



ID Access 1000 / ID Module 1000
13.56 MHz RFID Device
DLL Library Reference Manual

iDTRONIC GmbH
Ludwig-Reichling-Straße 4
67059 Ludwigshafen
Germany/Deutschland

Phone: +49 621 6690094-0
Fax: +49 621 6690094-9
E-Mail: info@idtronic.de
Web: idtronic.de

Issue 3.1
– 24. January 2018 –

Subject to alteration without prior notice.
© Copyright iDTRONIC GmbH 2018
Printed in Germany

Contents

1	API Function Sets.....	5
1.1	System Function	5
1.1.1	API_OpenClientSocket.....	5
1.1.2	API_CloseSocket	5
1.1.3	SearchDevice	5
1.1.4	API_SetDeviceAddress.....	5
1.1.5	API_SetBaudrate.....	6
1.1.6	API_SetSerNum	6
1.1.7	API_GetSerNum.....	7
1.1.8	API_GetVersionNum.....	7
1.1.9	API_ControlLED	7
1.1.10	API_ControlBuzzer	8
1.1.11	GetC2000Setting.....	8
1.1.12	SetC2000Setting	8
1.1.13	ApplySetting	9
1.1.14	API_ControlRelay.....	9
1.2	ISO14443 Type-A Function	9
1.2.1	API_MF_GET_SNR	9
1.2.2	API_MF_Halt.....	10
1.3	ISO14443A Mifare Function.....	10
1.3.1	API_MF_Read	10
1.3.2	API_MF_Write	11
1.3.3	API_MF_InitVal	11
1.3.4	API_MF_Dec	12
1.3.5	API_MF_Inc.....	12
1.4	Mifare UltraLight	13
1.4.1	API_MF_Read	13
1.4.2	API_MF_Write	13
1.5	CPU Card Function.....	14
1.5.1	API_MF_PowerOn	14
1.5.2	API_MF_TransferCMD	14
1.5.3	API_MF_RST_Antenna.....	15
1.6	ISO14443 Type-B Function.....	15
1.6.1	API_Request_B	15
1.6.2	API_Anticoll_B	15
1.6.3	API_Attrib_B	16
1.6.4	API_RESET_B.....	16
1.6.5	API_TransferCMD_B	16
1.7	ISO15693 Function	17
1.7.1	API_ISO15693_Inventory.....	17
1.7.2	API_ISO15693_Read	17
1.7.3	API_ISO15693_Write	17
1.7.4	API_ISO15693_Lock	18
1.7.5	API_ISO15693_StayQuiet	18
1.7.6	API_ISO15693_Select	18
1.7.7	API_ISO15693_ResetToReady	19
1.7.8	API_ISO15693_WriteAFI.....	19
1.7.9	API_ISO15693_LockAFI.....	19

1.7.10	API_ISO15693_WriteDSFID	20
1.7.11	API_ISO15693_LockDSFID.....	20
1.7.12	API_ISO15693_GetSysInfo	20
1.7.13	API_ISO15693_GetMulSecurity	21
1.7.14	API_ISO15693_TransCmd	21
2	Appendix	22
2.1	API Function return value.....	22
2.2	Reader Status return code	22
3	Structure Type Declaration	23
4	Flow of setting network parameter.....	25

1 API Function Sets

1.1 System Function

1.1.1 API_OpenClientSocket

Function	Open client socket
Prototype	SOCKET API_OpenClientSocket(unsigned char *ip, unsigned int port)
Description	Input parameter: Ip: such as: ip = {192, 168, 1, 1}; Port: such as: port = 1024 Output parameter: None
Return	Return SOCKET when success, otherwise will return error code

1.1.2 API_CloseSocket

Function	Close socket
Prototype	int API_CloseSocket(SOCKET s)
Description	Input parameter: s: socket description symbol Output parameter: None
Return	Return as 0 when succeed, otherwise will be not 0

1.1.3 SearchDevice

Function	Search device to get device information
Prototype	int SearchDevice(DEVICEINFO *pDI, int nCount, DWORD dwTimeout);
Description	Input parameter: nCount: amount of devices information to be kept in pDI dwTimeout: maximum time(Unit: ms) of executing this CMD, Output : pDI: use to keep device information searched description: typedef struct _DEVICEINFO { char strMAC[20]; //MAC address, format: 00:09:F6:01:02:03 char strIP[20]; //IP address, format: 10.1.1.1 DWORD dwModel; //device model DWORD dwVersion; //version number char strName[20]; //device name BYTE pbUnused[100]; //not be used }
Return	Number of device be found

1.1.4 API_SetDeviceAddress

Function	Set Device address
----------	--------------------

Prototype	int API_SetDeviceAddress(HANDLE commHandle, int DeviceAddress, unsigned char NewAddress, unsigned char *Buffer)
Decription	<p>Input parameter:</p> <p>commHandle: comport handle</p> <p>DeviceAddress: device address to be communicate, range from 0 ~ 255</p> <p>NewAddress: new device address, range from 0~255</p> <p>Output parameter:</p> <p>Buffer</p> <p>If succeed, Buffer[0] is the new address after set</p> <p>If failed, Buffer[0] is the status code returned from reader, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.5 API_SetBaudrate

Function	Set device baudrate
Prototype	int API_SetBaudrate(HANDLE commHandle, int DeviceAddress, unsigned char NewBaud, unsigned char *Buffer)
Decription	<p>Input parameter:</p> <p>commHandle: comport handle</p> <p>DeviceAddress: device address to be communicated, range from 0~255</p> <p>NewBaud: new baudrate code</p> <p>0x00 – 9600 bps</p> <p>0x01 – 19200 bps</p> <p>0x02 – 38400 bps</p> <p>0x03 – 57600 bps</p> <p>0x04 – 115200 bps</p> <p>> 0x04—9600 bps</p> <p>Output parameter:</p> <p>Buffer</p> <p>If succeed, Buffer[0] is the new baudrate code after set</p> <p>If failed, Buffer[0] is the status code returned from reader, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.6 API_SetSerNum

Function	Set device serial number
Prototype	int API_SetSerNum(HANDLE commHandle, int DeviceAddress, unsigned char *NewValue, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0 ~ 255</p> <p>NewValue: new serial number (8byte)</p> <p>Output parameter:</p> <p>Buffer</p> <p>If succeed, Buffer[0]=0x80</p>

	If failed, Buffer[0] is the status code returned from reader, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.7 API_GetSerNum

Function	Get device serial number
Prototype	int API_GetSerNum(HANDLE commHandle, int DeviceAddress, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0~255</p> <p>Output parameter:</p> <p>Buffer</p> <p>If succeed, Buffer[0] is the reader's address, then Buffer[1-8] is the serial number</p> <p>If failed, Buffer[0] is the status code returned from reader, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.8 API_GetVersionNum

Function	Get reader version number
Prototype	int API_GetVersionNum(HANDLE commHandle, int DeviceAddress, char *VersionNum)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0~255</p> <p>Output parameter:</p> <p>VersionNum</p> <p>If succeed, VersionNum[0-7] is the version</p> <p>If failed, VersionNum[0] is the status code returned from reader, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.9 API_ControlLED

Function	Control reader's LED
Prototype	int API_ControlLED(HANDLE commHandle, int DeviceAddress, unsigned char freq, unsigned char duration, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0~255</p> <p>freq: cycle count of LED lighting in one cycle (20ms for one cycle)</p> <p>duration: cycle time of LED status(1seconds for one cycle)</p> <p>Output parameter:</p> <p>Buffer</p>

	<p>If succeed, Buffer[0]=0x80</p> <p>If failed, Buffer[0] is the status code returned from reader, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.10 API_ControlBuzzer

Function	Control reader's buzzer
Prototype	int API_ControlBuzzer(HANDLE commHandle, int DeviceAddress, unsigned char freq, unsigned char duration, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0~255</p> <p>freq: cycle counts of Buzzer making sound in one cycle(100ms for one second)</p> <p>duration: cyclic time of the buzzer status (1 second for one cycle)</p> <p>Output parameter:</p> <p>Buffer</p> <p>If succeed, Buffer[0]=0x80</p> <p>If failed, Buffer[0] is the status code returned from reader, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.11 GetC2000Setting

Function	To get setting information of designated device
Prototype	BOOL GetC2000Setting(const char *lpszMac, C2000NETSETTING *pNS, int *nNetCount, C2000SOCKETSETTING *pSS, int *nSocketCount, C2000COMSETTING *pCS, int *nComCount, DWORD dwTimeout)
Description	<p>Parameter:</p> <p>lpszMac: to get MAC address of the device</p> <p>pNS: to keep the network setting information returned, the space can be assigned by invoker</p> <p>nNetCount: the count of pNS when inputting, right count will be returned</p> <p>pSS: use to store the socket setting information returned, the space can be assigned by invoker</p> <p>nSocketCount: the count of pSS when inputting, right count will be returned</p> <p>pCS: to store the port's setting information</p> <p>nComCount: the count of pCs when inputting, right count will be returned</p> <p>dwTimeout: maximum time of operating this CMD, over it, the function will be returned. Unit as ms</p>
return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.12 SetC2000Setting

Function	To set the device
Prototype	BOOL SetC2000Setting(const char *lpszMac, C2000NETSETTING *pNS, int

	nNetCount, C2000SOCKETSETTING *pSS, int nSocketCount, C2000COMSETTING *pCS, int nComCount, DWORD dwTimeout);
Description	Parameter: lpszMac: the MAC address of device to be set pNS: network parameter to be set nNetCount: amount of network pSS: socket parameter to be set nSocketCount: amount of socket pCS: port parameter to be set nComCount: amount of port dwTimeout: maximum time of operating this CMD, after it, the function will be returned. Unit as ms
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.13 ApplySetting

Function	Apply setting device
Prototype	BOOL ApplySetting(const char *lpszMac, DWORD dwTimeout)
Description	Parameter: lpszMac: the MAC address of device dwTimeout: maximum time of operating this CMD, after it, the function will be returned. Unit as ms
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.1.14 API_ControlRelay

Function	Control reader's relay
Prototype	int API_ControlRelay(SOCKET commHandle, int DeviceAddress, unsigned char flag, unsigned char *Buffer)
Description	Input parameter: commHandle: comport handle DeviceAddress: device address to be communicated, range from 0~255 flag: 0x01:ON 0x00:OFF Output parameter: Buffer If succeed, Buffer[0]=0x80 If failed, Buffer[0] is the status code returned from reader, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.2 ISO14443 Type-A Function

1.2.1 API_MF_GET_SNR

Function	Get card SNR
----------	--------------

Prototype	int API_MF_GET_SNR(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Cmd, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: com handle</p> <p>DeviceAddress: device address, range from 0x00~0xFF</p> <p>Mode: request mode</p> <p>Idle: 0x26</p> <p>All: 0x52</p> <p>Cmd: Cmd=0x00</p> <p>Output parameter:</p> <p>Buffer[0]: if succeed, return data length</p> <p>If faild, is the status code returned from reader, detail definition, please refer to APPENDIX</p> <p>Buffer[1-N]: card serial number</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.2.2 API_MF_Halt

Function	To halt the reader device
Prototype	int API_MF_Halt(HANDLE commHandle, int DeviceAddress)
Description	<p>Input parameter:</p> <p>commHandle: com handle</p> <p>DeviceAddress: device address to be communicated, range from 0x00~0xFF</p> <p>Output parameter:</p> <p>None</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.3 ISO14443A Mifare Function

1.3.1 API_MF_Read

Function	Integrated functions with request, anticollision, select card, password authentication,, etc in one command to finish reading card
Prototype	int API_MF_Read(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Blk_add, unsigned char Num_blk, unsigned char *Key, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Mode: control reading mode</p> <p>0x00: Idle mode + Key A</p> <p>0x01: All mode + Key A</p> <p>0x02: Idle mode + Key B</p> <p>0x03: All mode + Key B</p> <p>Blk_add: start address of the block to be read, range from 0x00~0xFE</p> <p>Num_blk: lengths of block to be read, range from 1~4 (Mifare 1k card)</p> <p>Key[0-5]: 6 bytes key</p>

	<p>Output parameter:</p> <p>Buffer:</p> <p>If succeed, Buffer[0]: return data length</p> <p>Buffer[1-4]: 4 bytes card serial number (from low to high))</p> <p>Buffer[5-N]: data of card</p> <p>If failed, Buffer[0]: is the status code returned from reader, detail definition, please refer to APPENDIX</p> <p>Note: this function enable to read block of this sector only, because each sector has its own unique key</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.3.2 API_MF_Write

Function	Integrated functions with request, anticollision, select card, password authentication, write card, etc in one command, to finish writing operation
Prototype	int API_MF_Write(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Blk_add, unsigned char Num_blk, unsigned char *Key, unsigned char *Senddata, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Mode: control reading mode</p> <p>0x00: Idle mode + Key A</p> <p>0x01: All mode + Key A</p> <p>0x02: Idle mode + Key B</p> <p>0x03: All mode + Key B</p> <p>Blk_add: start address of block to be written, range from 0x00~0xFE</p> <p>Num_blk: numbers of blocks to be written, range from 1~4 (Mifare 1K)</p> <p>Key[0-5]: 6 bytes Key</p> <p>Senddata[0-M]: data to be written (16 * Num_blk byte)</p> <p>Output parameter:</p> <p>Buffer:</p> <p>If succeed, Buffer[0]: length of data be return</p> <p>Buffer[1-4]: 4 bytes card serial number (low byte in front)</p> <p>If failed, Buffer[0]: is the status code returned from reader, detail definition, please refer to APPENDIX</p> <p>Note: this function enable to read block of this sector only, because each sector has its own unique key</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.3.3 API_MF_InitVal

Function	E-wallet initialize
Prototype	int API_MF_InitVal(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Sec_num, unsigned char *Key, unsigned char *Value, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p>

	<p>DeviceAddress: range from 0x00~0xFF</p> <p>Mode: (operation mode)</p> <p>0x00: Idle mode + Key A</p> <p>0x01: All mode + Key A</p> <p>0x02: Idle mode + Key B</p> <p>0x03: All mode + Key B</p> <p>Sec_num: Sector Number</p> <p>Key[0-5]: 6 bytes key</p> <p>Value[0-3]: initialize value</p> <p>Output parameter:</p> <p>Buffer:</p> <p>If succeed, Buffer[0]: length of data be returned</p> <p>Buffer[1-N]: data returned</p> <p>If failed, Buffer[0]: is the status code returned from reader, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.3.4 API_MF_Dec

Function	E-wallet decrement
Prototype	int API_MF_Dec(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Sec_num, unsigned char *Key, unsigned char *Value, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Mode: (operation mode)</p> <p>0x00: Idle mode + Key A</p> <p>0x01: All mode + Key A</p> <p>0x02: Idle mode + Key B</p> <p>0x03: All mode + Key B</p> <p>Sec_num: Sector Number</p> <p>Key[0-5]: 6 bytes key</p> <p>Value[0-3]: value to be decreased</p> <p>Output parameter:</p> <p>Buffer:</p> <p>If succeed, Buffer[0]: data length returned</p> <p>Buffer[1-N]: data returned</p> <p>If failed, Buffer[0]: is the status code returned from reader, detail definition, please refer to APPENDIX.</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.3.5 API_MF_Inc

Function	E-wallet increment
Prototype	int API_MF_Inc(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Sec_num, unsigned char *Key, unsigned char *Value, unsigned char *Buffer)

Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Mode: (operation mode)</p> <p>0x00: Idle mode + Key A</p> <p>0x01: All mode + Key A</p> <p>0x02: Idle mode + Key B</p> <p>0x03: All mode + Key B</p> <p>Sec_num: Sector Number</p> <p>Key[0-5]: 6 bytes key</p> <p>Value[0-3]: value to be increased</p> <p>Output parameter:</p> <p>Buffer:</p> <p>If succeed, Buffer[0]: data length returned</p> <p>Buffer[1-N]: data returned</p> <p>If failed, Buffer[0]: is the status code returned from reader, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.4 Mifare UltraLight

1.4.1 API_MF_Read

Function	To read one block of Mifare Ultralight card
Prototype	int API_MF_Read(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Blk_add, unsigned char Num_blk, unsigned char *Key, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle — Com handle</p> <p>DeviceAddress — range from 0—255</p> <p>Mode: Read-Page Mode, Mode=0x00</p> <p>Blk_add: address of Page (0x00 ~0xFF)</p> <p>Num_blk: numbers of Page, Num_blk=0x01</p> <p>Key[0-5]: Key=NULL</p> <p>Output parameter</p> <p>Buffer[0]: If succeed, data length returned</p> <p>If failed, it is the status returned from reader, detail description please refer to APPENDIX</p> <p>Buffer[1-3]: card block data(4 byte)</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.4.2 API_MF_Write

Function	To write one block of Mifare Ultralight
Prototype	int API_MF_Write(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Blk_add, unsigned char Num_blk, unsigned char *Key, unsigned char *Senddata, unsigned char *Buffer)

Description	<p>Input parameter:</p> <p>commHandle — Com handle</p> <p>DeviceAddress — range from 0x00 ~ 0xFF</p> <p>Mode: Write page Mode, Mode=0x00</p> <p>Blk_add: address of Page (0x00 ~ 0xFF)</p> <p>Num_blk: number of Page, Num_blk=0x01</p> <p>Key [0-5]: Key=NULL</p> <p>Senddata[0-3]: data to be written (4 byte)</p> <p>Output parameter:</p> <p>Buffer[0]:</p> <p>If succeed, buffer[0]=0x80</p> <p>If failed, it is the status returned from reader, detail description please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.5 CPU Card Function

1.5.1 API_MF_PowerOn

Function	CPU card Poweron
Prototype	int API_MF_PowerOn(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Cmd, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com port handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Mode: CPU card Poweron mode</p> <p>Idle: 0x26</p> <p>All: 0x53</p> <p>Cmd: Power-on command parameter, fixed as 0x00</p> <p>Output parameter:</p> <p>Buffer[0]: if succeed, return length of data</p> <p>If failed, return error status, detail definition, please refer to APPENDIX</p> <p>Buffer[1-N]: card UID</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.5.2 API_MF_TransferCMD

Function	CPU card data transferring
Prototype	int API_MF_TransferCMD(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Cmdlength, unsigned char *Cmd, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com port handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Mode: CPU card data transfer mode, Mode=0x00</p> <p>Cmdlength: data length to be transferred</p> <p>Cmd[0-M]: data to be transferred</p> <p>Output parameter:</p>

	Buffer[0]: if success, return data length If failed, return error status, detail definition, please refer to APPENDIX Buffer[1-N]: data returned
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.5.3 API_MF_RST_Antenna

Function	Reset Antenna
Prototype	int API_MF_RST_Antenna(HANDLE commHandle, int DeviceAddress, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.6 ISO14443 Type-B Function

1.6.1 API_Request_B

Function	ISO14443 TypeB card Request
Prototype	int API_Request_B(HANDLE commHandle, int DeviceAddress, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Output parameter: Buffer[0]: if success, return data length If failed, return error status, detail definition, please refer to APPENDIX Buffer[1-N]: output data
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.6.2 API_Anticoll_B

Function	ISO14443 TypeB card anticollision
Prototype	int API_Anticoll_B(HANDLE commHandle, int DeviceAddress, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Output parameter: Buffer[0]: if succeed, return data length If failed, return error status, detail definition, please refer to APPENDIX

	Buffer[1-N]: output data
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.6.3 API_Attrib_B

Function	ISO14443 TypeB card Attrib
Prototype	int API_Attrib_B(HANDLE commHandle, int DeviceAddress, unsigned char *SerialNum, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com port handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>SerialNum[0-3]: card serial number (4byte)</p> <p>Output parameter:</p> <p>Buffer[0]: if succeed, Buffer[0]=0x80</p> <p>If failed, return error status, detail definition, please refer to APPENDIX</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.6.4 API_RESET_B

Function	ISO14443 TypeB card Reset
Prototype	int API_RESET_B(HANDLE commHandle, int DeviceAddress, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com port handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Output parameter:</p> <p>Buffer[0]: if succeed, return data length</p> <p>If failed, return error status, detail definition, please refer to APPENDIX</p> <p>Buffer[1-N]: output data</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.6.5 API_TransferCMD_B

Function	ISO14443 TypeB data transferring
Prototype	int API_TransferCMD_B(HANDLE commHandle, int DeviceAddress, unsigned char CmdSize, unsigned char *Cmd, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com port handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>CmdSize: length of data input</p> <p>Cmd[0-M]: data to be input (CmdSize byte)</p> <p>Output parameter:</p> <p>Buffer[0]: if succeed, Buffer[0]=0x80</p> <p>If failed, return error status, detail definition, please refer to APPENDIX</p> <p>Buffer[1-N]: output data</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7 ISO15693 Function

1.7.1 API_ISO15693_Inventory

Function	ISO15693 card Inventory
Prototype	int API_ISO15693_Inventory(HANDLE commHandle, int DviceAddress, unsigned char Flag, unsigned char Afi, unsigned char Datalen, const unsigned char *pData, unsigned char *pBuffer)
Description	<p>Input parameter:</p> <p>commHandle: Com port handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Flag: flags</p> <p>Afi: AFI</p> <p>Datalen: sending data length</p> <p>pData: sending data</p> <p>Output parameter:</p> <p>pBuffer[0]: if success, return data length</p> <p>If failed, return error status, detail definition, please refer to APPENDIX</p> <p>pBuffer[1-N]: card information</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.2 API_ISO15693_Read

Function	ISO15693 card Reading
Prototype	int API_ISO15693_Read(HANDLE commHandle, int DeviceAddress, unsigned char Flag, unsigned char Blk_add, unsigned char Num_blk, unsigned char *uid, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com port handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Flag: flags</p> <p>Blk_add: block number to be read</p> <p>Num_blk: number of blocks to be read</p> <p>uid: card UID</p> <p>Output parameter:</p> <p>Buffer[0]: if success, return data length</p> <p>If failed, return error status, detail definition, please refer to APPENDIX</p> <p>Buffer[1-N]: Read card information</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.3 API_ISO15693_Write

Function	ISO15693 card Writing
Prototype	int API_ISO15693_Write(HANDLE commHandle, int DeviceAddress, unsigned char Flag, unsigned char Blk_add, unsigned char Num_blk, unsigned char *uid, unsigned char *data)
Description	Input parameter:

	commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flag: flags Blk_add: block number to be written Num_blk: number of blocks to be written uid: card UID data: data to be written Output parameter: None
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.4 API_ISO15693_Lock

Function	ISO15693 card locking
Prototype	int API_ISO15693_Lock(HANDLE commHandle, int DeviceAddress, unsigned char Flag, unsigned char Addr_blk, unsigned char *uid, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flag: flags Addr_blk: block number to be locked uid: card UID Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.5 API_ISO15693_StayQuiet

Function	ISO15693 set card to be stayed quiet
Prototype	int API_ISO15693_StayQuiet(HANDLE commHandle, int DeviceAddress, unsigned char Flag, unsigned char *uid, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flag: flags uid: card UID Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.6 API_ISO15693_Select

Function	ISO15693 Card Selecting
Prototype	int API_ISO15693_Select(HANDLE commHandle, int DeviceAddress, unsigned char Flags, unsigned char *uid, unsigned char *Buffer)

Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flags uid: card UID Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.7 API_ISO15693_ResetToReady

Function	ISO15693 reset card to be READY status
Prototype	int API_ISO15693_ResetToReady(HANDLE commHandle, int DeviceAddress, unsigned char Flags, unsigned char *uid, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flags uid: card UID Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.8 API_ISO15693_WriteAFI

Function	ISO15693 Write AFI
Prototype	int API_ISO15693_WriteAFI(HANDLE commHandle, int DeviceAddress, unsigned char Flags, unsigned char Afi, unsigned char *uid, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flags Afi: AFI uid: card UID Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.9 API_ISO15693_LockAFI

Function	ISO15693 Lock AFI
Prototype	int API_ISO15693_LockAFI(HANDLE commHandle, int DeviceAddress, unsigned char Flags, unsigned char *uid, unsigned char *Buffer)

Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flags uid: card UID Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.10 API_ISO15693_WriteDSFID

Function	ISO15693 write DSFID
Prototype	int API_ISO15693_WriteDSFID(HANDLE commHandle, int DeviceAddress, unsigned char Flags, unsigned char DSFID, unsigned char *uid, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flag DSFID: DSFID uid: card UID Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.11 API_ISO15693_LockDSFID

Function	ISO15693 DSFID
Prototype	int API_ISO15693_LockDSFID(HANDLE commHandle, int DeviceAddress, unsigned char Flags, unsigned char *uid, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flags uid: card UID Output parameter: Buffer[0]: if success, Buffer[0]=0x80 If failed, return error status, detail definition, please refer to APPENDIX
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.12 API_ISO15693_GetSysInfo

Function	ISO15693 get card system information
Prototype	int API_ISO15693_GetSysInfo(HANDLE commHandle, int deviceAddress, unsigned char Flag, unsigned char *uid, unsigned char *Buffer)
Description	Input parameter:

	commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flags uid: card UID Output parameter: Buffer[0]: if success, Return information length If failed, return error status, detail definition, please refer to APPENDIX Buffer[1-N]: Return information
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.13 API_ISO15693_GetMulSecurity

Function	ISO15693 get card secure information
Prototype	int API_ISO15693_GetMulSecurity(HANDLE commHandle, int DeviceAddress, unsigned char Flags, unsigned char BlkAddr, unsigned char BlkNum, const unsigned char *uid, unsigned char *pBuffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flags BlkAddr: block number BlkNum: number of blocks uid: card UID Output parameter: Buffer[0]: if success, Return information length If failed, return error status, detail definition, please refer to APPENDIX Buffer[1-N]: Return information
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

1.7.14 API_ISO15693_TransCmd

Function	ISO15693 transfer any data and commands to card
Prototype	int API_ISO15693_TransCmd(HANDLE commHandle, int DeviceAddress, int CmdSize, unsigned char *Cmd, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flags: flags CmdSize: command length Cmd[0-M]: command uid: card UID Output parameter: Buffer[0]: if success, return data length If failed, return error status, detail definition, please refer to APPENDIX Buffer[1-N]: Return data
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer to APPENDIX

2 Appendix

2.1 API Function return value

Return	Description
0x00	CMD execute succeed
0x02	Data length received not matching up
0x03	Serial port transmitting failed
0x04	No data received by serial port
0x05	Device address not matching up
0x07	Checksum error
0x0A	Input parameter error, please refer to unspecific functions declaration

2.2 Reader Status return code

Status	Description
0x00	CMD execute succeed
0x01	CMD operate failed (detail description please refer to functions)
0x80	Parameter setup succeed
0x81	Parameter setup failed
0x82	Communication timeout
0x83	Card not existing
0x84	Receiving card data error
0x85	Input parameter or input CMD format error
0x87	Unknown error
0x89	input parameter or input CMD format error
0x8A	Block initializing mistake
0x8B	Get wrong serial number when anticollision
0x8C	Password authentication error
0x8f	Input command code not existing
0x90	command not supported by card
0x91	command format mistake

3 Structure Type Declaration

```

typedef struct _DEVICEINFO
{
char strMAC[20]; //MAC address of this device, for example 00.09.F6.01.02.03
char strIP[20]; // IP address of this device for example 10.1.1.1
DWORD dwModel; //the model of converter
DWORD dwVersion; //version number
char strName[20]; //name
BYTE pbUnused[100]; //unused
}DEVICEINFO;

typedef struct _C2000NETSETTING
{
int nNetIndex; //index number, start from 0
int bUseStaticIP; //0: auto IP ; 1: assigned IP
char ipAddr[20]; // the IP address of this device
char ipNetMask[20]; //network mask locating
char ipGateway[20]; //IP address of gateway
char ipDnsServer[20]; //IP address of DNS server
BYTE pbUnused[200]; //unused
}C2000NETSETTING;

typedef struct _C2000SOCKETSETTING
{
int nComIndex; //port number,
//if it's not belong with any port, this value will be negative and specific value reserved for
other explanation
int nSocketIndex; //index of socket, start from 0
int nWorkMode; //0: TCP client ; 1: TCP server ; 2: UDP1 or auto ;
//3: UDP2 ; 4: auto, depends on different product
int nLocalPort; //local port
char szPeerName[64]; //domain name or IP of the peer
int nPeerPort; //the port of peer
int bUseSocket; //whether to use Socket or not, 0: No, 1: Yes
BYTE pbUnused[200]; //unused
}C2000SOCKETSETTING;

typedef struct _C2000COMSETTING
{
int nComIndex; //port number, start from 0
int nBaudrate; //baudrate
int nDatabit; //databit
int nParity; //parity method
int nStopbit; //stop bit
int nFlowCtrl; //flow control method
u_int nIntervalTimeout; //interval timeout
u_int nMaxFrameLength; //Max frame length
int nInceptBytesLength; //length of packet header (not available for yet)
BYTE pbInceptBytes[4]; //bytes of packet header (not available for yet)
int nTermBytesLength; //length of packet trail (not available for yet )
BYTE pbTermBytes[4]; //bytes of packet trail (not available for yet)
int nWorkMode; //0: default 1: 232 2: 485 3: 422
BYTE pbUnused[200]; //unused
}C2000COMSETTING;

typedef struct _C2000NETSTATUS
{
int nNetIndex; //index number, start from 0
char ipAddr[20]; //IP address of this device
char ipNetMask[20]; //network mask locating
char ipGateway[20]; //IP address of gateway
char ipDnsServer[20]; //IP address of DNS server
BYTE pbUnused[200]; //unused
}C2000NETSTATUS;

```

```
typedef struct _C2000SOCKETSTATUS
{
    int nComIndex; //port number,
    //if it's not belong with any port, this value will be negative and specific value reserved for
    other explanation
    int nSocketIndex; //index of socket, start from 0
    int nWorkMode; //0: TCP client; 1: TCP server , 2: UDP or auto
    int nLocalPort; //local port
    char szPeerName[64]; //IP address or domain of the peer be set
    int nPeerPort; //the port of the peer
    int bUseSocket; //whether to use Socket or not 0: no 1: Yes
    int bConnected; //connecting status ,0: not connecting 1: connecting
    char ipCurPeerAddr[20]; //the current ip address of the peer
    int nCurPeerPort; //the current port of the peer
    BYTE pbUnused[200]; //unused
}C2000SOCKETSTATUS;
```

```
typedef struct _C2000COMSTATUS
{
    int nComIndex; //index number
    int nBaudrate; //baudrate
    int nDatabit; //data bit
    int nParity; //parity method
    int nStopbit; //stop bit
    int nFlowCtrlMode; //control mode
    int nFlowCtrlStatus; //control status
    int nIntervalTimeout; //minimum time of sending
    int nMaxFrameLength; //maximum length of frame
    int nWorkMode; //0: unknown 1: 232 2: 485 3: 422
    BYTE pbUnused[200]; //unused
}C2000COMSTATUS;
```


4 Flow of setting network parameter

