DAILY RFID Long Range HF Reader DL810_5510MUL.DLL

Dynamic Link Library

User's Manual V6.2

DL810_5510MUL.DLL is a dynamic link library designed to facilitate ISO/IEC 15693 protocol HF tag application software development when using DL810 & DL5510 HF long range reader.

1) Operation System Requirement

WINDOWS 2000/XP

2) Function List

DL810_5510MUL.DLL includes the following functions for the ISO/IEC 15693 HF tag operation.

2.1) General Function

1)long WINAPI AutoOpenComPort(long *Port, unsigned char *ComAdr, long *FrmHandle);

2)long WINAPI OpenComPort(long Port, unsigned char *ComAdr, long *FrmHandle);

3)long WINAPI CloseComPort(void);

4)long WINAPI CloseSpecComPort(long FrmHandle);

5)long WINAPI GetReaderInformation(unsigned char *ComAdr, unsigned char *VersionInfo, unsigned char *ReaderType, unsigned char *TrType, unsigned char *InventoryScanTime, long FrmHandle);

6)long WINAPI OpenRf(unsigned char *comadr, long FrmHandle);

7)long WINAPI CloseRf(unsigned char *comadr, long FrmHandle);

8)long WINAPI WriteComAdr(unsigned char *ComAdr, unsigned char *ComAdrData, long FrmHandle);

9)long WINAPI WriteInventoryScanTime(unsigned char *ComAdr, unsigned char *InventoryScanTime, long FrmHandle);

10)long WINAPI SetGeneralOutput(unsigned char *ComAdr, unsigned char *OutputData, long FrmHandle);

11)long WINAPI GetGeneralInput(unsigned char *ComAdr, unsigned char *InputData, long FrmHandle);

12)long WINAPI SetRelay(unsigned char *ComAdr, unsigned char *RelayAction, long FrmHandle);

13)long WINAPI SetActiveANT(unsigned char *ComAdr, unsigned char *_ANT_Status, long FrmHandle);

14)long WINAPI GetANTStatus(unsigned char *ComAdr, unsigned char *Get_ANT_Status, long FrmHandle);

15)long WINAPI SetUserDefinedBlockLength(unsigned char *ComAdr, unsigned char *_Block_Len, long FrmHandle);

16)long WINAPI GetUserDefinedBlockLength(unsigned char *ComAdr, unsigned char *_Block_Len, long FrmHandle);

17)long WINAPI SetScanMode(unsigned char *ComAdr, unsigned char *_Scan_Mode_Data, long FrmHandle);

18)long WINAPI GetScanModeStatus(unsigned char *ComAdr, unsigned char *_Scan_Mode_Status, long FrmHandle);

19)long WINAPI ReadScanModeData(unsigned char *ScanModeData, long *ValidDataLength, long FrmHandle);

20)long WINAPI SetAccessTime(unsigned char *ComAdr, unsigned char *AccessTime, long FrmHandle);

21)long WINAPI GetAccessTime(unsigned char *ComAdr, unsigned char *AccessTimeRet, long FrmHandle);

22)long WINAPI SetReceiveChannel(unsigned char *ComAdr, unsigned char *ReceiveANT, long FrmHandle);

23)long WINAPI GetReceiveChannelStatus(unsigned char *ComAdr, unsigned char *ReceiveANTStatus, long FrmHandle);

24)long WINAPI SetParseMode(unsigned char *ComAdr, unsigned char *ParseMode, long FrmHandle);

25)long WINAPI GetParseMode(unsigned char *ComAdr, unsigned char *ParseMode, long FrmHandle);

2.2) Basic Operation Function

1)long WINAPI Inventory(unsigned char *ComAdr, unsigned char *State, unsigned char *AFI, unsigned char *DSFIDAndUID, unsigned char *CardNum, long FrmHandle);

2)long WINAPI StayQuiet(unsigned char *ComAdr, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

3)long WINAPI ReadSingleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *BlockSecStatus, unsigned char *Data, unsigned char *ErrorCode, long FrmHandle);

4)long WINAPI WriteSingleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *Data, unsigned char *ErrorCode, long FrmHandle);

5)long WINAPI LockBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *ErrorCode, long FrmHandle);

6)long WINAPI ReadMultipleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char BlockCount, unsigned char *BlockSecStatus, unsigned char *Data, unsigned char *ErrorCode, long FrmHandle);

7)long WINAPI Select(unsigned char *ComAdr, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

8)long WINAPI ResetToReady(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

9)long WINAPI WriteAFI(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char AFI, unsigned char *ErrorCode, long FrmHandle);

10)long WINAPI LockAFI(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

11)long WINAPI WriteDSFID(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char DSFID, unsigned char *ErrorCode, long FrmHandle);

12)long WINAPI LockDSFID(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

13)long WINAPI GetSystemInformation(unsigned char *ComAdr, unsigned char *State, unsigned char *UIDI, unsigned char *InformationFlag, unsigned char *UIDO, unsigned char *DSFID, unsigned char *AFI, unsigned char *MemorySize, unsigned char *ICReference, unsigned char *ErrorCode, long FrmHandle);

14)long WINAPI CustomizedReadSingleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *DataBuffer, unsigned char *ErrorCode, long FrmHandle);

15)long WINAPI CustomizedWriteSingleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *DataBuffer, unsigned char ByteCount, unsigned char *ErrorCode, long FrmHandle);

16)long WINAPI CustomizedReadMultipleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char BlockCount, unsigned char *DataBuffer, unsigned char *ErrorCode, long FrmHandle);

2.3) Transparent Command Function

1)long WINAPI TransparentRead(unsigned char *ComAdr, unsigned char RspLength, unsigned char CustomDataLength, unsigned char *CustomData, unsigned char *FeedbackDataLength, unsigned char *FeedbackData, unsigned char *ErrorCode, long FrmHandle);

2)long WINAPI TransparentWrite(unsigned char *ComAdr, unsigned char *Option, unsigned char RspLength, unsigned char CustomDataLength, unsigned char *CustomData, unsigned char *FeedbackDataLength, unsigned char *FeedbackData, unsigned char *ErrorCode, long FrmHandle);

3)long WINAPI TransparentCustomizedCmd(unsigned char *ComAdr, unsigned char *RspTime, unsigned char RspLength, unsigned char CustomDataLength, unsigned char *CustomData, unsigned char *FeedbackDataLength, unsigned char *FeedbackData, unsigned char *ErrorCode, long FrmHandle);

3) Function Explanation

3.1) General Function

3.1.1) AutoOpenComPort()

Function description:

This function is used to automatically detect the communication port unoccupied by other application and attached with a reader. The function try to establish the connection between them. The protocol parameters are 19200bps, 8 data bits, 1 start bit, 1 stop bit, no parity bit.

If the connection is established successfully, the function will open the communication port and return a valid handle, otherwise the function will return an error code with a invalid handle(value as -1).

Usage:

AutoOpenComPort(long * Port, unsigned char *ComAdr, long *FrmHandle);

Parameter:

Port: Pointed to the communication port number(COM1~COM12) that the reader is

detected and connected.

ComAdr: Pointed to the address of the reader.

When using broadcasting address 0xFF as ComAdr to call the function, the port number to which the reader is detected and the address of the reader will be writed back to parameter Port and ComAdr;

When using a designated address 0x00~0xFE as ComAdr to call the function, the port number to which the reader with the specified address is detected will be writed back to parameter Port.

Constants COM1~COM10 are defined as follows:

```
#define COM1    1
#define COM2    2
#define COM3    3
#define COM4    4
#define COM5    5
#define COM6    6
#define COM7    7
#define COM8    8
#define COM9    9
#define COM10   10
#define COM11   11
#define COM12   12
```

FrmHandle: Pointed to the communication handle which is binding with the communication port opened successfully. The application software should use this handle to manipulate the reader connected to the port.

Returns:

Zero value when successful, non-zero value(ErrorCode) when error occurred.

3.1.2) OpenComPort()

Function description:

This function is used to establish the connection between the reader and a specified communication port. The protocol parameters are 19200bps, 8 data bits, 1 start bit, 1 stop bit, no parity bit.

Usage:

OpenComPort(long Port, unsigned char *ComAdr, long *FrmHandle);

Parameter:

Port: Communication port number which is a constant from COM1 to COM12 defined as following:

```
#define COM1    1
#define COM2    2
#define COM3    3
#define COM4    4
#define COM5    5
#define COM6    6
#define COM7    7
```

#define COM8 8
#define COM9 9
#define COM10 10
#define COM11 11
#define COM12 12

ComAdr: Pointed to the address of the reader.

When using broadcasting address 0xFF as ComAdr to call the function, the address of the reader will be writed back to parameter ComAdr;

When using a designated address 0x00~0xFE as ComAdr to call the function, the function will detect whether a specified address reader is connected to the designaged communication port.

FrmHandle: Pointed to the communication handle which is binding with the communication port opened successfully. The application software should use this handle to manipulate the reader connected to the port.

Returns:

Zero value when successful, non-zero value(ErrorCode) when error occurred.

3.1.3) CloseComPort

Function description:

This function is used to disconnect the reader and release the corresponding communication port resources. In some development environment, the communication port resources must be released before exiting. Otherwise the operation system will become unstable.

Usage:

CloseComPort(void);

Parameter: None.

Returns:

Zero value when successful, non-zero value(ErrorCode) when error occurred.

3.1.4) CloseSpecComPort

Function description:

This function is used to disconnect the reader with the designated communication port and release the corresponding resources.

Usage:

CloseSpecComPort (long FrmHandle);

Parameter:

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value(ErrorCode) when error occurred.

3.1.5) GetReaderInformation()

Function description:

This function is used to get reader-related information such as reader address(ComAdr),

firmware version, supported protocol type and InventoryScanTime.

Usage:

GetReaderInformation(unsigned char *ComAdr, unsigned char *VersionInfo, unsigned char *ReaderType, unsigned char *TrType, unsigned char *InventoryScanTime, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

VersionInfo: Pointed to 2 bytes firmware version information. The first byte is version number, the second byte is sub-version number.

ReaderType: Pointed to the reader type byte. Please refer to User's manual for details.

TrType: Pointed to 2 bytes supported protocol information. Please refer to User's manual for details.

InventoryScanTime: Point to the value of time limit for inventory command. Please refer to User's manual for details.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.6) OpenRf()

Function description:

This function is used to turn on the RF inductive field.

Usage:

OpenRf(unsigned char *ComAdr, long FrmHandle);

Parameter: none.

ComAdr: Pointed to the address of the reader.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.7)CloseRf()

Function description:

This function is used to turn off the RF inductive field.

Usage:

CloseRf(unsigned char *ComAdr, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.8)WriteComAdr()

Function description:

This function is used to set a new address of the reader. The address value will store in reader's inner nonvolatile memory. Default address value is 0x00. The value range is 0x00~0xFE. The address 0xFF is reserved as the broadcasting address. When user try to write a 0xFF to ComAdr, the reader will set the value to 0x00 automatically.

Usage:

WriteComAdr(unsigned char *ComAdr, unsigned char *ComAdrData, long FrmHandle);

Parameter:

ComAdr: Pointed to the original address of the reader.

ComAdrData: Pointed to the new address of the reader.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.9)WriteInventoryScanTime()

Function description:

This function is used to set a new value to InventoryScanTime of an appointed reader. The range is 3~255 corresponding to 3*100ms~255*100ms InventoryScanTime. The default value of InventoryScanTime is 30*100ms.

Usage:

WriteInventoryScanTime(unsigned char *ComAdr, unsigned char *InventoryScanTime, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

InventoryScanTime: Pointed to the value of InventoryScanTime.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.10)SetGeneralOutput()

Function description:

This function is used to set the state of two built-in general output terminals(TTL level, default is low level).

Usage:

SetGeneralOutput(unsigned char *ComAdr, unsigned char *OutputData, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

OutputData: Pointed to the state value of two general outputs with Bit0 for G_Out1 and

bit1 for G_Out2.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.11)GetGeneralInput()

Function description:

This function is used to get the state of the general input terminal of the reader. The terminal is internally pulled up to +5V through a 20Kohm resistor(TTL level).

Usage:

GetGeneralInput(unsigned char *ComAdr, unsigned char *InputData, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

InputData: Pointed to the state value of the general input G_IN1 with bit0.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.12)SetRelay()

Function description:

This function is used to activate or deactivate the built-in relay of the reader. The relay's default state is deactivated.

Usage:

SetRelay(unsigned char *ComAdr, unsigned char *RelayAction, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

RelayAction: Pointed to the relay state value. Set bit0 to 1 to deactivate the relay and set bit0 to 0 to activate it.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.13)SetActiveANT()

Function description:

This function is used to select one of the antennas of the reader to be active. The default active antenna is Antenna 1.

Usage:

SetActiveANT(unsigned char *ComAdr, unsigned char *_ANT_Status, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

_ANT_Status: Pointed to the antenna state value. Set bit0 to 1 to activate Antenna 1, set bit1 to 1 to activate Antenna 2, set bit2 to 1 to activate Antenna 3, set bit3 to 1 to activate Antenna 4. Only one bit can be set at one time.
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.
Returns:
Zero value when successful, non-zero value when error occurred.

3.1.14)GetANTStatus()

Function description:
This function is used to get the current state of the reader's antennae.
Usage:
GetANTStatus(unsigned char *ComAdr, unsigned char *Get_ANT_Status, long FrmHandle);
Parameter:
ComAdr: Pointed to the address of the reader.
Get_ANT_Status: Pointed to the current antenna state value. Bit0 is set when Antenna 1 is active, Bit1 is set when Antenna 2 is active, Bit2 is set when Antenna 3 is active and Bit3 is set when Antenna 4 is active.
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.
Returns:
Zero value when successful, non-zero value when error occurred.

3.1.15)SetUserDefinedBlockLength()

Function description:
This function is used to set the user defined block length which to be used in user defined read/write operations or in ScanMode.
Usage:
SetUserDefinedBlockLength(unsigned char *ComAdr, unsigned char *_Block_Len, long FrmHandle);
Parameter:
ComAdr: Pointed to the address of the reader.
_Block_Len: Pointed to block length value in byte unit which is from 1 to 8.
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.
Returns:
Zero value when successful, non-zero value when error occurred.

3.1.16)GetUserDefinedBlockLength()

Function description:
This function is used to get the user defined block length value.
Usage:
GetUserDefinedBlockLength(unsigned char *ComAdr, unsigned char *_Block_Len,

long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

_Block_Len: Pointed to block length value in byte unit.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.17)SetScanMode()

Function description:

This function is used to set the reader in ScanMode and define the output content and format of the reader in ScanMode.

Usage:

SetScanMode(unsigned char *ComAdr, unsigned char *_Scan_Mode_Data, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

_Block_Len: Pointed to 11 bytes ScanMode control data. For details of the control data, please refer to the reader's user's manual.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.18)GetScanModeStatus()

Function description:

This function is used to get the status control data of the reader in ScanMode.

Usage:

GetScanModeStatus(unsigned char *ComAdr, unsigned char *_Scan_Mode_Data, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

_Block_Len: Pointed to current value of 11 bytes ScanMode control data. For details of the control data, please refer to the reader's user's manual.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.19)ReadScanModeData()

Function description:

This function is used to fetch the data which the reader send out in ScanMode.

Usage:

ReadScanModeData(unsigned char *ScanModeData, long *ValidDataLength, long FrmHandle);

Parameter:

ScanModeData: Pointed to the data the reader sends out in ScanMode. Please make sure that there is enough space in ScanModeData to accommodate the data from the reader.

ValidDataLength: Pointed to valid byte count in ScanModeData.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.20)SetAccessTime()

Function description:

This function is used to set the access time of tag passing through the inductive field.

Usage:

SetAccessTime(unsigned char *ComAdr, unsigned char *AccessTime, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

AccessTime: Pointed to time value in 100ms unit that the tag passes the field. If set to 0, the reader will ignore the value. For details of the access time setting, please refer to the reader user's manual.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.21)GetAccessTime()

Function description:

This function is used to get the current access time value of tag passing through the inductive field.

Usage:

GetAccessTime(unsigned char *ComAdr, unsigned char *AccessTimeRet, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

AccessTimeRet: Pointed to current access time value in 100ms unit that the tag passes the field. For details of the access time setting, please refer to the reader user's manual.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.22)SetReceiveChannel()

Function description:

This function is used to set the effective receiving channel of the reader. This function is available for the readers equipped with one main channel(TX/RX) and one auxiliary channel(RX only).

Usage:

SetReceiveChannel(unsigned char *ComAdr, unsigned char *ReceiveChannel, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

ReceiveChannel: Pointed to receive channel setting value. If set to 1, the reader will only use its auxiliary channel. If set to 2, the reader will only use its main channel. If set to 3, the reader will use both of its receive channels.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.23)GetReceiveChannelStatus()

Function description:

This function is used to get the current receive channel setting value.

Usage:

GetReceiveChannelStatus(unsigned char *ComAdr, unsigned char *ReceiveChannelStatus, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

ReceiveChannelStatus: Pointed to current receive channel setting value. If equal to 1, the reader is using its auxiliary channel. If equal to 2, the reader is using its main channel. If equal to 3, the reader is using both of its receive channels.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.24)SetParseMode()

Function description:

This function is used to set the reader’s parsing mode in multiple tags operation.

Usage:

SetParseMode(unsigned char *ComAdr, unsigned char *ParseMode, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

ParseMode: Pointed to parsing mode setting value. If set to 1, the reader will adopt DPPM(Depth Priority Parse Mode). If set to 0, the reader will adopt WPPM(Width Priority Parse Mode). Please refet to the reader’s user’s manual for the definition of other values.

FrmHandle: Handle of the corresponding communication port the reader is connected.

The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.1.25)GetParseMode()

Function description:

This function is used to get the reader's current parsing mode setting.

Usage:

GetParseMode(unsigned char *ComAdr, unsigned char *ParseMode, long FrmHandle);

Parameter:

ComAdr: Pointed to the address of the reader.

ParseMode: Pointed to current parsing mode setting value. If equal to 1, the reader is in DPPM(Depth Priority Parse Mode). If equal to 0, the reader is in WPPM(Width Priority Parse Mode). Please refet to the reader's user's manual for the definition of other values.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2) Basic Operation Function

3.2.1) Inventory()

Function description:

This function is used to detect the tags in the inductive area and get their UID and DSFID values.

Usage:

Inventory(unsigned char *ComAdr, unsigned char *State, unsigned char *AFI, unsigned char *DSFIDAndUID, unsigned char *CardNum, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Inventory without AFI: 0x00
Inventory with AFI: 0x01
InventoryScan without AFI: 0x02(consecutive style)
InventoryScan with AFI: 0x03(consecutive style)
InventoryScan without AFI: 0x06(renewed style)
InventoryScan with AFI: 0x07(renewed style)

The value of State defines various operation style of Inventory command, please refer to User's Manual for details.

AFI: Input. Application Family Information of the tag.

DSFIDAndUID: Output. Pointed to the array storing the inventory result. The unit of the array is 9 bytes including 1 byte DSFID and 8 bytes UID. The volume of the result is CardNum*9 bytes.

CardNum: Output. Pointed to the number of tags detected. As CardNum is a byte-type variant, the DLL limits the maximum card number that can be collected at one time to

255. Actually, the reader sets no limit to the quantity of the tags in the field.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

When returned value is 0x0e, it means there is no tag in field or all the tags have been collected;

When returned value is 0x0a, it means none tag has been collected when the time defined by InventoryScanTime overflows;

When returned value is 0x0b, it means parts (not all) of the tags have been collected when the time defined by InventoryScanTime overflows.

For details, please refer to the reader's user's manual.

3.2.2) StayQuiet()

Function description:

This function is used to set the designated tag into Quiet status.

Usage:

StayQuiet(unsigned char *ComAdr, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.3) ReadSingleBlock()

Function description:

This function is used to read out the content of one block and its security status byte.

Usage:

ReadSingleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *BlockSecStatus, unsigned char *Data, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Block size = 4 bytes	Addressed mode:	0x00
	Selected mode:	0x01
Block size = 8 bytes	Addressed mode:	0x04
	Selected mode:	0x05

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first.

When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.
BlockNum: Input. The block number to read.
BlockSecStatus: Output. Pointed to the security value of the designated block.
Data: Output. Pointed to the array of the block content with the size of 4 or 8 bytes.
ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.
Returns:
Zero value when successful, non-zero value when error occurred.

3.2.4) WriteSingleBlock()

Function description:
This function is used to write data into a block of the designated tag.
Usage:
WriteSingleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *Data, unsigned char *ErrorCode, long FrmHandle);
Parameter:
ComAdr: Input. Pointed to the address of the reader.
State: Input. Operation mode indicator byte defined as follows:
Block size=4 bytes Addressed mode: 0x00(Type A tag) 0x08(Type B tag)
Selected mode: 0x01(Type A tag) 0x09(Type B tag)
Block size=8 bytes Addressed mode: 0x04(Type A tag) 0x0C(Type B tag)
Selected mode: 0x05(Type A tag) 0x0D(Type B tag)
As to the type of a tag, please refer to appendix 1 for details.
UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.
BlockNum: Input. The block number to write.
Data: Iutput. Pointed to the data to be written into the block with the size of 4 or 8 bytes.
ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.
Returns:
Zero value when successful, non-zero value when error occurred.

3.2.5) LockBlock()

Function description:
This function is used to lock a block of the designated tag. When a block is locked, it will be permanently write-protected and its content could not be altered.
Usage:

LockBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0x00(Type A tag) 0x08(Type B tag)

Selected mode: 0x01(Type A tag) 0x09(Type B tag)

As to the type of a tag, please refer to appendix 1 for details.

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

BlockNum: Input. The block number to lock.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.6) ReadMultipleBlock()

Function description:

This function is used to read out the content of several blocks and their security status bytes in the designated tag.

Usage:

ReadMultipleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char BlockCount, unsigned char *BlockSecStatus, unsigned char *Data, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Block size = 4 bytes Addressed mode: 0x00

Selected mode: 0x01

Block size = 8 bytes Addressed mode: 0x04

Selected mode: 0x05

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

BlockNum: Input. The start block number to read.

BlockCount: Input. The number of the blocks to read. The maximum number is 28 when block size is 4 bytes and the maximum number is 15 when block size is 8 bytes.

BlockSecStatus: Output. Pointed to the security value of the designated blocks with the size of BlockCount bytes..

Data: Output. Pointed to the block content data with the size of 4 (or 8)*BlockCount bytes.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.7) Select()

Function description:

This function is used to set the tag into selected status.

Usage:

Select(unsigned char *ComAdr, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.8) ResetToReady()

Function description:

This function is used to set back the tag into ready status.

Usage:

ResetToReady(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0x00

Selected mode: 0x01

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.9) WriteAFI()

Function description:

This function is used to write the value of the designated tag's Application Family Information.

Usage:

WriteAFI(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char AFI, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0x00(Type A tag) 0x08(Type B tag)

Selected mode: 0x01(Type A tag) 0x09(Type B tag)

As to the type of a tag, please refer to appendix 1 for details.

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

AFI: Input. The value of the tag's Application Family Information.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.10) LockAFI()

Function description:

This function is used to permanently lock the AFI value of the designated tag.

Usage:

LockAFI(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0x00(Type A tag) 0x08(Type B tag)

Selected mode: 0x01(Type A tag) 0x09(Type B tag)

As to the type of a tag, please refer to appendix 1 for details.

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.11) WriteDSFID()

Function description:

This function is used to write the value of the designated tag's Data Storage Format Identifier.

Usage:

WriteDSFID(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char DSFID, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0x00(Type A tag) 0x08(Type B tag)

Selected mode: 0x01(Type A tag) 0x09(Type B tag)

As to the type of a tag, please refer to appendix 1 for details.

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

DSFID: Input. The value of the tag's Data Storage Format Identifier.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.12) LockDSFID()

Function description:

This function is used to permanently lock the DSFID value of the designated tag.

Usage:

LockAFI(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0x00(Type A tag) 0x08(Type B tag)

Selected mode: 0x01(Type A tag) 0x09(Type B tag)

As to the type of a tag, please refer to appendix 1 for details.

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:
Zero value when successful, non-zero value when error occurred.

3.2.13) GetSystemInformation()

Function description:
This function is used to acquire the detail description information of the designated tag including its Information Flag, UID, DSFID, AFI, Memory, IC Reference etc. The contents may be different for tags of various manufacturers. Please refer to tag's datasheet for details.

Usage:
GetSystemInformation(unsigned char *ComAdr, unsigned char *State, unsigned char *UIDI, unsigned char *InformationFlag, unsigned char *UIDO, unsigned char *DSFID, unsigned char *AFI, unsigned char *MemorySize, unsigned char *ICReference, unsigned char *ErrorCode, long FrmHandle);

Parameter:
ComAdr: Input. Pointed to the address of the reader.
State: Input. Operation mode indicator byte defined as follows:
Addressed mode: 0x00
Selected mode: 0x01
UIDI: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UIDI value should be given. When operating in Selected mode, UIDI value will be neglected.
InformationFlag: Output. Pointed to the tag's information flag.
UIDO: Output. Pointed to the 8 bytes of tag's UID value with least significant byte first.
DSFID: Output. Pointed to the tag's DSFID value.
AFI: Output. Pointed to the tag's AFI value.
MemorySize: Output. Point to a 2 bytes array of tag's storage size information. The first byte indicates the total number of data blocks of the tag and the second byte for the size of the data block.
ICReference: Output. Pointed to a reference byte.
ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:
Zero value when successful, non-zero value when error occurred.

3.2.14) CustomizeReadSingleBlock()

Function description:
This function is used to read out the content of one block and its security status byte. The block length is predefined by users.

Usage:
CustomizeReadSingleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *DataBuffer, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0xa0

Selected mode: 0xa1

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

BlockNum: Input. The block number to read.

DataBuffer: Output. Pointed to the security status value and content of the designated block. The first byte in DataBuffer is security status byte and the others are block content bytes. The block content length is predefined by user.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.15) CustomizeWriteSingleBlock()

Function description:

This function is used to write in the content of one block. The block length is predefined by users.

Usage:

CustomizeWriteSingleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char *DataBuffer, unsigned char ByteCount, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0xa0(Type A tag) 0xa8(Type B tag)

Selected mode: 0xa1(Type A tag) 0xa9(Type B tag)

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

BlockNum: Input. The block number to write.

DataBuffer: Input. Pointed to the content of the designated block to write into.

ByteCount: Input. The number of bytes to be written. This value should be equal to the block length predefined by user.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.2.16) CustomizeReadMultipleBlock()

Function description:

This function is used to read out the content of multiple blocks and their security status bytes. The block length is predefined by users.

Usage:

CustomizeReadMultipleBlock(unsigned char *ComAdr, unsigned char *State, unsigned char *UID, unsigned char BlockNum, unsigned char BlockCount, unsigned char *DataBuffer, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

State: Input. Operation mode indicator byte defined as follows:

Addressed mode: 0xa0

Selected mode: 0xa1

UID: Input. Pointed to the 8 bytes of tag's UID value with least significant byte first. When operating in Addressed mode, UID value should be given. When operating in Selected mode, UID value will be neglected.

BlockNum: Input. The block number to read.

BlockCount: Input. Number of consecutive blocks to read.

DataBuffer: Output. Pointed to the security status value and content of multiple blocks. The data from every block is consecutively organized in DataBuffer. The data from one block comprise one security status byte and its block content bytes. The block content length is predefined by user.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.3) Transparent Command Function

3.3.1) TransparentRead ()

Function description:

This function is used to turn the reader into an RF front-end engine to read data from tags. The reader encapsulates the host's data in ISO15693 protocol format and sends them to tags. Then, the reader decodes the feedback data from tag and sends back the results to the host. All interactive timing is according to ISO15693 protocol definition.

Usage:

TransparentRead(unsigned char *ComAdr, unsigned char RspLength, unsigned char CustomDataLength, unsigned char *CustomData, unsigned char *FeedbackDataLength, unsigned char *FeedbackData, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

RspLength: Input. The response data byte length expected from the tag which should be between 3 and 120.

CustomDataLength: The byte length of the data the reader should send to the tag with value between 1 and 120.

CustomData: Input. Pointed to the data the reader should send to the tag.

FeedbackDataLength:Output. Byte Length of the tag response data. If the operation successful, it should be 4+RspLength. For details, please refer to the reader user's manual.

FeedbackData: Output. Pointed to the data the tag responds.

ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.

FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.

Returns:

Zero value when successful, non-zero value when error occurred.

3.3.2) TransparentWrite ()

Function description:

This function is used to turn the reader into an RF front-end engine to write data into tags. The reader encapsulates the host's data in ISO15693 protocol format and sends them to tags. Then, the reader decodes the feedback data from tag and sends back the results to the host. All interactive timing is according to ISO15693 protocol definition.

Usage:

TransparentWrite(unsigned char *ComAdr, unsigned char *Option, unsigned char RspLength, unsigned char CustomDataLength, unsigned char *CustomData, unsigned char *FeedbackDataLength, unsigned char *FeedbackData, unsigned char *ErrorCode, long FrmHandle);

Parameter:

ComAdr: Input. Pointed to the address of the reader.

Option: Input. Pointed to a 4 bytes writing operation option data. For details, please refer to the reader user's manual.

RspLength: Input. The response data byte length expected from the tag which should be between 3 and 120.

CustomDataLength: Input. The byte length of the data the reader should send to the tag with value between 1 and 120.

CustomData: Input. Pointed to the data the reader should send to the tag.

FeedbackDataLength:Output. Byte Length of the tag response data. If the operation successful, it should be 4+RspLength. For details, please refer to the reader user's manual.

FeedbackData: Output. Pointed to the data the tag responds.
ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.
Returns:
Zero value when successful, non-zero value when error occurred.

3.3.3) TransparentCustomizedCmd ()

Function description:
This function is used to turn the reader into an RF front-end engine to read data from tags. The reader encapsulates the host's data in ISO15693 protocol format and sends them to tags. Then, the reader decodes the feedback data from tag and sends back the results to the host. Response decode start timing can be defined by users.
Usage:
TransparentCustomizedCmd(unsigned char *ComAdr, unsigned char *RspTime, unsigned char RspLength, unsigned char CustomDataLength, unsigned char *CustomData, unsigned char *FeedbackDataLength, unsigned char *FeedbackData, unsigned char *ErrorCode, long FrmHandle);
Parameter:
ComAdr: Input. Pointed to the address of the reader.
RspTime: Input. Pointed to a 2 bytes data defining the response decode start timing of the reader. For details, please refer to the reader user's manual.
RspLength: Input. The response data byte length expected from the tag which should be between 3 and 120.
CustomDataLength: Input. The byte length of the data the reader should send to the tag with value between 1 and 120.
CustomData: Input. Pointed to the data the reader should send to the tag.
FeedbackDataLength:Output. Byte Length of the tag response data. If the operation successful, it should be 4+RspLength. For details, please refer to the reader user's manual.
FeedbackData: Output. Pointed to the data the tag responds.
ErrorCode: Output. Pointed to an explanation byte when the function return value equals 0x0F.
FrmHandle: Handle of the corresponding communication port the reader is connected. The handle value is got when calling function AutoOpenComPort or OpenComPort.
Returns:
Zero value when successful, non-zero value when error occurred.

4) Return Value Definition

```
#define OK                    0x00
#define LengthError           0x01
#define OperationNotSupport   0x02
```

```
#define RfClosed            0x05
#define EEPROM              0x06
#define TimeOut             0x0a
#define MoreUID             0x0b
#define ISOError            0x0c
#define NoElectronicTag     0x0e
#define OperationError      0x0f
#define CommunicationErr    0x30
#define RetCRCErr           0x31
#define DataLengthErr       0x32
#define CommunicationBusy   0x33
#define ExecuteCmdBusy      0x34
#define ComPortOpened       0x35
#define ComPortClose        0x36
#define Invalidhandle       0x37
#define InvalidPort         0x38
```

5) ErrorCode Definition

```
#define CmdNotSupport               0x01
#define CmdNotIdentify              0x02
#define OperationNotSupport         0x03
#define UnknownError                0x0f
#define BlockError                  0x10
#define BlockLockedAndCntLock       0x11
#define BlockLockedAndCntWrite      0x12
#define BlockCntOperate             0x13
#define BlockCntLock                0x14
```