



DLL Library Reference Manual

(ISO14443 Type A/B + 15693 Standard)
(TTL, Serial, USB Virtual Interface)

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

Issue 1.0
10/2014

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

Subject to alteration without prior notice
© Copyright iDTRONIC GmbH 2014
Printed in Germany

Contents

Contents	2
API Function Sets.....	4
1.1 System Function.....	4
1.1.1 API_GetSysComm.....	4
1.1.2 API_OpenComm.....	4
1.1.3 API_CloseComm.....	4
1.1.4 API_SetDeviceAddress.....	5
1.1.5 API_SetBaudrate.....	5
1.1.6 API_SetSerNum.....	6
1.1.7 API_GetSerNum.....	6
1.1.8 API_GetVersionNum.....	6
1.1.9 API_ControlLED.....	7
1.1.10 API_ControlBuzzer.....	7
1.2 ISO14443 Type-A Function.....	8
1.2.4 API_MF_Halt.....	8
1.2.5 API_MF_Read.....	8
1.2.6 API_MF_Write.....	9
1.2.7 API_MF_InitVal.....	9
1.2.8 API_MF_Dec.....	10
1.2.9 API_MF_Inc.....	11
1.3 MIFARE Ultralight.....	12
1.3.1 ULRead.....	12
1.3.2 ULWrite.....	12
1.3.3 getULCardSN.....	13
2 ISO14443 Type-B Function.....	13
2.1.1 API_Request_B.....	13
2.1.2 API_Anticoll_B.....	14
2.1.3 API_Attrib_B.....	14
2.1.4 API_RESET_B.....	14
2.2.5 API_TransferCMD_B.....	15
2.2 CPU Card Function.....	15
2.2.1 API_MF_PowerOn.....	15
2.2.2 API_MF_TransferCMD.....	16
2.2.3 API_MF_RST_Antenna.....	16
3 ISO15693 Function.....	17
3.1 API_ISO15693_Inventory.....	17
3.2 API_ISO15693_Read.....	17
3.3 API_ISO15693_Write.....	18
3.4 API_ISO15693_Lock.....	18
35 API_ISO15693_StayQuiet.....	19

3.6	API_ISO15693_Select.....	19
3.7	API_ISO15693_ResetToReady.....	19
3.8	API_ISO15693_WriteAFI.....	20
3.9	API_ISO15693_LockAFI.....	20
3.10	API_ISO15693_WriteDSFID.....	21
3.11	API_ISO15693_LockDSFID.....	21
3.12	API_ISO15693_GetSysInfo.....	21
3.13	API_ISO15693_GetMulSecurity.....	22
3.14	API_ISO15693_TransCmd.....	22
A	APPENDIX.....	23

API Function Sets

1.1 System Function

1.1.1 API_GetSysComm

Function	Get COM port information from system
Prototype	int API_GetSysComm(unsigned char *Buffer)
Description	Input parameter: none Output parameter: Buffer[0]: serial port numbers had been detected in system Buffer[1]: if Buffer[0] > 0, Buffer[1] is kept as port number Buffer[2]: if Buffer[0] > 1, Buffer[2] is kept as port number ... Buffer[N]: if Buffer[0] > N-1, Buffer[2] is kept as port number
Return	return as 0 when succeed, otherwise will return 1

1.1.2 API_OpenComm

Function	Open assigned Com port and set baudrate
Prototype	HANDLE API_OpenComm(int nCom, int nBaudrate)
Description	Input parameter: nCom: assigned Com number nBaudrate: baudrate (9600,19200,38400,57600,115200) Output parameter: none
Return	return with Com handle when succeed, otherwise will be 0

1.1.3 API_CloseComm

Function	Close assigned Com
Prototype	BOOL API_CloseComm(HANDLE commHandle)
Description	Input parameter: commHandle: assigned Com Handle Output parameter: none
Return	return as TRUE when succeed, otherwise will be FALSE

1.1.4 API_SetDeviceAddress

Function	Set reader device address
Prototype	int API_SetDeviceAddress(HANDLE commHandle, int DeviceAddress, unsigned char NewAddress, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: range from 0~255</p> <p>NewAddress: new address, range from 0~255</p> <p>Output parameter:</p> <p>Buffer</p> <p>If succeed, Buffer[0] will be the new address after setted</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

1.1.5 API_SetBaudrate

Function	Set reader's communication baudrate
Prototype	int API_SetBaudrate(HANDLE commHandle, int DeviceAddress, unsigned char NewBaud, unsigned char *Buffer)
Description	<p>Input parameter:</p> <p>commHandle: Com handle</p> <p>DeviceAddress: 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>Other value--9600 bps</p> <p>Output parameter:</p> <p>Buffer</p> <p>If succeed, Buffer[0] will be the new baudrate code after setted</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> <ul style="list-style-type: none">commHandle: Com handleDeviceAddress: range from 0 ~ 255NewValue: new serial number (8byte) <p>Output parameter:</p> <ul style="list-style-type: none">BufferIf succeed, Buffer[0]=0x80If 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

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> <ul style="list-style-type: none">commHandle: Com handleDeviceAddress: range from 0~255 <p>Output parameter:</p> <ul style="list-style-type: none">BufferIf succeed, Buffer[0] is the reader's address, then Buffer[1-8] is the serial numberIf 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

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> <ul style="list-style-type: none">commHandle: Com handleDeviceAddress: range from 0~255

	Output parameter: VersionNum If succeed, VersionNum[0-7] is the version If failed, VersionNum[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

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	Input parameter: commHandle: Com handle DeviceAddress: range from 0~255 freq: cycle count of LED lighting in one cycle (20ms for one cycle) duration: cycle time of LED status(1seconds for one cycle) 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

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	Input parameter: commHandle: Com handle DeviceAddress: range from 0~255 freq: cycle counts of Buzzer making sound in one cycle(100ms for one second) duration: cyclic time of the buzzer status (1 second for one cycle) 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

1.2.4 API_MF_Halt

Function	Operate to halt the reader
Prototype	int API_MF_Halt(HANDLE commHandle, int DeviceAddress)
Description	Input parameter: commHandle: Com handle DeviceAddress: range from 0x00~0xFF Output parameter: none
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

1.2.5 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	Input parameter: commHandle: Com handle DeviceAddress: range from 0x00~0xFF Mode: control reading mode 0x00: Idle mode + Key A 0x01: All mode + Key A 0x02: Idle mode + Key B 0x03: All mode + Key B Blk_add: start address of the block to be read, range from 0x00~0xFE Num_blk: lengths of block to be read, range from 1~4 (Mifare 1k card) Key[0-5]: 6 bytes key Output parameter: Buffer: If succeed, Buffer[0]: return data length Buffer[1-4]: 4 bytes card serial number (from low to high)) Buffer[5-N]: data of card If failed, Buffer[0]: is the status code returned from reader, detail

	definition, please refer to APPENDIX Note: this function enable to read block of this sector only, because each sector has its own unique key
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

1.2.6 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</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

1.2.7 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

1.2.8 API_MF_Dec

Function	E-wallet decrement
Prototype	<pre>int API_MF_Dec(HANDLE commHandle, int DeviceAddress, unsigned char Mode, unsigned char Sec_num, unsigned char *Key, unsigned char *Value, unsigned char *Buffer)</pre>
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
---------------	---

1.2.9 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

1.3 MIFARE Ultralight

1.3.1 ULRead

Function	To read one block of Mifare Ultralight card
Prototype	int ULRead(int handle,int address,byte blk_add,byte[] SN,byte[] buffer);
Description	<p>Read one block of Mifare Ultralight card</p> <p>Input parameter:</p> <p>commHandle — Com handle</p> <p>DeviceAddress — range from 0—255</p> <p>blk_add</p> <p>Start address of block to be read, range from 0--63</p> <p>Output parameter snr</p> <p>buffer</p> <p>If succeed, buffer[0-4]: data returned from card(4 byte)</p> <p>If failed, buffer[0]: 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

1.3.2 ULWrite

Function	To write one block of Mifare Utralight
Prototype	int ULWrite(int handle,int address,byte blk_add,byte[] SN,byte[] buffer);
Description	<p>Input parameter: commHandle —</p> <p>Com handle DeviceAddress — range from 0—255 blk_add</p> <p>Start address of block to be written, range from 0--63</p> <p>SN</p> <p>Reserved</p> <p>buffer</p> <p>Data to be written (4 bytes)</p> <p>Output parameter:</p> <p>buffer</p> <p>If succeed, buffer[0-6]: 7 bytes card UID (low byte in front)</p>

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

1.3.3 getULCardSN

Function	Integrated functions with request, anticollision, select card, etc in one command, to
Prototype	int getULCardSN(int handle,int address,byte[] SN);
Description	<p>Input parameter: commHandle — Com handle DeviceAddress — range from 0—255</p> <p>Output parameter: SN— succeed, return card number 0x00 — one card be detected 0x01 — several cards be detected.(but another card UID detected when invoking again)</p> <p>If failed , SN[0]: is the status returned from reader, detail description please</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

2 ISO14443 Type-B Function

2.1.1 API_Request_B

Function	ISO14443 TypeB card Request
Prototype	int API_Request_B(HANDLE commHandle, int DeviceAddress, unsigned char *Buffer)
Description	<p>Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF</p> <p>Output parameter: Buffer[0]: if success, return data length</p>

	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

2.1.2 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

2.1.3 API_Attrib_B

Function	ISO14443 TypeB card Attrib
Prototype	int API_Attrib_B(HANDLE commHandle, int DeviceAddress, unsigned char *SerialNum, unsigned char *Buffer)
Description	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF SerialNum[0-3]: card serial number (4byte) Output parameter: Buffer[0]: if succeed, 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

2.1.4 API_RESET_B

Function	ISO14443 TypeB card Reset
Prototype	int API_RESET_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

2.2.5 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	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF CmdSize: length of data input Cmd[0-M]: data to be input (CmdSize byte) Output parameter: Buffer[0]: if succeed, Buffer[0]=0x80 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

2.2 CPU Card Function

2.2.1 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	Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF

	<p>Mode: CPU card Poweron mode Idle: 0x26 All: 0x53</p> <p>Cmd: Power-on command parameter, fixed as 0x00</p> <p>Output parameter: Buffer[0]: if succeed, return length of data If failed, return error status, detail definition, please refer to APPENDIX Buffer[1-N]: card UID</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

2.2.2 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: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Mode: CPU card data transfer mode, Mode=0x00 Cmdlength: data length to be transferred Cmd[0-M]: data to be transferred</p> <p>Output parameter: Buffer[0]: if success, return data length If failed, return error status, detail definition, please refer to APPENDIX Buffer[1-N]: data returned</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

2.2.3 API_MF_RST_Antenna

Function	Reset Antenna
Prototype	int API_MF_RST_Antenna(HANDLE commHandle, int DeviceAddress, unsigned char *Buffer)
Description	<p>Input parameter: commHandle: Com port handle DeviceAddress: range from 0x00~0xFF</p> <p>Output parameter: Buffer[0]: if success, Buffer[0]=0x80 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
---------------	---

3 ISO15693 Function

3.1 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> <ul style="list-style-type: none"> commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flag: flags Afi: AFI Datalen: sending data length pData: sending data <p>Output parameter:</p> <ul style="list-style-type: none"> pBuffer[0]: if success, return data length If failed, return error status, detail definition, please refer to APPENDIX pBuffer[1-N]: card information
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

3.2 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> <ul style="list-style-type: none"> commHandle: Com port handle DeviceAddress: range from 0x00~0xFF Flag: flags Blk_add: block number to be read Num_blk: number of blocks to be read uid: card UID

	Output parameter: Buffer[0]: if success, return data length 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

3.3 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

3.4 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
---------------	---

35 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
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

3.6 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
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

3.7 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
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

3.8 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
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

3.9 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
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please

	to APPENDIX
--	-------------

3.10 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	<p>Input parameter:</p> <p>commHandle: Com port handle</p> <p>DeviceAddress: range from 0x00~0xFF</p> <p>Flags: flag</p> <p>DSFID: DSFID</p> <p>uid: card UID</p> <p>Output parameter</p> <p>:</p> <p>Buffer[0]: if success, Buffer[0]=0x80</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

3.11 API_ISO15693_LockDSFID

Function	ISO15693 锁DSFID
Prototype	int API_ISO15693_LockDSFID(HANDLE commHandle, int DeviceAddress, unsigned char Flags, 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>Flags: flags</p> <p>uid: card UID</p> <p>Output parameter</p> <p>:</p> <p>Buffer[0]: if success, Buffer[0]=0x80</p>
Return	Return as 0 when succeed, otherwise will be not 0, detail definition please refer

3.12 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	<p>Input parameter:</p> <p>commHandle: Com port handle</p>

	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

3.13 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

3.14 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

A APPENDIX

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

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