFList: Frequency band list protocolList: Protocol list
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerInfo = ReaderInfo_Model() readerResult = reader.paramGet(EReaderEnum.RO_ReaderInformation, readerInfo) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(readerInfo) **else**: print(**"Query failed!"**) **else**: print(**"Failed to create connection!"**)

2.3.4 Antenna VSWR (read-only)

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RO_ReaderAntennaStandingWaveRatio Val: Return object AntennaStandingWave_Model Property antNum is required to specify an antenna port.
Return	EReaderResult Result enumeration AntennaStandingWave_Model properties are: antNum: Antenna No. forwardPower: Forward power detection value backwardPower: Backward power detection value returnLoss: Return loss standingWaveRatio: Standing wave ratio
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): antennaStandingWave = AntennaStandingWave_Model(EAntennaNo._1) readerResult = reader.paramGet(EReaderEnum.RO_ReaderAntennaStandingWaveRatio, antennaStandingWave) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(antennaStandingWave) **else**: print(**"Query failed!"**) **else**: print(**"Failed to create connection!"**)

2.3.5 Reader Temperature (read-only)

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RO_ReaderTemperature Val: Return object ReaderString_Model
Return	EReaderResult Result enumeration Data in ReaderString_Model.data: Reader temperature
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerString = ReaderString_Model() readerResult = reader.paramGet(EReaderEnum.RO_ReaderTemperature,

	readerString)
	if readerResult == EReaderResult.RT_OK:
	Reader.print_object(readerString)
	else:
	print("Query failed!")
	else:
	print("Failed to create connection!")

2.3.6 GPI Status (read-only)

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RO_ReaderGPIState Val: Return object ReaderGPIState_Model
Return	EReaderResult Result enumeration ReaderGPIState_Model.dicState data is a key-value pair of GPI enumerations and states

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	readerGPIState_Model = ReaderGPIState_Model()
	readerResult = reader.paramGet(EReaderEnum.RO_ReaderGPIState, readerGPIState_Model)
	if readerResult == EReaderResult.RT_OK:
	Reader.print_object(readerGPIState_Model)
	else:
	print("Query failed!")
	else:
	print("Failed to create connection!")

2.3.7 Serial Port Params

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderSerialPortParam Val: Return object ReaderSerial_Model
Return	EReaderResult Result enumeration ReaderSerial_Mode property is: Baudrate: Serial port baud rate enumeration
Example code	reader = Reader() log = Text()

	if reader.initReader("TCP:192.168.1.116:9090", log):
	readerSerial = ReaderSerial_Model()
	readerResult = reader.paramGet(EReaderEnum.RW_ReaderSerialPortParam, readerSerial)
	if readerResult == EReaderResult.RT_OK:
	Reader.print_object(readerSerial)
	else:
	print("Query failed!")
	else:
	print("Failed to create connection!")

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderSerialPortParam Val: Configuration object ReaderSerial_Model
Return	EReaderResult Result enumeration ReaderSerial_Model property is: Baudrate: Serial port baud rate

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	readerSerial = ReaderSerial_Model(EBaudrate._115200bps)
	readerResult = reader.paramSet(EReaderEnum.RW_ReaderSerialPortParam, readerSerial)
	if readerResult == EReaderResult.RT_OK:
	print("Set successfully!")
	else:
	print("Set failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

2.3.8 RS485 Params

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_Reader485Param Val: Return object ReaderSerial_Model
Return	EReaderResult Result enumeration ReaderSerial_Model properties are: Address: 485 address Baudrate: Serial port baud rate

Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerSerial = ReaderSerial_Model() readerResult = reader.paramGet(EReaderEnum.RW_Reader485Param, readerSerial) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(readerSerial) **else**: print(**"Query failed!""**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_Reader485Param Val: Configuration object ReaderSerial_Model ReaderSerial_Model properties are: Address: 485 address Baudrate: Serial port baud rate
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader(**"TCP:192.168.1.116:9090"**, log):
	# Set the serial address to 1 and the baud rate to 115200bps.
	readerSerial = ReaderSerial_Model(1, EBaudrate._115200bps)
	readerResult = reader.paramSet(EReaderEnum.RW_Reader485Param, readerSerial)
	if readerResult == EReaderResult.RT_OK:
	print(**"Set successfully!"**)
	else:
	print(**"Set failed!"**)
	else:
	print(**"Failed to create connection!"**)
	reader.closeConnect()

2.3.9 Network Configuration (MAC, Ipv4, Ipv6)

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum. RW_ReaderNetwork

	Val: Return object ReaderNetwork_Model
Return	EReaderResult Result enumeration ReaderNetwork_Model network object properties are: Mac: reader MAC address Ipv4Address: reader Ipv4 address Ipv4Mask: reader Ipv4 mask Ipv4GateWay: reader Ipv4 gateway Ipv4Dns: reader Ipv4 DNS DhcpSwitch: DHCP switch, True is on Ipv6Switch: Ipv6 switch Ipv6Address: reader Ipv6 address Ipv6Mask: reader Ipv6 mask Ipv6GateWay: reader Ipv6 gateway Ipv6Dns: reader Ipv6 DNS
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerNetwork = ReaderNetwork_Model() readerResult = reader.paramGet(EReaderEnum.RW_ReaderNetwork, readerNetwork) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(readerNetwork) **else**: print(**"Query failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderNetwork Val: Configuration object ReaderNetwork_Model
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader(**"TCP:192.168.1.116:9090"**, log):
	# Set IP to 192.168.1.116, subnet mask to 255.255.255.0 and gateway to 192.168.1.1.
	readerNetwork = ReaderNetwork_Model(**"192.168.1.116"**, **"255.255.255.0"**, **"192.168.1.1"**, **""**)
	readerResult = reader.paramSet(EReaderEnum.RW_ReaderNetwork, readerNetWork)
	if readerResult == EReaderResult.RT_OK:
	print(**"Set successfully!"**)

	else: print("Set failed!") else: print("Failed to create connection!") reader.closeConnect()

2.3.10 Reader Time (UTC, NTP)

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderTime Val: Return object ReaderTime_Model
Return	EReaderResult Result enumeration ReaderTime_Model object properties are: UTC: UTC Date and time, for example: "2023.10.10 10:00:00" NTP_Switch: NTP switch, True is on IP: NTP server IP address

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	readerTime = ReaderTime_Model()
	readerResult = reader.paramGet(EReaderEnum.RW_ReaderTime, readerTime)
	if readerResult == EReaderResult.RT_OK:
	Reader.print_object(readerTime)
	else:
	print("Query failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderTime Val: Configuration object ReaderTime_Model The object parameters UTC and NTP are independent of each other, meaning that only UTC or NTP can be configured. UTC: UTC Date and time, for example: "2023.10.10 10:00:00" NTP_Switch: NTP switch, True is on IP: NTP server IP address
Return	EReaderResult Result enumeration
Example code	reader = Reader()

	log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): *# set time only* readerTime = ReaderTime_Model(**"2022.10.10 10:00:00"**) *# set NTP only* *# readerTime = ReaderTime_Model(True,"192.168.1.11")* *# set both time and NTP* *# readerTime = ReaderTime_Model("2022.10.10 10:00:00",True,"192.168.1.11")* readerResult = reader.paramSet(EReaderEnum.RW_ReaderTime, readerTime) **if** readerResult == EReaderResult.RT_OK: print(**"Set successfully!"**) **else**: print(**"Set failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

2.3.11 Server/client Mode Parameters

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderWorkMode Val: Return object ReaderWorkMode_Model
Return	EReaderResult Result enumeration ReaderWorkMode_Model object properties are: workMode: TCP working mode, EWorkMode enumeration（Server,Client） Ip: If the Client mode is selected, you need to configure an IP address. The IP address is the IP address of the Server to which the reader actively connects in Client mode. Port: Specified port
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerWorkMode = ReaderWorkMode_Model() readerResult = reader.paramGet(EReaderEnum.RW_ReaderWorkMode, readerWorkMode) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(readerWorkMode) **else**: print(**"Query failed!"**)

	else: print("Failed to create connection!") reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderWorkMode Val: Configuration object ReaderWorkMode_Model If the working mode is set to Server, you only need to configure port. If Client is specified, you need to configure the IP address and port. workMode: TCP working mode(Server,Client) Ip: If the Client mode is selected, you need to configure an IP address. The IP address is the IP address of the Server to which the reader actively connects in Client mode. Port: Specified port
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	# Set as TCP server, port 9090
	readerWorkMode = ReaderWorkMode_Model(9090)
	# Set as TCP client, IP is 192.168.1.11, port is 9090
	# readerWorkMode = ReaderWorkMode_Model("192.168.1.11", 9090)
	readerResult = reader.paramSet(EReaderEnum.RW_ReaderWorkMode, readerWorkMode)
	if readerResult == EReaderResult.RT_OK:
	print("Set successfully!")
	else:
	print("Set failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

2.3.12 Buzzer (control and switch)

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderBuzzerSwitch Val: Return object ReaderBuzzer_Model
Return	EReaderResult Result enumeration ReaderBuzzer_Model properties are: buzzerControl: EBuzzerControl enumeration (Reader control: ReaderControl, that

	is, after reading a tag, the built-in buzzer of the reader will sound; upper computer control: PCControl, that is, through buzzerSwitch and buzzerType to control whether the buzzer sounds.) buzzerSwitch: True is on, false is off. Applicable when buzzerControl is PCControl buzzerType: EBuzzerType enumeration (Once: ring once, Always: ring all the time)
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): buzzer = ReaderBuzzer_Model() readerResult = reader.paramGet(EReaderEnum.RW_ReaderBuzzerSwitch, buzzer) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(buzzer) **else**: print(**"Query failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderBuzzerSwitch Val: Configuration object ReaderBuzzer_Model properties are: buzzerControl: EBuzzerControl enumeration (Reader control: ReaderControl, that is, after reading a tag, the built-in buzzer of the reader will sound; upper computer control: PCControl, that is, through buzzerSwitch and buzzerType to control whether the buzzer sounds.) buzzerSwitch: True is on, false is off. Applicable when buzzerControl is PCControl buzzerType: EBuzzerType enumeration (Once: ring once, Always: ring all the time) ReaderBuzzer_Model contains simple construction methods suitable for computer to control buzzer: buzzerOnlyOne (ring once), buzzerAlways (always ring), buzzerStop (stop)
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): buzzer = ReaderBuzzer_Model() # *Simple call, ring once* buzzer.buzzerOnlyOne() # *Ring all the time*

```
    # buzzer.buzzerAlways()
    #     Stop ringing
    # buzzer.buzzerStop()
    #     Turn on the tag reading prompt sound, that is, the buzzer will sound once when a tag is read
    # buzzer = ReaderBuzzer_Model(EBuzzerControl.ReaderControl,None,None)

    readerResult = reader.paramSet(EReaderEnum.RW_ReaderBuzzerSwitch, buzzer)
    if readerResult == EReaderResult.RT_OK:
        print("Set successfully!")
    else:
        print("Set failed!")
else:
    print("Failed to create connection!")
reader.closeConnect()
```

2.3.13 LED Status Indicator

It is only suitable for the model with hidden light belt in the integrated reader, which is used for tag reading prompt.

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderStateLED Val: Return object ReaderLED_Model
Return	EReaderResult Result enumeration ReaderLED_Model properties are: LEDState: Integrated reader hidden light strip switch. True is on LEDTime: Turn on time after reading a tag, in milliseconds.

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	  readerLED = ReaderLED_Model()
	  readerResult = reader.paramGet(EReaderEnum.RW_ReaderStateLED, readerLED)
	  if readerResult == EReaderResult.RT_OK:
	  Reader.print_object(readerLED)
	  else:
	  print("Set successfully!")
	else:
	  print("Failed to create connection!")

	reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderStateLED Val: Configuration object ReaderLED_Mode properties are: LEDState: Integrated reader hidden light strip switch. True is on LEDTime: Turn on time after reading a tag, in milliseconds.
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): # *Set LED strip to light up for 1 second after reading a tag.* readerLED = ReaderLED_Model(**True**,1000) readerResult = reader.paramSet(EReaderEnum.RW_ReaderStateLED, readerLED) **if** readerResult == EReaderResult.RT_OK: print(**"Set successfully!"**) **else**: print(**"Set failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

2.3.14 Custom Tag Output Format

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderDataOutputFormat Val: Return object ReaderDataOutput_Model
Return	EReaderResult Result enumeration ReaderDataOutput_Model properties are: outputSwitch: Special format switch, EOutputSwitch enumeration (Close, Open, UDPOutput) **outputFormat: Output format, EOutputFormat Enumeration (Hex, ASCII, Decimal)** **tagData:** Tag upload class, configure the matching area, start byte, and character length. (Matching area: EPC, TID) **startData: Start of Text** **endData: End of Text**
Example code	reader = Reader()

	log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerDataOutput = ReaderDataOutput_Model() readerResult = reader.paramGet(EReaderEnum.RW_ReaderDataOutputFormat, readerDataOutput) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(readerDataOutput) **else**: print(**"Query failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	ReaderResult Result enumeration ReaderDataOutput_Model properties are: outputSwitch: Special format switch, EOutputSwitch enumeration (Close, Open, UDPOutput) **outputFormat: Output format, EOutputFormat Enumeration (Hex, ASCII, Decimal)** **tagData:** Tag upload class, configure the matching area, start byte, and character length. (Matching area: EPC, TID) **startData: Start of Text** **endData: End of Text**
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerDataOutput = ReaderDataOutput_Model() readerDataOutput.outputSwitch = EOutputSwitch.UDPOutput readerDataOutput.outputFormat = EOutputFormat.ASCII tagDataModel = TagData_Model(EReadBank.EPC) readerDataOutput.tagData = tagDataModel readerDataOutput.startData = **"40"** readerDataOutput.endData = **"25"** readerResult = reader.paramSet(EReaderEnum.RW_ReaderDataOutputFormat, readerDataOutput) **if** readerResult == EReaderResult.RT_OK: print(**"Set successfully!"**) **else**:

	print("Set failed!") else: print("Failed to create connection!") reader.closeConnect()

2.3.15 Custom ID

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderCustomCode Val: Return object ReaderString_Model
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	readerString = ReaderString_Model()
	readerResult = reader.paramGet(EReaderEnum.RW_ReaderCustomCode, readerString)
	if readerResult == EReaderResult.RT_OK:
	Reader.print_object(readerString)
	else:
	print("Query failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderCustomCode Val: Configuration object ReaderString_Model
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	readerString = ReaderString_Model("1")
	readerResult = reader.paramSet(EReaderEnum.RW_ReaderCustomCode, readerString)
	if readerResult == EReaderResult.RT_OK:
	print("Set successfully!")
	else:
	print("Set failed!")
	else:

	print("Failed to create connection!") reader.closeConnect()

2.3.16 Antenna Power (power, enable)

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDAntPower Val: Return object List<ReaderAntPower>
Return	EReaderResult Result enumeration ReaderAntPower_Model properties are: antennaNo: Antenna No. power: Power in dBm enable: Enable

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	readerAntPowerList = []
	readerResult = reader.paramGet(EReaderEnum.RW_RFIDAntPower, readerAntPowerList)
	if readerResult == EReaderResult.RT_OK:
	for item in readerAntPowerList:
	Reader.print_object(item)
	else:
	print("Query failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDAntPower Val: Configuration object List<ReaderAntPower_Model>, ReaderAntPower_Model properties are: antennaNo: Antenna no. enumeration power: Power in dBm enable: Enable
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() if reader.initReader("TCP:192.168.1.116:9090", log): *# Set the output power of antennas 1,2,3,4 to 30, and enable antennas 1, 2, 3*

	and 4 at the same time. readerAntPowerList = ReaderAntPower_Model.AntPowerAllConfig(30, **True**, 1, 2, 3, 4) *# Only set the power of antenna 1 to 20, and do not set enable* *readerAntPower1 = ReaderAntPower_Model(EAntennaNo._1,20,None)* *readerAntPowerList = []* *readerAntPowerList.append(readerAntPower1)* readerResult = reader.paramSet(EReaderEnum.RW_RFIDAntPower, readerAntPowerList) **if** readerResult == EReaderResult.RT_OK: print(**"Set successfully!"**) **else**: print(**"Set failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

2.3.17 RF Configuration (frequency band, frequency point)

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDRF Val: Return object ReaderRF_Model
Return	EReaderResult Result enumeration ReaderRF_Model properties are: readerWorkFrequency: Working frequency band RFHoppingMode: Frequency hopping mode, EWF_Mode enumeration(Specified, Auto) readerWorkPoint: Specify the frequency points, such as "0", "3"
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerRF_model = ReaderRF_Model() readerResult = reader.paramGet(EReaderEnum.RW_RFIDRF, readerRF_model) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(readerRF_model) **else**: print(**"Query failed!"**) **else**:

	print("Failed to create connection!") reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDRF Val: Configuration object ReaderRF_Model properties are: The readerWorkFrequency and (RFHoppingMode, readerWorkPoint) functions are relatively independent. readerWorkFrequency: Working frequency band RFHoppingMode: Frequency hopping mode, EWF_Mode enumeration(Specified, Auto) readerWorkPoint: Specify the frequency points, such as "0", "3"
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	readerRF_model = ReaderRF_Model()
	readerRF_model.readerWorkFrequency = ERF_Range.FCC_902_to_928MHz
	readerRF_model.RFHoppingMode = EWF_Mode.Specified
	rfList = ReaderRF_Model.WorkPointConfig(1, 2, 3, 4)
	readerRF_model.readerWorkPoint = rfList
	readerResult = reader.paramSet(EReaderEnum.RW_RFIDRF, readerRF_model)
	if readerResult == EReaderResult.RT_OK:
	print("Set successfully!")
	else:
	print("Set failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

2.3.18 EPC Baseband Params

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDEpcBasebandParam Val: Return object EpcBaseband_Model properties are: eBasebandRate: EPC baseband rate qValue: The starting Q value used by the reader. Session: session value

	searchType: Inventory flag parameters (FlagA only, FlagB only, FlagA and FlagB double-sided inventory)
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): epcBaseband = EpcBaseband_Model() readerResult = reader.paramGet(EReaderEnum.RW_RFIDEpcBasebandParam, epcBaseband) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(epcBaseband) **else**: print(**"Query failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDEpcBasebandParam Val: Configuration object EpcBaseband_Model properties are: eBasebandRate: EPC baseband rate(0, Tari=25us, FM0, BLF=40KHz 1, Tari=25us, Miller4, BLF=250KHz(Dense) 2, Tari=25us, Miller4, BLF=300KHz 3, Tari=6.25us, FM0, BLF=400KHz(fast)) qValue: The starting Q value used by the reader. Session: session value searchType: Inventory flag parameters (FlagA only, FlagB only, FlagA and FlagB double-sided inventory)
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): epcBaseband = EpcBaseband_Model() epcBaseband.eBasebandRate = EBasebandRate.Tari_25us_Miller4_BLF_250KHz epcBaseband.qValue = 4 epcBaseband.session = 1 epcBaseband.searchType = ESearchType.FlagA readerResult = reader.paramSet(EReaderEnum.RW_RFIDEpcBasebandParam, epcBaseband) **if** readerResult == EReaderResult.RT_OK:

	print("Set successfully!") else: print("Set failed!") else: print("Failed to create connection!") reader.closeConnect()

2.3.19 EPC Baseband Extended Params

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDEpcBaseExpandBandParam Val: Return object EPCExtendedParam_Model properties are: TAG extended parameter object: TagExtendedParam properties are: IMJ_Tag_Focus: bool type, IMJ_Tag_Focus function switch IMJ_Fast_Id: bool type, IMJ_Fast_Id function switch NXP_Fast_ID: bool type, NXP_Fast_ID function switch DNQ extended parameter object: DNQExtendedParam properties are: MaxQ: int type, decimal MinQ: int type, decimal Tmult: int type, decimal AutoQ: bool type, Dynamic start Q function switch ForceQ: bool type, Forced loop algorithm function switch AST extended parameter object: ASTExtendedParam properties are: AntSwitchMode: antenna working switching mode, EASTSwitchMode enumeration (Switch_Immediately_Without_Tags, Running_Out_Of_Residence_Time). Retry:int type, decimal, number of retries (number of retries identified without tags, a reference option for antenna switching) ResidenceTime: int, decimal, maximum antenna residence time (x10ms) AST2 extended parameter object: ASTExtendedParam2 properties are: WaitingTime:int type, decimal, antenna switching wait time (x10ms) AntStep:int type, decimal, antenna switching step value AntThreshold: int, decimal, antenna protection threshold (return loss dBm). Set it to 0 to disable protection. LBT extended parameter object:LBTExtendedParam properties are: WorkMode: working mode, ELBTWorkMode enumeration (Disable,

	LBT_Listening_Only, Read_Tag_After_Listening, Read_Tag_After_Meeting_RSSI) MaxRSSI: RSSI maximum
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): extendParam = EPCExtendedParam_Model() readerResult = reader.paramGet(EReaderEnum.RW_RFIDEpcBaseExpandBandParam, extendParam) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(extendParam.TagExtendedParam) Reader.print_object(extendParam.DNQExtendedParam) Reader.print_object(extendParam.ASTExtendedParam) Reader.print_object(extendParam.ASTExtendedParam2) Reader.print_object(extendParam.LBTExtendedParam) **else**: print(**"Query failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDEpcBaseExpandBandParam Val: Configuration object EPCExtendedParam_Model properties are: TAG extended parameter object: TagExtendedParam properties are: IMJ_Tag_Focus: bool type, IMJ_Tag_Focus function switch IMJ_Fast_Id: bool type, IMJ_Fast_Id function switch NXP_Fast_ID: bool type, NXP_Fast_ID function switch DNQ extended parameter object: DNQExtendedParam properties are: MaxQ: int type, decimal MinQ: int type, decimal Tmult: int type, decimal AutoQ: bool type, Dynamic start Q function switch ForceQ: bool type, Forced loop algorithm function switch AST extended parameter object: ASTExtendedParam properties are: AntSwitchMode: antenna working switching mode, EASTSwitchMode enumeration (Switch_Immediately_Without_Tags, Running_Out_Of_Residence_Time).

<table>
<tr><td></td><td>Retry:int type, decimal, number of retries (number of retries identified without tags, a reference option for antenna switching)
ResidenceTime: int, decimal, maximum antenna residence time (x10ms)

AST2 extended parameter object: ASTExtendedParam2 properties are:
WaitingTime:int type, decimal, antenna switching wait time (x10ms)
AntStep:int type, decimal, antenna switching step value
AntThreshold: int, decimal, antenna protection threshold (return loss dBm). Set it to 0 to disable protection.

LBT extended parameter object:LBTExtendedParam properties are:
WorkMode: working mode, ELBTWorkMode enumeration (Disable, LBT_Listening_Only, Read_Tag_After_Listening, Read_Tag_After_Meeting_RSSI)
MaxRSSI: RSSI maximum</td></tr>
<tr><td>Return</td><td>EReaderResult Result enumeration。</td></tr>
<tr><td>Example code</td><td>reader = Reader()
log = Text()
if reader.initReader("TCP:192.168.1.116:9090", log):
extendParam = EPCExtendedParam_Model()
set TAG extended parameters
TagExtendedParam = TagExtendedParam_Model()
TagExtendedParam.IMJ_Fast_Id = True
TagExtendedParam.IMJ_Tag_Focus = True
TagExtendedParam.NXP_Fast_ID = True
extendParam.TagExtendedParam = TagExtendedParam
set DNQ extended parameters
DNQExtendedParam = DNQExtendedParam_Model()
DNQExtendedParam.maxQ = 10
DNQExtendedParam.minQ = 0
DNQExtendedParam.tmult = 4
DNQExtendedParam.autoQ = False
DNQExtendedParam.forceQ = False
extendParam.DNQExtendedParam = DNQExtendedParam
ser AST extended parameters
ASTExtendedParam = ASTExtendedParam_Model()
ASTExtendedParam.antSwitchMode = EASTSwitchMode.Running_Out_Of_Residence_Time
ASTExtendedParam.retry = 5
ASTExtendedParam.residenceTime = 10
extendParam.ASTExtendedParam = ASTExtendedParam
set AST2 extended parameters
ASTExtendedParam2 = ASTExtendedParam2_Model()
ASTExtendedParam2.waitingTime = 11</td></tr>
</table>

```
        ASTExtendedParam2.antStep = 0
        ASTExtendedParam2.antThreshold = 5
        extendParam.ASTExtendedParam2 = ASTExtendedParam2
        # set LBT extended parameters
        LBTExtendedParam = LBTExtendedParam_Model()
        LBTExtendedParam.workMode = ELBTWorkMode.LBT_Listening_Only
        LBTExtendedParam.maxRSSI = 0
        extendParam.LBTExtendedParam = LBTExtendedParam
        readerResult =
reader.paramSet(EReaderEnum.RW_RFIDEpcBaseExpandBandParam,
extendParam)
        if readerResult == EReaderResult.RT_OK:
            print("Set successfully!")
        else:
            print("Set failed!")
else:
        print("Failed to create connection!")
reader.closeConnect()
```

2.3.20 Tag Upload Params

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDTagUpdateParam Val: Return object ReaderTagUpdate_Model
Return	EReaderResult Result enumeration ReaderTagUpdate_Model properties are: repeatTimeFilter: Duplicate tag upload filter time rssiFilter: RSSI filter dBmFilter: RSSI_dBm Threshold (optional parameters, not supported by some devices)

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	  readerTagUpdate = ReaderTagUpdate_Model()
	  readerResult = reader.paramGet(EReaderEnum.RW_RFIDTagUpdateParam, readerTagUpdate)
	  if readerResult == EReaderResult.RT_OK:
	  Reader.print_object(readerTagUpdate)
	  else:
	  print("Query failed!")
	else:

	print("Failed to create connection!") reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDTagUpdateParam Val: Configuration object ReaderTagUpdate_Model properties are: repeatTimeFilter: Duplicate tag upload filter time rssiFilter: RSSI filter dBmFilter: RSSI_dBm Threshold (optional parameters, not supported by some devices)
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	# Set the filtering time for uploading duplicate tags to 10 * 10ms and the RSSI filtering threshold to 0.
	readerTagUpdate = ReaderTagUpdate_Model(10, 0)
	readerResult = reader.paramSet(EReaderEnum.RW_RFIDTagUpdateParam, readerTagUpdate)
	if readerResult == EReaderResult.RT_OK:
	print("Set successfully!")
	else:
	print("Set failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

2.3.21 Reader Auto Idle Mode

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDAutoIdleParam Val: Return object ReaderAutoSleep_Model
Return	EReaderResult Result enumeration ReaderAutoSleep_Model properties are: autoIdleSwitch: Automatic idle mode switch Time: Automatic idle time, Unit 10ms
Example code	reader = Reader() log = Text() if reader.initReader("TCP:192.168.1.116:9090", log): readerAutoSleep = ReaderAutoSleep_Model()

	readerResult = reader.paramGet(EReaderEnum.RW_RFIDAutoIdleParam, readerAutoSleep)
	if readerResult == EReaderResult.RT_OK:
	Reader.print_object(readerAutoSleep)
	else:
	print("Query failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_RFIDAutoIdleParam Val: Configuration object ReaderAutoSleep_Model properties are: autoIdleSwitch: Automatic idle mode switch Time: Automatic idle time, Unit 10ms
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	# Set the automatic idle time to 100ms.
	readerAutoSleep = ReaderAutoSleep_Model(True, 10)
	readerResult = reader.paramSet(EReaderEnum.RW_RFIDAutoIdleParam,readerAutoSleep)
	if readerResult == EReaderResult.RT_OK:
	print("Set successfully!")
	else:
	print("Set failed!")
	else:
	print("Failed to create connection!")
	reader.closeConnect()

2.3.22 GPI Params

Get

Namespace	com.rfid.Reader
Function	**EReaderResult paramGet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.RW_ReaderGPIParam Val: Return object ReaderGPIParam_Model
Return	EReaderResult Result enumeration ReaderGPIParam_Model properties are: GPINum: GPI port number triggerStart: Trigger start condition, eTriggerStart

	enumeration(OFF,Low_level,High_level,Rising_edge,Falling_edge,Any_edge) triggerCode: Trigger execution command, eTriggerCode enumeration(Single_Antenna_read_EPC Single_Antenna_read_EPC_and_TID Double_Antenna_read_EPC Double_Antenna_read_EPC_and_TID Four_Antenna_read_EPC Four_Antenna_read_EPC_and_TID Eight_Antenna_read_EPC Eight_Antenna_read_EPC_and_TID Twelve_Antenna_read_EPC Twelve_Antenna_read_EPC_and_TID Twenty_four_Antenna_read_EPC Twenty_four_Antenna_read_EPC_and_TID Custom_Read_0 Custom_Read_1 Custom_Read_2) triggeerStop: Trigger stop condition, eTriggerStop, enumeration(OFF,Low_level,High_level,Rising_edge,Falling_edge,Any_edge,Delay) delayTime: Stop delay time isUpload: Upload Flag customTriggerCode: Custom command
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): readerGPIParam = ReaderGPIParam_Model(EGPIO._1) readerResult = reader.paramGet(EReaderEnum.RW_ReaderGPIParam, readerGPIParam) **if** readerResult == EReaderResult.RT_OK: Reader.print_object(readerGPIParam) **else**: print(**"Query failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum. RW_ReaderGPIParam Val: Configuration object ReaderGPIParam_Model properties are: GPINum: GPI port number triggerStart: Trigger start condition, eTriggerStart

<table>
<tr><td></td><td>enumeration(OFF,Low_level,High_level,Rising_edge,Falling_edge,Any_edge)
triggerCode: Trigger execution command, eTriggerCode enumeration(
Single_Antenna_read_EPC
Single_Antenna_read_EPC_and_TID
Double_Antenna_read_EPC
Double_Antenna_read_EPC_and_TID
Four_Antenna_read_EPC
Four_Antenna_read_EPC_and_TID
Eight_Antenna_read_EPC
Eight_Antenna_read_EPC_and_TID
Twelve_Antenna_read_EPC
Twelve_Antenna_read_EPC_and_TID
Twenty_four_Antenna_read_EPC
Twenty_four_Antenna_read_EPC_and_TID
Custom_Read_0
Custom_Read_1
Custom_Read_2
)
triggeerStop: Trigger stop condition, eTriggerStop,
enumeration(OFF,Low_level,High_level,Rising_edge,Falling_edge,Any_edge,Delay)
delayTime: Stop delay time
isUpload: Upload Flag
customTriggerCode: Custom command</td></tr>
<tr><td>Return</td><td>EReaderResult Result enumeration</td></tr>
<tr><td>Example code</td><td>reader = Reader()
log = Text()
if reader.initReader("TCP:192.168.1.116:9090", log):
Set GPI1, trigger reading on high level, stop reading on low level, and use custom code to read EPC+TID+User for tags
readerGPIParam = ReaderGPIParam_Model()
readerGPIParam.GPINum = EGPIO._1
readerGPIParam.triggerStart = ETriggerStart.High_level
readerGPIParam.triggerCode = ETriggerCode.Custom_Read_0
readerGPIParam.triggeerStop = ETriggerStop.Low_level
readerGPIParam.delayTime = 10
readerGPIParam.isUpload = EGPIUpload.Upload_Start_Trigger_Msg
readerGPIParam.customTriggerCode = "021000090101030000020020006"
readerResult = EReaderEnum.RW_ReaderGPIParam,readerGPIParam))
if readerResult == EReaderResult.RT_OK:
print("Set successfully!")
else:
print("Set failed!")
else:</td></tr>
</table>

	print("Failed to create connection!") reader.closeConnect()

3.3.23 Set GPO Status

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum. WO_SetGPOState Val: Configuration object ReaderGPOState_Model properties are: ReaderGPOState_Model. dicState is Key-value pairs for GPIO enumerations and states
Return	EReaderResult Result enumeration

Example code	reader = Reader()
	log = Text()
	if reader.initReader("TCP:192.168.1.116:9090", log):
	    # Set GPO1 to high level, set GPO2 to low level
	    dic = {EGPIO._1:EGPOState._High,EGPIO._2:EGPOState._Low}
	    setGPOdic = ReaderGPOState_Model(dic)
	    readerResult = reader.paramSet(EReaderEnum.WO_SetGPOState, setGPOdic)
	
	    if readerResult == EReaderResult.RT_OK:
	        print("Set successfully!")
	    else:
	        print("Set failed!")
	else:
	    print("Failed to create connection!")
	reader.closeConnect()

2.3.24 Reader Working Antenna Configuration

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum. WO_RFIDWorkingAnt Val: Configuration object ReaderWorkingAntSet_Model, properties are: antennaList: Integer array. An array of antennas used to configure the reader to read tags, which can be one antenna or multiple antennas.
Return	EReaderResult Result enumeration
Example code	reader = Reader()

	log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): *# Set working antennas 1,2,3.* readerAntPlan = ReaderWorkingAntSet_Model([1, 2, 3]) readerResult = reader.paramSet(EReaderEnum.WO_RFIDWorkingAnt, readerAntPlan) **if** readerResult == EReaderResult.RT_OK: print(**"Set successfully!"**) **else**: print(**"Set failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

2.3.25 Write (Read Extended Area)

Note: After configuring to read the extended area, the extended area will be read afterwards, if you want to cancel reading the extended area, you need to reconfigure it.

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.WO_RFIDReadExtended Val:List<ReadExtendedArea_Model> ReadExtendedArea_Model object properties are: Bank: The area to be read during inventory. readStart: Start address in bank. Unit is word. readLen: The number of words to read. Unit is word. passWord: Access password
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): *# set to read TID,UserData and Reserved data at the same time* readTID = ReadExtendedArea_Model(EReadBank.TID, 0, 6, **""**) readUserData = ReadExtendedArea_Model(EReadBank.UserData, 0, 6, **"00000000"**) readReserved = ReadExtendedArea_Model(EReadBank.Reserved, 0, 4, **""**) readExtendedAreaList = [] readExtendedAreaList.append(readTID) readExtendedAreaList.append(readUserData)

	readExtendedAreaList.append(readReserved) readerResult = reader.paramSet(EReaderEnum.WO_RFIDReadExtended, readExtendedAreaList) **if** readerResult == EReaderResult.RT_OK: print(**"Set successfully!"**) **else**: print(**"Set failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

2.3.26 Write (Read Special Area)

Note: After configuring to read the extended area, the extended area will be read afterwards, if you want to cancel reading the extended area, you need to reconfigure it.

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.WO_RFIDReadSpecial Val: List<EReadSpecial> Enumerates the special areas to be read contained in EreadSpecial.
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): *# set to read special area -- NXP brand ID* eReadSpecialList = [] eReadSpecialList.append(EReadSpecial.NXP_BrandID) readerResult = reader.paramSet(EReaderEnum.WO_RFIDReadSpecial, eReadSpecialList) **if** readerResult == EReaderResult.RT_OK: print(**"Set successfully!"**) **else**: print(**"Set failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

2.3.27 Write (Read Filter Rules)

Note: After configuring to read the extended area, the extended area will be read afterwards, if you want to cancel reading the extended area, you need to reconfigure it.

Set

Namespace	com.rfid.Reader
Function	**EReaderResult paramSet(EReaderEnum key, Object val)**
Params	Key: Reader info enumeration: EReaderEnum.WO_RFIDReadTagFilter Val: Configuration object List<TagFilter_Model>, TagFilter_Model properties are: Bank: The area to be read during inventory(EPC,TID,UserData). tagStart: Start address in bank. EPC starts at 32, other starts at 0, unit is bit. Data: tag data need to be matched. Note: You can set up to three filter rules. All readers support filter rule 01, but may not support filter rule 02 and filter rule 03. The filter rule 01 and the filter rule 02 are in a logical AND relationship, and the filter rule 01/02 and the filter rule 03 are in a logical OR relationship.
Return	EReaderResult Result enumeration
Example code	reader = Reader() log = Text() **if** reader.initReader(**"TCP:192.168.1.116:9090"**, log): *# Read tags while satisfying the EPC start at 22 and the TID start at E28011052000* tagFilter1 = TagFilter_Model(EReadBank.TID, 0, **" E28011052000"**) tagFilter2 = TagFilter_Model(EReadBank.EPC, 32, **"22"**) tagFilterList = [] tagFilterList.append(tagFilter1) tagFilterList.append(tagFilter2) readerResult = reader.paramSet(EReaderEnum.WO_RFIDReadTagFilter, tagFilterList) **if** readerResult == EReaderResult.RT_OK: print(**"Set successfully!"**) **else**: print(**"Set failed!"**) **else**: print(**"Failed to create connection!"**) reader.closeConnect()

3. Programming Example

Requirement: Use antennas 1, 2, and 3 to read the tag with TID E280110520058CDC92D02C1, while also reading TID and UserData for 2 seconds.

Synchronous way to complete:

```
class Text():
    def main(self):
        reader = Reader()
        if reader.initReader("TCP:192.168.1.116:9090"):
            #  Set the working antennas. If not, antenna 1 will be used by default.
            readerAntPlan = ReaderWorkingAntSet_Model([1, 2, 3])
            print(reader.paramSet(EReaderEnum.WO_RFIDWorkingAnt,
readerAntPlan))
            #  Set the extended read TID and User areas. If not, only EPC will be read.
            readTID = ReadExtendedArea_Model(EReadBank.TID, 0, 6, "")
            readUserData = ReadExtendedArea_Model(EReadBank.UserData, 0, 6,
"00000000")
            readExtendedAreaList = []
            readExtendedAreaList.append(readTID)
            readExtendedAreaList.append(readUserData)
            print(reader.paramSet(EReaderEnum.WO_RFIDReadExtended,
readExtendedAreaList))
            #  Set the filter to read only the tag with TID of
E2801105200058CDC92D02C1. If it is not set, all tags nearby will be read.
            tagFilter1 = TagFilter_Model(EReadBank.TID, 0,
"E2801105200058CDC92D02C1")
            tagFilterList = []
            tagFilterList.append(tagFilter1)
            print(reader.paramSet(EReaderEnum.WO_RFIDReadTagFilter,
tagFilterList))
            #  Start synchronous reading
            readList = []
            reader.read(2000,readList)
            for tag in readList:
                print("EPC:" + tag._EPC + ",TID:" + tag._TID    + ",_UserData:" +
tag._UserData)
            #  Stop reading and close the connection.
            reader.stop()
            reader.closeConnect()

        else:
```

```
            print("Failed to create connection!")
        reader.closeConnect()

if __name__ == '__main__':
    s = Text()
    s.main()
```

Asynchronous way to complete:

```
class Text(IAsynchronousMessage):
    def main(self):
        reader = Reader()
        log = Text()
        if reader.initReader("TCP:192.168.1.116:9090", log):
            #  Set the working antenna. If not, antenna 1 will be used by default.
            readerAntPlan = ReaderWorkingAntSet_Model([1, 2, 3])
            print(reader.paramSet(EReaderEnum.WO_RFIDWorkingAnt,
readerAntPlan))
            #  Set the extended read TID and User areas. If not, only EPC will be read.
            readTID = ReadExtendedArea_Model(EReadBank.TID, 0, 6, "")
            readUserData = ReadExtendedArea_Model(EReadBank.UserData, 0, 6,
"00000000")
            readExtendedAreaList = []
            readExtendedAreaList.append(readTID)
            readExtendedAreaList.append(readUserData)
            print(reader.paramSet(EReaderEnum.WO_RFIDReadExtended,
readExtendedAreaList))
            #  Set the filter to read only the tag with TID of
E2801105200058CDC92D02C1. If it is not set, all tags nearby will be read.
            tagFilter1 = TagFilter_Model(EReadBank.TID, 0,
"E2801105200058CDC92D02C1")
            tagFilterList = []
            tagFilterList.append(tagFilter1)
            print(reader.paramSet(EReaderEnum.WO_RFIDReadTagFilter,
tagFilterList))
            #  Start asynchronous reading
            reader.inventory()
            time.sleep(2.0)
            #  Stop reading and close the connection.
            reader.stop()
            reader.closeConnect()

        else:
            print("Failed to create connection!")
```

```
            reader.closeConnect()


        def OutputTags(self, tag):
            try:
                print("EPC:" + tag._EPC + ",TID:" + tag._TID + ",_EpcData:" +
tag._EpcData + ",_UserData:" + tag._UserData + ",_TagetData:" + tag._TagetData)
            except Exception as e:
                print("Method OutputTags failed with error message %s" % e)

if __name__ == '__main__':
        s = Text()
        s.main()
```

4. FAQ and Solutions

Question	Solution
Device couldn't work normally	1. Check power light normal or not. 2. If normal, there should be some notice sound when power on.
serial port couldn't work normally	1. Check connection cable connecting normal or not . 2. If conditional, use another device to check this cable normal or not. 3. Try to use RJ45 to communicate. 4. Default baud rate: 115200.
RJ45 couldn't work normally	1. Check LED working normal or not. 2. To use Ping reader IP to check cable working normal or not 3. Try serial port connection, check if IP is correct by Demo 4. Default IP and port: "192.168.1.116:9090"

Appendix A: Enumeration of Return Results

EreaderResult mainly as follows:

RT_OK	Succeed
RT_FAILED_ERR	Failed
RT_SYSTEM_ERR	system error

RT_NOT_SUPPORTED_ERR	Not Supported
RT_INVALID_PARA_ERR	parameter error
RT_TIMEOUT_ERR	Reader response timeout
RT_NOT_CONNECT_ERR	Reader is not connected.
RT_NOT_FILTER_ERR	Filter error
RT_AREA_LOCKED_ERR	This area is locked.
RT_NOT_CONNECT_INFINITY_ERR	Antenna not connected, standing wave ratio: infinite.