RR9000.DLL Dynamic Link Library

User's Manual V3.32

RR9000.DLL is a dynamic link library designed to facilitate ISO/IEC 15693 protocol HF tag application software development when using RR9 series HF tag reader/writer.

RR9 series HF tag reader/writer include the following models: RR9001, RR9003A, RR9201, RR9203A, RR9001Lite, RR9201Lite, RR9036SR, RR9036USB, RR9036CF etc.

1) Operation System Requirement

WINDOWS 2000/XP

2) Function List

RR9000.DLL includes the following functions for the ISO/IEC 15693 HF tag operation.

Remark: The functions supported may be different for various models of RR9 series HF tag reader/writer Please refer to appendix 1 for details..

2.1) General Function

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

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

3)long WINAPI CloseComPort(void);

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

5)long WINAPI OpenRf(unsigned char *comadr);

6)long WINAPI CloseRf(unsigned char *comadr);

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

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

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

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

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

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

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

2.2) Basic Operation Function

1)long WINAPI Inventory(unsigned char *ComAdr, unsigned char *State, unsigned char *AFI, unsigned char *DSFIDAndUID, unsigned char *CardNum);
2)long WINAPI StayQuiet(unsigned char *ComAdr, unsigned char *UID, unsigned char *ErrorCode);

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);

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

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

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);

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

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

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

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

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

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

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);

3) Function Explanation

3.1) General Function

3.1.1) AutoOpenComPort(long *Port, unsigned char *ComAdr)

Function description:

This function is used to detect the RR9001 reader attached automatically and establish the connection with it through the communication port. The communication protocol is 19200bps, 8 data bits, 1 start bit, 1 stop bit, no parity bit.

Usage:

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

Parameter:

Port: Pointed to the communication port number(COM1~COM10) 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    0
#define COM2    1
#define COM3    2
#define COM4    3
#define COM5    4
#define COM6    5
#define COM7    6
#define COM8    7
#define COM9    8
#define COM10    9
```

Returns:

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

3.1.2) OpenComPort(long Port, unsigned char *ComAdr)

Function description:

This function is used to establish the connection with the RR9001 reader through a specified communication port. The communication protocol is 19200bps, 8 data bits, 1 start bit, 1 stop bit, no parity bit.

Usage:

OpenComPort(long Port, unsigned char *ComAdr);

Parameter:

Port: Communication port number which is a constant from COM1 to COM10 defined in the head file RR9001.h

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 function will detect whether a specified address reader is connected to the designaged communication 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) GetReaderInformation(unsigned char *ComAdr, unsigned char *VersionInfo, unsigned char *ReaderType, unsigned char *TrType, unsigned char *InventoryScanTime)

Function description:

This function is used to get reader-related information such as reader address(ComAdr), firmware version, supported protocol type and InventoryScanTime.

Usage:

BUZZER(long Device , long OnTime , long BetweenTime , long Times);

Parameter:

ComAdr: Pointed to the address of the reader.

When using broadcasting address 0xFF as ComAdr to call the function, the reader's actual address will be written to ComAdr. When using a designated address 0x00~0xFE as ComAdr to call the function, the function will be executed on the reader with the designated address.

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 RR9001 User's manual for details.

TrType: Pointed to 2 bytes supported protocol information. Please refer to RR9001 User's manual for details.

InventoryScanTime: Point to the value of time limit for inventory command. Please refer to RR9001 User's manual for details.

Returns:

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

3.1.5) OpenRf()

Function description:

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

Usage:

OpenRf(unsigned char *ComAdr);

Parameter: none.

Returns:

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

3.1.6)CloseRf()

Function description:

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

Usage:

CloseRf(unsigned char *ComAdr);

Parameter: none.

Returns:

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

3.1.7)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);

Parameter:

ComAdr: Pointed to the original address of the reader.

ComAdrData: Pointed to the new address of the reader.

Returns:

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

3.1.8)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);

Parameter:

ComAdr: Pointed to the address of the reader.

InventoryScanTime: Pointed to the value of InventoryScanTime.

Returns:

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

3.1.9)SetGeneralOutput()

Function description:

This function is used to set the state of two general output terminals of RR9001. The default state is low level(TTL level).

Usage:

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

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.

Returns:

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

3.1.10)GetGeneralInput()

Function description:

This function is used to get the state of the general input terminal of RR9001. The terminal is internally pulled up to +5V through a 20Kohm resistor(TTL level).

Usage:

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

Parameter:

ComAdr: Pointed to the address of the reader.

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

Returns:

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

3.1.11)SetRelay()

Function description:

This function is used to activate or deactivate the relay in RR9001. The relay’s default state is deactivated when powered on.

Usage:

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

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.

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

3.1.12)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);
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.
Returns:
Zero value when successful, non-zero value when error occurred.

3.1.13)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);
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,
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);
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 RR9001 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.

Returns:

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

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);

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.

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);

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.
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);
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.
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);
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.

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);

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.

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);

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.

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);

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.

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);

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.
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);
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.
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);
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.
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);
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.
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);
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.
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
```

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
```

Appendix 1

	RR9001	RR9003A	RR9201	RR9203A	RR9001Lite	RR9201Lite	RR9036SR	RR9036USB	RR9036CF
AutoOpenComPort	√	√	√	√	√	√	√	√	√
OpenComPort	√	√	√	√	√	√	√	√	√
CloseComPort(void)	√	√	√	√	√	√	√	√	√
GetReaderInformation	√	√	√	√	√	√	√	√	√
OpenRf	√	√	√	√	√	√	√	√	√
CloseRf	√	√	√	√	√	√	√	√	√
WriteComAdr	√	√	√	√	√	√	√	√	√
WriteInventoryScanTime	√	√	√	√			√	√	√
SetGeneralOutput	√	√	√	√					
GetGeneralInput	√	√	√	√					
SetRelay	√	√	√	√					
SetActiveANT		√		√					
GetANTStatus		√		√					
Inventory	√	√	√	√	√	√	△*	△*	△*
StayQuiet	√	√	√	√			√	√	√
ReadSingleBlock	√	√	√	√			√	√	√
WriteSingleBlock	√	√	√	√			√	√	√
LockBlock	√	√	√	√			√	√	√
ReadMultipleBlock	√	√	√	√			√	√	√
Select	√	√	√	√			√	√	√
ResetToReady	√	√	√	√			√	√	√
WriteAFI	√	√	√	√			√	√	√
LockAFI	√	√	√	√			√	√	√
WriteDSFID	√	√	√	√			√	√	√
LockDSFID	√	√	√	√			√	√	√
GetSystemInformation	√	√	√	√			√	√	√

* InventoryScan mode not supported.

Appendix 2

Manufacturer	Manu.Code	Block Information		TYPE	
		Number	Size	A	B
Infineon (ISO Address mode)	0x05	256(user range:0～249)	4bytes		√
		64(user range:0～57)	4bytes		√
STMicroelectronics (LRI512)	0x02	16(user range:0~15)	4bytes		√
Fujitsu (MB89R116)	0x08	256(user range:0～249)	8bytes	√	√
Philips (I-Code SLI)	0x04	32(user range:0～27)	4bytes		√
Texas Instruments (Tag-it HF-I)	0x07	64(user range:0～63)	4bytes	√	

Remark: For detail information and other tags, please refer to corresponding tag's datasheet.