Type-A and CPU IC Card Reader

DLL Library Reference Manual

(Android 2.2　ISO14443 typeA)

2012-10-18

1. IC card reader API function set

1.1 System Function

1.1.1 open

Features	Open the specified serial port and baud rate
Function prototype	HANDLE open (String nComPath, int nBaudrate)
Description	Open the specified serial port and baud rate Input parameters: nCom: Serial port "/dev/ttyS0" nBaudrate: serial port baud rate (9600, 19200, 38400, 57600, 115200) Output parameters: none
Return value(s)	If successful the function returns the window handle Otherwise it returns 0 (INVALID_HANDLE_VALUE)

1.1.2 close

Features	Close the specified port
Function prototype	BOOL close (HANDLE commHandle)
Description	Close the specified window Input parameter: CommHandle Specifies the window handle Output parameters: none
Return value(s)	If successful the function returns 1, otherwise 0 (ERROR)

1.1.3 getVersion

Features	Read version information
Function prototype	int getVerion(byte[] ver);
Description	Send ISO14443A card writing instruction Input parameters: none Output parameters: If successful buffer, buffer: version information

Return value(s)	If successful the function returns 0, otherwise it returns an error code (pls. see appendix)

1.2 ISO14443 Type-A Function

1.2.1 MFRead

Features	Integrated card inventory, anti-collision, card selection, password verification, card readers and other operations, a card reader to complete the command operation.
Function prototype	int MFRead (HANDLE commHandle, int DeviceAddress,unsigned char mode, unsigned char blk_add, unsigned char num_blk,unsigned char *snr, unsigned char *buffer)
Description	ISO14443 A reader sent instructions Input parameters: commHandle - Serial handle DeviceAddress - Device Address To communicate with the device address number, ranging from 0-255 Read mode control mode 0x00: Idle Mode + Key A 0x01: All Mode + Key A 0x02: Idle Mode + Key B 0x03: All Mode + Key B blk_add To read start address block, ranging from 0--63 num_blk The number of blocks to be read length value, ranging from 1-4 (m1 card) snr 6 byte key Output parameters: snr If successful, the 4-byte card serial number (from low to high) buffer If successful, buffer [0-N]: Returns the data from the card (16 * num_blk bytes) If it fails, buffer [0]: Returns the state, the specific meaning of Annex. Note: You can only read the block of the sector, because each sector has its own independent key
Return value(s)	If successful the function returns 0, otherwise it returns an error code (pls. see appendix)

1.2.2 MFWrite

Features	Integrated card inventory, anti-collision, card selection, password verification, write card operation, a write command is completed card operation.
Function prototype	int MFWrite (HANDLE commHandle, int DeviceAddress,unsigned char mode, unsigned char blk_add, unsigned char num_blk, unsigned char *key, unsigned char *buffer)
Description	Send ISO14443 A written instruction card Input parameters: commHandle - Serial handle DeviceAddress - Device Address To communicate with the device address number, ranging from 0-255 write mode control mode 0x00: Idle Mode + Key A 0x01: All Mode + Key A 0x02: Idle Mode + Key B 0x03: All Mode + Key B blk_add Start to write the address block, ranging from 0--63 num_blk Write length value of the number of blocks, ranging from 1-4 (m1 card) key 6 byte key buffer Data to be written ((16 * num_blk bytes)) Output parameters: buffer If successful, buffer [0-3]: 4-byte card serial number (from low to high) If it fails, buffer [0]: Returns the state, the specific meaning of Annex. Note: You can only write block this sector, because each sector has its own independent key
Return value(s)	If successful the function returns 0, otherwise it returns an error code (pls. see appendix)

1.2.3 getCardSN

Features	Integrated card inventory, anti-collision, card selection, a command to read the card serial number to complete the operation
Function prototype	int getCardSN (HANDLE commHandle,int DeviceAddress, unsigned char mode,unsigned char cmd,unsigned char *flag, unsigned char *buffer)

Description	Input parameters: commHandle - Serial handle DeviceAddress - Device Address To communicate with the device address number, ranging from 0-255 mode 0x00: All mode 0x01: Idle Mode cmd 0x00: no need to perform halt instruction 0x01: reader instruction execution halt Output parameters: flag - single sign Kadoka 0x00 - detected a card 0x01 - detects multiple cards (called again to get another Uid). buffer If successful, buffer [0-3]: 4-byte card serial number (from low to high) If it fails, buffer [0]: Returns the state, the specific meaning of Annex.
Return value(s)	If successful the function returns 0, otherwise it returns an error code (pls. see appendix)

1.3.1 CPU_RATS

Features	RATS Operation
Function prototype	int CPU_RATS(int handle, int address, byte[] param, byte[] buff, byte[] retlen)
Description	Input parameters: handle - handle serial address - device address To communicate with the device address number, ranging from 0-255 param Type the command parameter Output parameters: buff If successful, RATS returned data If it fails, buffer [0]: Returns the state, the specific meaning of Annex. Retlen Returns the length of the data
Return value(s)	If successful the function returns 0, otherwise it returns an error code (pls. see appendix)

1.3.2 CPU_APDU

Features	APDU Operation
Function prototype	int CPU_APDU(int handle, int address, byte[] param, byte paramlen, byte[] buff, byte[] retlen)
Description	Input parameters: handle - window handle address - device address To communicate with the device address number, ranging from 0-255 Param Type the command parameter Paramlen enter the command parameter length Output parameters: buff If successful, the data returned APDU Retlen Returns the length of the data
Return value(s)	If successful the function returns 0, otherwise it returns an error code (pls. see appendix)

1.3.3 CPU_APDU

Features	Reset Antenna Operation
Function prototype	int CPU_RST_Ant(int handle, int address, byte[] buff, byte[] retlen)
Description	Input parameters: handle - window handle address - device address To communicate with the device address number, ranging from 0-255 Output parameters: buff If successful, the data returned APDU Retlen Returns the length of the data
Return value(s)	If successful the function returns 0, otherwise it returns an error code (pls. see appendix)

Appendix

API function return code (return value)

Return	Description
0x00	It indicates that the command is executed successfully
0x02	Indicates the length of the received data does not match
0x03	It indicates the serial port failure
0x04	Serial not received any data showing
0x05	Indicates the device address does not match
0x07	It indicates the checksum is not correct
0x0A	Represents the input parameters is incorrect, see description of the function

Reader status return codes (S)

Status	Description
0x00	Successful execution
0x01	Command execution failure (see specific instructions function)
0x80	Setting parameters indicating success
0x81	It represents the parameter setting failure
0x82	Communication timeout
0x83	Card does not exist
0x84	Card data receiving error
0x85	Input parameters or command format false
0x87	Indicating an unknown error
0x89	Parameter of the command or the command format error
0x8A	An error occurred in the initialization block
0x8B	Get the wrong serial number in the anti-collision process
0x8C	No password authentication through
0x8f	Input instruction code does not exist
0x90	Card does not support this command
0x91	Command format error
0x92	FLAG parameter represents the command is not supported OPTION mode
0x93	He said to operate BLOCK does not exist
0x94	Object to be operated has locked and can not be modified
0x95	Showing the locking operation is unsuccessful
0x96	A write operation is not successful
0x89	Flag in 15693 card is incorrect

Example

1. Copy the text file libReaderAndroid.so and java ReaderAndroid.java to your own projects, they need to reside the same directory. Or it can not successfully compiled.

2. Reader Android class function calls in your own projects you can achieve the appropriate Features

```
  try {
        byte[] bPWS=new byte[6];
        byte[] buf=new byte[30];
        ReaderAndroid mSerialPort = new ReaderAndroid(new File("/dev/ttyS0"), 9600);

        int handle = mSerialPort.getHandle();
        mSerialPort.MFRead(handle,0,(byte)0x00,(byte)0x01,(byte)0x01,bPWS,buf);
        mSerialPort.close(handle );
       }
  catch (SecurityException e){
           // Exception Handling
       }
  catch (IOException e) {
          // Exception Handling
       }
  catch (InvalidParameterException e) {
       // Exception Handling
      }
```