



@MAXX lite

Multifunctional token (smartcard, flash, antenna)

Reference manual

**@MAXX lite (contact smartcard reader,
masstorage, RF-antenna) USB-Stick**

© SCM Microsystems

Oskar-Messter-Strasse, 13

85737 Ismaning

Germany

Phone +49 89 9595 5000 • Fax +49 89 9595 5555

Document history

| Date | Version | Description of change | Responsible person |
|------------|---------|-----------------------|--------------------|
| 30/09/2009 | 0.1 | Creation | Stephan Rasch |
| | | | |
| | | | |
| | | | |

Contact information

<http://www.scmmicro.com/products-services/smart-card-readers-terminals/multifunctional-token.html>

For sales information, please email sales@scmmicro.com

Table of Contents

- 1. Legal information 6
 - 1.1. Disclaimers..... 6
 - 1.2. Licenses 6
 - 1.3. Trademarks 6
- 2. Introduction to the manual 7
 - 2.1. Objective of the manual 7
 - 2.2. Target audience 7
 - 2.3. Product version corresponding to the manual 7
 - 2.4. Definition of various terms and acronyms 7
 - 2.5. References 8
 - 2.6. Conventions 8
- 3. General information about @MAXX lite 10
 - 3.1. @MAXX lite key benefits..... 10
 - 3.2. @MAXX lite key features 10
 - 3.3. @MAXX lite ordering information..... 10
 - 3.4. @MAXX lite customization options 10
 - 3.5. Hardware features and their principle usage 11
 - 3.5.1. Contact Smart Card Reader 12
 - 3.5.2. Internal RF antenna 13
 - 3.5.3. Embedded Flash 14
 - 3.6. Applications 14
 - 3.6.1. General 14
 - 3.6.2. Applications provided by SCM Microsystems..... 16
- 4. @MAXX lite characteristics 17
 - 4.1. @MAXX lite high level architecture..... 17
 - 4.1.1. Block diagram 17
 - 4.2. Quick reference data..... 19
 - 4.2.1. @MAXX lite dimensions 19
 - 4.2.2. LED behavior 19
 - 4.2.3. Other data 19
- 5. Software modules 21
 - 5.1. Installation 21
 - 5.2. Utilities 21
 - 5.3. Driver 21
 - 5.3.1. @MAXX listing 21
 - 5.3.2. Supported operating systems 21
 - 5.4. Firmware 22
 - 5.4.1. CCID transport protocol 22
 - List of CCID bulk messages supported 22
 - List of CCID bulk messages not supported 22

- 6. Commands description 24
 - 6.1. Escape commands for the contact interface 24
 - 6.1.1. Sending escape commands to @MAXX lite 24
 - 6.1.2. Escape command codes 24
 - 6.1.3. CCID_ESC_GETINFO 25
 - 6.1.4. CCID_ESC_SETMODE 25
 - 6.1.5. CCID_ESC_GETMODE 25
 - 6.1.6. CCID_ESC_GET_FW_VERSION 26
 - 6.1.7. CCID_ESC_SET_POWER_ON_RESET_ORDER 26
 - 6.1.8. CCID_ESC_EMV_LOOPBACK 26
 - 6.1.9. CCID_ESC_APDU_TRANSFER 26
 - 6.1.10. CCID_ESC_CLK_FREQUENCY 27
 - 6.1.11. CCID_ESC_GET_SET_ETU 27
 - 6.1.12. CCID_ESC_GET_SET_WAITTIME 27
 - 6.1.13. CCID_ESC_GET_SET_GUARDTIME 28
 - 6.1.14. CCID_ESC_GET_SET_EGT 28
 - 6.1.15. CCID_ESC_GET_SET_ATR_TIMEOUT 28
 - 6.1.16. CCID_ESC_POWER 28
 - 6.1.17. CCID_ESC_ROUGH_TANSFER 29
 - 6.1.18. CCID_ESC_GET_SET_PROTOCOL 29
 - 6.1.19. CCID_ESC_GET_SET_TA1_RFU 29
- 7. Annexes 30
 - 7.1. Annex A – Status words table 30
 - 7.2. Annex B – Sample code using escape commands through Escape IOCTL 30

1. Legal information

1.1. Disclaimers

The content published in this document is believed to be accurate. SCM Microsystems does not, however, provide any representation or warranty regarding the accuracy or completeness of its content and regarding the consequences of the use of information contained herein. If this document has the status "Draft", its content is still under internal review and yet to be formally validated.

SCM Microsystems reserves the right to change the content of this document without prior notice. The content of this document supersedes the content of previous versions of the same document. The document may contain application descriptions and/or source code examples, which are for illustrative purposes only. SCM Microsystems gives no representation or warranty that such descriptions or examples are suitable for the application that the reader may want to use them for.

Should you notice problems with the provided documentation, please provide your feedback to support@scmmicro.com.

1.2. Licenses

If the document contains source code examples, they are provided for illustrative purposes only and subject to the following restrictions:

- You MAY at your own risk use or modify the source code provided in the document in applications you may develop. You MAY distribute those applications ONLY in form of compiled applications.
- You MAY NOT copy or distribute parts of or the entire source code without prior written consent from SCM Microsystems.
- You MAY NOT combine or distribute the source code provided with Open Source Software or with software developed using Open Source Software in a manner that subjects the source code or any portion thereof to any license obligations of such Open Source Software.

If the document contains technical drawings related to SCM Microsystems products, they are provided for documentation purposes only. SCM Microsystems does not grant you any license to its designs.

1.3. Trademarks

MIFARE is a registered trademark of NXP Semiconductors BV.

Windows is a trademark of Microsoft Corporation.

MAC is a trademark of Apple Corporation.

2.Introduction to the manual

2.1. Objective of the manual

This manual provides an overview of the hardware and software features of the @MAXX lite (contact smartcard reader, mass storage, RF-antenna) reader, hereafter referred to as “@MAXX lite”.

This manual describes in details interfaces and supported commands available for developers using @MAXX lite in their applications.

2.2. Target audience

This document describes the technical implementation of @MAXX lite.

The manual targets software developers.

Should you have questions, you may send them to support@scmmicro.com .

2.3. Product version corresponding to the manual

| Item | Version |
|--------------------------------|---------------|
| Hardware | SD-03-M-Ver-F |
| Firmware (SIM) | 2.02 |
| Windows Driver (PC/SC) | 4.41 |
| MAC OS Driver (PC/SC) | 5.0.6 |
| WinCE Driver for SIM interface | 3.14 |
| Linux Driver for SIM Interface | 5.0.2 |

2.4. Definition of various terms and acronyms

| Term | Expansion |
|--------|---|
| APDU | Application Protocol Data Unit |
| ATR | Answer to Reset, defined in ISO/IEC 7816 |
| ATS | Answer to select, defined in ISO/IEC 14443 |
| Byte | Group of 8 bits |
| CCID | Chip Card Interface Device |
| CID | Card Identifier |
| CL | Contactless |
| DFU | Device Firmware Upgrade |
| DR | Divider receive: used to determine the baud rate between the reader to the card |
| DS | Divider send: used to determine the baud rate between the card to the reader |
| LED | Light emitting diode |
| MIFARE | is the NXP Semiconductors (a spin-off company formed out of Philips Semiconductors)-owned trademark of the reputedly most widely installed contactless smartcard |
| NA | Not applicable |
| NAD | Node Address |
| Nibble | Group of 4 bits. 1 digit of the hexadecimal representation of a byte. <i>Example:</i> 0xA3 is represented in binary as (10100011)b. The least significant nibble is 0x3 or (0011)b and the most significant nibble is 0xA or (1010)b |

| | |
|-----------|---|
| PCD | Proximity Coupling Device |
| PC/SC | Personal Computer/Smart Card: software interface to communicate between a PC and a smart card |
| PICC | Proximity Integrated Chip Card |
| PID | Product ID |
| Proximity | Distance coverage in the range of 5cm |
| PUPI | Pseudo unique PICC identifier |
| RFU | Reserved for future use |
| RF | Radio Frequency |
| STCII | Smart card reader controller ASIC from SCM Microsystems |
| USB | Universal Serial Bus |
| VID | Vendor ID |
| (xyz)b | Binary notation of a number x, y, z ∈ {0,1} |
| 0xYY | The byte value YY is represented in hexadecimal |

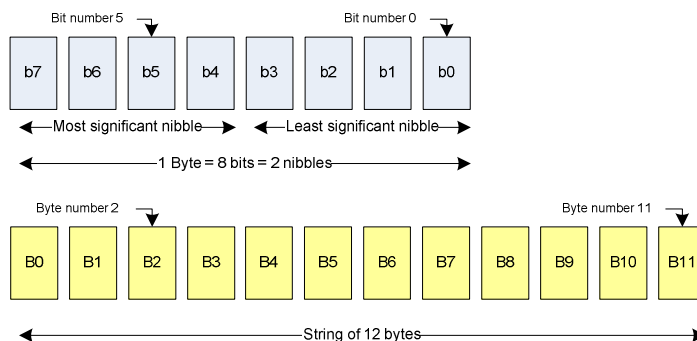
2.5. References

| Doc ref in the manual | Description | Issuer |
|-----------------------|--|-----------------|
| ISO/IEC 7816-3 | Identification cards — Integrated circuit cards — Part 3: Cards with contacts — Electrical interface and transmission protocols | ISO / IEC |
| ISO/IEC 7816-4 | Identification cards - Integrated circuit(s) cards with contacts Part 4: Interindustry commands for interchange ISO/IEC 7816-4: 1995 (E) | ISO / IEC |
| ISO/IEC 14443-3 | Identification cards — Contactless integrated circuit(s) cards — Proximity cards — Part 3: Initialization and anticollision | ISO / IEC |
| ISO/IEC 14443-4 | Identification cards — Contactless integrated circuit(s) cards — Proximity cards Part 4: Transmission protocol ISO/IEC 14443-4:2001(E) | ISO / IEC |
| PC/SC | Interoperability Specification for ICCs and Personal Computer Systems v2.01 | PC/SC Workgroup |
| CCID | Specification for Integrated Circuit(s) Cards Interface Devices 1.1 | USB-IF |
| USB | Universal Serial Bus Specification 2.0 | USB-IF |

2.6. Conventions

Bits are represented by lower case 'b' where followed by a numbering digit.

Bytes are represented by upper case 'B' where followed by a numbering digit.



Example:

163 decimal number is represented

- in hexadecimal as 0xA3
- in binary as (10100011)_b

The least significant nibble of 0xA3 is

- 0x3 in hexadecimal
- (0011)_b in binary

The most significant nibble of =xA3 is

- 0xA in hexadecimal
- (1010)_b in binary

3. General information about @MAXX lite


3.1. @MAXX lite key benefits

The @MAXX lite is an all-in-one device and belongs to SCM's @MAXX token family. Designed as a USB-stick the @MAXX lite is portable and can be connected to stationary PCs as well as Laptops. In addition to the normal smartcard reader the internal RF antenna converts the normal USB-stick together with a dual interface smartcard to a portable mobile passive NFC token. The typical application field of the @MAXX lite is the combination of physical and logical access within one device.

3.2. @MAXX lite key features

- 8-pin, ISO7816 compliant contact smart card reader for ID-000 smartcards
 - PC/SC v2.0 compliant
 - CCID compatible
- Unique serial number for the Mass Storage interface
- Can be plugged into any USB slot on a PC without having to re-install the driver.
- embedded flash memory 2GB (other capacities on request)
- internal passive RF-antenna connected to C4/C8 of the smartcard connector

3.3. @MAXX lite ordering information

| Item | Part number | |
|-------------|-------------|--|
| @MAXX lite | 905110 |  |
| Contact SDK | 905129 | SDK for Contact SmartCard |

3.4. @MAXX lite customization options

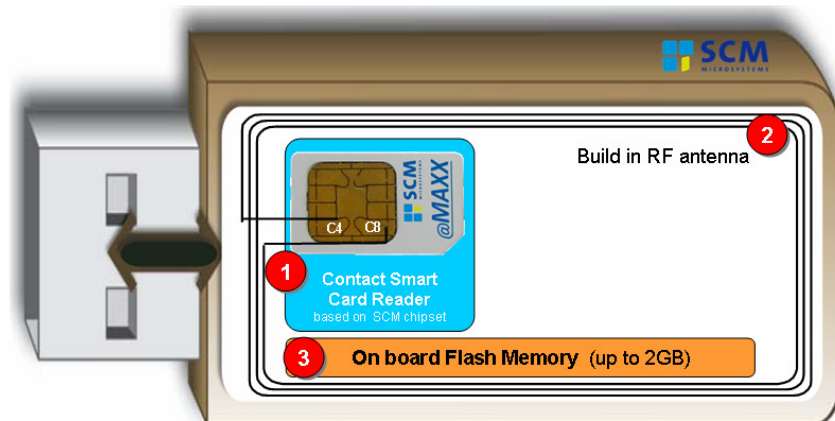
Upon request, SCM can customize:

- The color of the casing
- The logo
- The product label
- The USB strings for the Mass Storage Device
- Flash memory capacity

Terms and conditions apply, please contact your local SCM representative or send an email to sales@scmmicro.com.

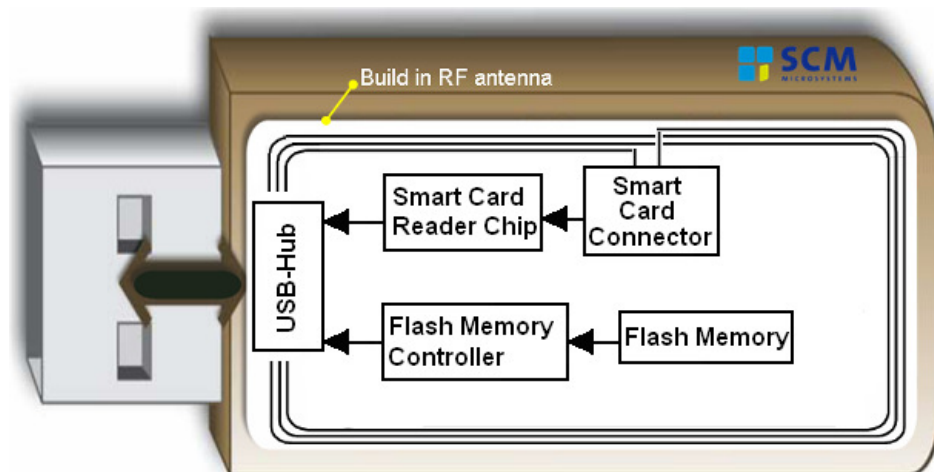
3.5. Hardware features and their principle usage

The @MAXX lite is a multifunctional token device, which can be used within a big application field. It can be used for one application or for a combination of several applications. In the following paragraphs the usage recommendations are outlined to ensure best user experience.



1. Contact Smart Card Reader for dual interface Smart Cards with antenna connection on C4 / C8
2. Internal RF antenna
3. On board flash memory

@MAXX lite schematics:



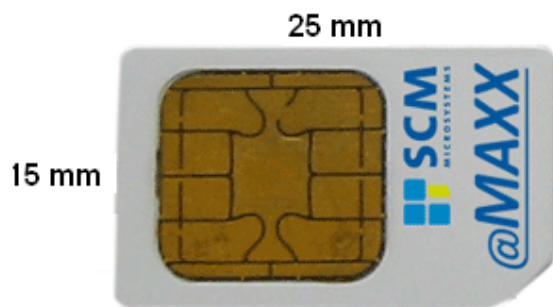
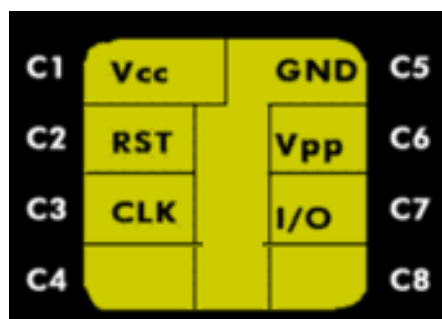
3.5.1. Contact Smart Card Reader

The contact smart card reader is an 8-pin, ISO7816 compliant contact smart card reader for ID-000 smartcards. Its contacts C4 and C8 are connected with the internal RF antenna. Thus it is possible to use beside normal smartcards (microprocessor and memory cards) also dual interface smartcards with additional contacts on C4 / C8. Memory cards can be accessed using SCM's Memory Card DLL.

Note: Memory cards which supports C4 /C8 will not be able to work as these pins are used for the RF antenna.

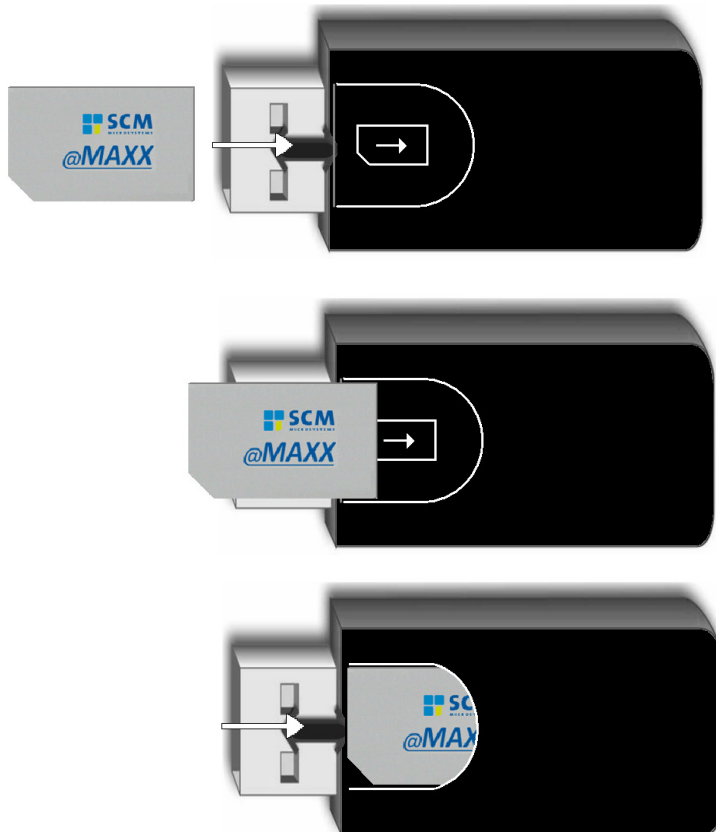
Within a Windows® environment the smart card reader can be used with the PC/SC driver or with a preinstalled USB-CCID driver.

Windows® Vista and future Windows® operation systems have the USB-CCID driver as standard driver already installed. If Windows® XP is used as operating system, the USB-CCID driver is available as optional update.



| Contact | Designation | Use |
|---------|----------------|--|
| C1 | Vcc | Power connection through which operating power is supplied to the microprocessor chip in the card |
| C2 | RST | Reset line through which the IFD can signal to the smart card's microprocessor chip to initiate its reset sequence of instructions |
| C3 | CLK | Clock signal line through which a clock signal can be provided to the microprocessor chip. This line controls the operation speed and provides a common framework for data communication between the IFD and the ICC |
| C4 | Antena | RF antenna |
| C5 | GND | Ground line providing common electrical ground between the IFD and the ICC |
| C6 | Vpp | Programming power connection used to program EEPROM of first generation ICCs. |
| C7 | I/O | Input/output line that provides a half-duplex communication channel between the reader and the smart card |
| C8 | Antenna | RF antenna |

Smart Card insertion:



3.5.2. Internal RF antenna

The internal RF antenna is connected with the pins C4 / C8 of the smartcard reader for the usage of dual interface cards.

When a dual interface card is inserted in the @MAXX and the @MAXX is put in the magnetic field of any contactless reader, the internal RF antenna couples with the reader and an induction current appears in the antenna thus providing power to the integrated circuit. The generated current is proportional to the magnetic flux going through the antenna of @MAXX reader

The carrier frequency of the magnetic field is used as a fundamental clock signal for the communication between the reader and the card. It is also used as a fundamental clock input for the integrated circuit microprocessor to function.

To send data to the user token the reader modulates the amplitude of the field. There are several amplitude modulation and data encoding rules defined in ISO/IEC 14443. The reader should be ISO 14443 compliant reader.

To answer to the reader, the integrated circuit card of the @MAXX modulates its way of loading (impedance) the field generated by the reader. Here also further details can be found in ISO/IEC 14443.

The best communication between the @MAXX (with inserted dual interface card) and a contactless reader can be established if the @MAXX is placed on the contactless reader with the @MAXX bottom side in direction to the contactless reader. The bottom side of the @MAXX

is the side where the label and the smartcard insertion slot can be found. Depending on the used contactless reader and the used dual interface card, the @MAXX lite establishes a read out distance between 2cm – 3cm. A little bit bigger antenna within the @MAXX pime A results to a bigger read out distance up to 3.5cm.

The communication between the reader and the @MAXX with dual interface card is sensitive to the presence of material or objects interfering with the magnetic field generated by the reader.

The presence of conductive materials like metal in the vicinity of the reader and the user token can severally degrade the communication and even make it impossible. The magnetic field of the reader generates Eddy or Foucault's currents in the conductive materials; the field is literally absorbed by that kind of material.

The presence of multiple @MAXXs in the field of contactless readers also interferes with the communication. When several @MAXXs are in the field of a contactless reader, load of the field increases which implies that less energy is available for each of them and that the system is detuned.

The communication between a contactless reader and the @MAXX is sensitive to the geometry of the system {reader, @MAXX}. Parameters like the geometry and specially the relative size of the reader and @MAXX antennas directly influence the inductive coupling and therefore the communication.

3.5.3. Embedded Flash

Also the embedded flash memory is connected with the internal USB-hub over the flash controller. The actual hardware design is able to support embedded flash memory up to 4GB.

Several different partitions can be created on the flash memory:

- CD-Rom partition (with Auto-run function)
- Secure partition (PIN protected)

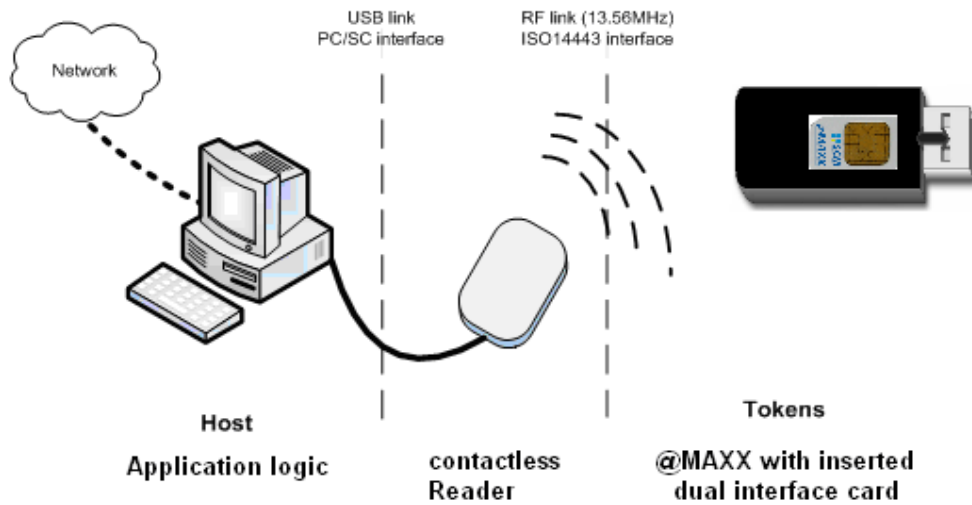
It is also possible to upload data on the flash and to create different partitions on production level.

3.6. Applications

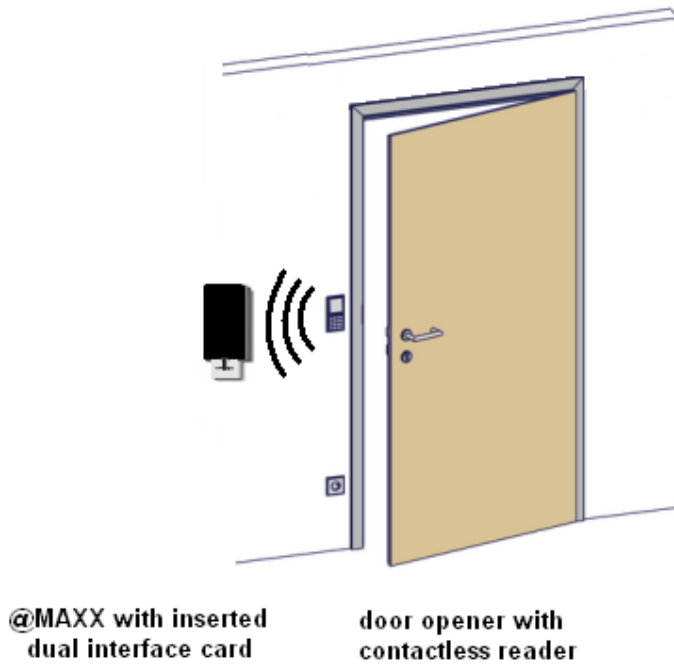
3.6.1. General

@MAXX lite is a multifunctional device, designed to interface a personal computer host supporting PC/SC interface or CCID interface with ISO7816 smartcards in ID-000 format. Further the device can be used as a passive NFC tag by using a dual interface smartcard. And the @MAXX can be used as USB Mass Storage device and Micro SD-card reader.

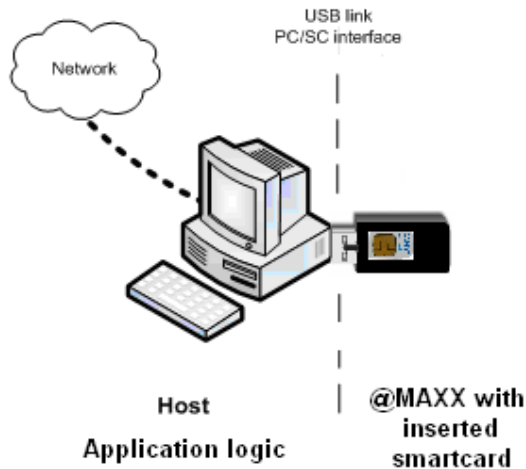
The following graphic shows the case study and the needed infrastructure if the @MAXX is used as a passive NFC token.



A similar use case is the door opener. In this case the host is not seen. The contactless reader is installed beside the door.



@MAXX lite itself handles the communication protocol but not the application related to the token. The application-specific logic has to be implemented by software developers on the host.



3.6.2. Applications provided by SCM Microsystems

SCM Microsystems does not provide payment or transport applications.

SCM Microsystems provides a few applications for development and evaluation purposes that can function with @MAXX lite. There are many tools provided; here are two of them:

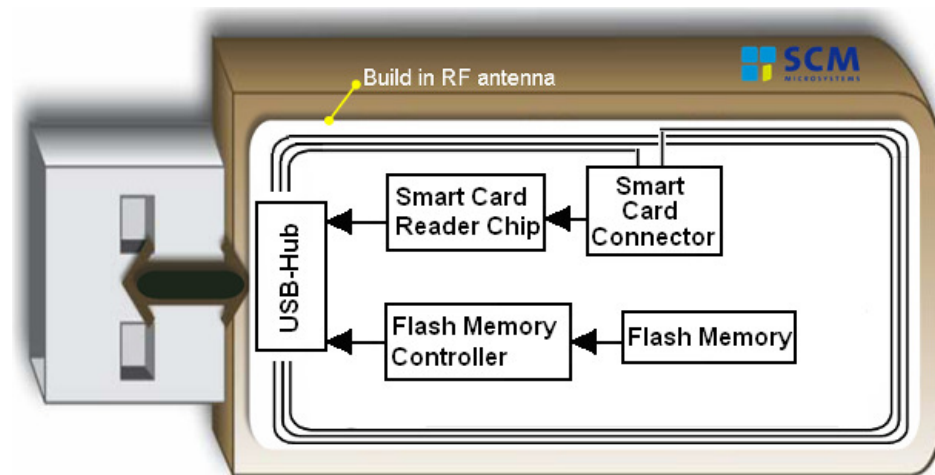
- The NFC forum tag reader/writer is a standalone application that enables the user to read and write NFC forum compliant records into NFC forum compatible tags. It is an easy to use tool to configure rapidly NFC forum tag demonstrations. Note: @MAXX lite supports NFC forum tag type 2 and 4, only.
- Smart card commander version 1.1 provides NFC forum record parsing functionality of NDEF records in XML format as well as scripting functionality which can be very useful for developers to develop and debug their applications. This tool can be used for both the contact and the contactless interfaces of @MAXX lite

4.@MAXX lite characteristics

4.1. @MAXX lite high level architecture

4.1.1. Block diagram

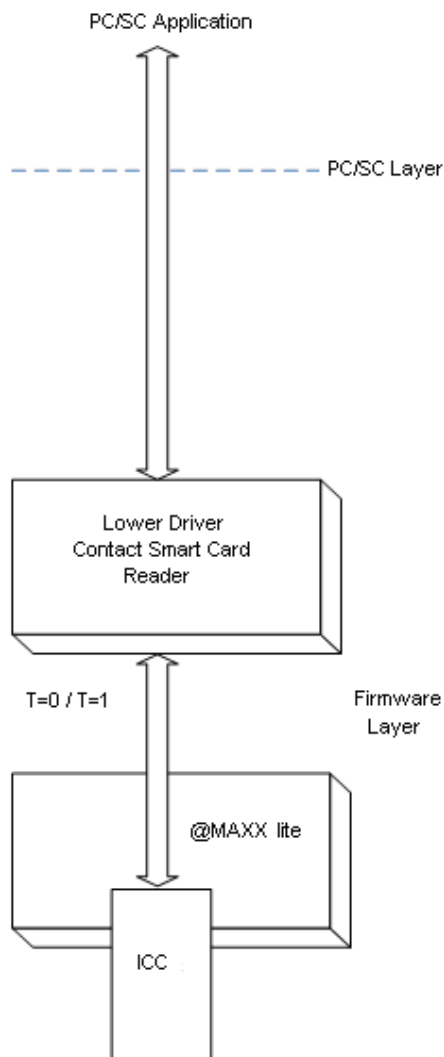
The link between @MAXX lite and the host to which it is connected is the USB interface providing both the power and the communication channel.



@MAXX lite contains the SCM Mask ROM Controller for the SIM Interface. The Mask ROM firmware can handle all the ISO7816 contact protocol and the PC/SC communication protocol with the host.

Software architecture

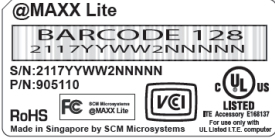
Applications can interface with the driver directly through the PC/SC interface.



The @MAXX lite driver implements PC/SC v2.0 API towards upper layers and full CCID for the contact slot.

4.2. Quick reference data

4.2.1. @MAXX lite dimensions

| Item | Characteristic | Value |
|------------|---------------------|--|
| @MAXX lite | Weight | 13,5 Grams |
| | External dimensions | L 71,0 mm × W 26,0mm × H 10,4mm |
| | USB connector | USB type A connector |
| | Default color | Black rubberized |
| | Default label |  |

Drawing with dimensions of the @MAXX lite and accessories can be found in annex.

4.2.2. LED behavior

@MAXX lite is equipped with a red LED

| @MAXX lite | LED Indication (red) |
|---|----------------------|
| Just after plug-in (with drivers already installed) | flashing |
| @MAXX connected (Idle State) | ON |
| Reader / card errors | OFF |
| Read or write on smartcard, flash storage | Flashing |
| Suspend State | OFF |

4.2.3. Other data

4.2.3.1. General

| Parameter | Value/Description |
|--------------------------------|--|
| Clock of the device controller | 24 MHz |
| API | PC/SC 2.0, CCID |
| Operating temperature range | -20°C to 60°C |
| Operating humidity range | Up to 95%RH non condensing |
| Certifications | CE FCC VCCI WEEE RoHS RoHS green (on request without rubberized casing) |

4.2.3.2. USB

| Parameter | Value/Description |
|----------------------------|--|
| Electrical Characteristics | High bus powered (@MAXX draws power from USB bus) Voltage: 5V Max. Current : 500mA |
| USB specification | USB 2.0 Device |
| USB Speed | High Speed Device (480Mbit/s), Tolerance +- 240kbit/s |
| PID | 0417 (Flash) , 5116 (SIM) and 03F3 (Hub) |
| VID | 04E6 |

4.2.3.3. Contact interface

| Parameter | Value/Description |
|----------------------------------|---|
| Smart card operating frequency | 4.8Mhz |
| Maximum supported card baud-rate | 500Kbps |
| Cards supported | Class A, Class B, and Class C smart cards Synchronous smart cards |
| ISO-7816 compliant | Yes |
| EMV'2000 compliant | Not applicable (F/w. is EMV ready, but isn't applicable for this product) |
| CT-API compliant | Support Available |
| Number of slots | Single smart card slot for smart cards ID-000 |
| Ejection mechanism | Manual |

5. Software modules

5.1. Installation

SCM provides an installer for Windows.

The installer can be used to install the driver as well as some utilities.

5.2. Utilities

The following utilities are available:

- A tool for testing the installation of the PC/SC driver
- A tool for testing the resource manager
- A tool called *PC/SC Diag* capable of providing basic information about the reader and a card through PC/SC stack

5.3. Driver

5.3.1. @MAXX listing

@MAXX lite is listed

by PC/SC applications as

- SCM Microsystems Inc. SCR33x USB Smart Card Reader 0

USB Mass Storage Device:

- SCMMICRO @MAXX Lite FlashDrive

5.3.2. Supported operating systems

Operating systems supported by the driver:

- Windows XP (32 & 64 bit)
- Windows Vista (32 & 64 bit)
- Linux (Kernel ver. 2.6.x) - 32bit
- MAC OS 10.5 (Leopard) – 32bit
- WinCE (32bit)



5.4. Firmware

5.4.1. CCID transport protocol

@MAXX lite implements a transport protocol that is compliant with USB Device Class: *Smart Card CCID Specification for Integrated Circuit(s) Cards Interface Devices Revision 1.10* for the contact smart card interface and CCID-like transport protocol for the contactless interface.

This paragraph describes the CCID specification features that are implemented and those that are not implemented.

5.4.1.1. CCID class requests supported

- Abort

5.4.1.2. CCID bulk messages supported

The following section provides information on the list of CCID bulk messages (10-byte header followed by the message specific data) that have been implemented.

List of CCID bulk messages supported

- PC_to_RDR_IccPowerOn
- PC_to_RDR_IccPowerOff
- PC_to_RDR_GetSlotStatus
- PC_to_RDR_XfrBlock
- PC_to_RDR_GetParameters
- PC_to_RDR_SetParameters
- PC_to_RDR_Escape
- PC_to_RDR_Abort
- PC_to_RDR_SetDataRateAndClockFrequency
- PC_to_RDR_T0APDU
- PC_to_RDR_ResetParameters
- PC_to_RDR_IccClock

List of CCID bulk messages not supported

- PC_to_RDR_Secure
- PC_to_RDR_Mechanical

5.4.1.3. CCID Error Codes

Extensive error codes are reported on many conditions during all CCID responses. Most of the error messages are reported by the CCID appropriately. Some of the main error codes for the contact interface are:

- HW_ERROR
- XFR_PARITY_ERROR
- BAD_ATR_TS
- BAD_ATR_TCK
- ICC_MUTE

The following sub-sections discuss when and why these error codes are returned:

5.4.1.3.1. HW_ERROR

This error code is returned when a hardware short circuit condition is detected, during application of power to the card or if any other internal hardware error is detected. This error code has been defined in the error code table 6.2-2 of the CCID specification.

5.4.1.3.2. ICC_MUTE

This error code is returned when the card does not respond until the reader time out occurs. This error will be reported in the response to PC_to_RDR_XfrBlock message and PC_to_RDR_IccPowerOn messages. This error code has been defined in the error code table 6.2-2 of the CCID specification.

6.Commands description

6.1. Escape commands for the contact interface

6.1.1. Sending escape commands to @MAXX lite

A developer can use the following method to send escape commands to @MAXX lite for the contact interface

- SCardControl method defined in PC/SC API

6.1.2. Escape command codes

Escape commands can be used by an application to configure @MAXX lite to function in a mode that is not its default configured mode or to get specific information. To put the @MAXX lite back into its default mode, either the @MAXX lite has to be unplugged and plugged again or the application can send again the same escape command.

The following escape commands are supported by @MAXX lite for the contact interface

| S.No. | Escape message ID | Value |
|-------|-----------------------------------|-------|
| 1 | CCID_ESC_GETINFO | 0x00 |
| 2 | CCID_ESC_SETMODE | 0x01 |
| 3 | CCID_ESC_GETMODE | 0x02 |
| 4 | CCID_ESC_GET_FW_VERSION | 0x03 |
| 5 | CCID_ESC_SET_POWER_ON_RESET_ORDER | 0x04 |
| 6 | CCID_ESC_EMV_LOOPBACK | 0x05 |
| 7 | CCID_ESC_APDU_TRANSFER | 0x08 |
| 8 | CCID_ESC_CLK_FREQUENCY | 0x1F |
| 9 | CCID_ESC_GET_SET_ETU | 0x80 |
| 10 | CCID_ESC_GET_SET_WAITTIME | 0x81 |
| 11 | CCID_ESC_GET_SET_GUARDTIME | 0x82 |
| 12 | CCID_ESC_GET_SET_EGT | 0x83 |
| 13 | CCID_ESC_GET_SET_ATR_TIMEOUT | 0x84 |
| 14 | CCID_ESC_POWER | 0xC0 |
| 15 | CCID_ESC_ROUGH_TANSFER | 0xC1 |
| 16 | CCID_ESC_GET_SET_PROTOCOL | 0xC2 |
| 17 | CCID_ESC_GET_SET_TA1_RFU | 0xC3 |

6.1.3. CCID_ESC_GETINFO

This escape message ID gets static reader specific information from firmware to the host. Information includes the major and minor version, capabilities of the reader etc. The first byte of input buffer shall have just one byte that will contain this escape function's value. The output buffer shall point to an application allocated SCARD_READER_GETINFO_PARAMS structure mentioned below.

```
typedef struct _SCARD_READER_GETINFO_PARAMS
{
    uint8      MajorVersion;
    uint8      MinorVersion;
    uint8      SupportedModes;
    uint16     SupportedProtocols;
    uint16     InputDevice;
    uint8      Personality;
    uint32     Serial;
    uint8      MaxSlots;
} SCARD_READER_GETINFO_PARAMS;
```

6.1.4. CCID_ESC_SETMODE

This escape message ID sets the current mode of the reader. Applications may call this function, to set the desired mode. Typically, this call is used to switch between the EMV and ISO7816 operation. The first byte of the input buffer will contain the escape function value and the second one will contain the value for the desired mode of operation. The output buffer field shall be NULL.

Following table gives the value of modes as interpreted by firmware

| S.No. | Mode | Value | Remarks |
|-------|------|-------|---------------|
| 1 | ISO | 0x02 | ISO 7816 mode |
| 2 | EMV | 0x04 | EMV |

6.1.5. CCID_ESC_GETMODE

This escape message ID retrieves the current mode in which the reader. The input buffer shall contain the escape function value. The output buffer shall point to a BYTE buffer.

Following table gives the value of modes as interpreted by firmware

| S.No. | Mode | Value | Remarks |
|-------|------|-------|---------------|
| 1 | ISO | 0x02 | ISO 7816 mode |
| 2 | EMV | 0x04 | EMV |

6.1.6. CCID_ESC_GET_FW_VERSION

This function code shall be used by the application / driver to retrieve the current firmware revision of the reader.

The input buffer shall contain the escape code. The reader shall return a WORD parameter containing the firmware revision.

```
typedef struct _FW_REV
{
    uint8  byFirmwareMajor;
    uint8  byFirmwareMinor;
} FW_REV;
```

6.1.7. CCID_ESC_SET_POWER_ON_RESET_ORDER

This code shall be used by the application / driver to change the smart card power-on sequence i.e. it shall direct the reader to start the card reset by applying by “Class A” voltage first and then retry sequentially with the other classes.

The input buffer shall contain the escape code followed by the RESET SEQUENCE data. The output buffer shall point to a byte and shall return the current RESET SEQUENCE selected.

| S.No. | Mode | Value | Remarks |
|-------|---------|-------|--|
| 1 | Class C | 0x00 | Starts with Class C voltage. This is the default mode of the reader. |
| 2 | Class A | 0x01 | Starts with Class A voltage |
| 3 | QUERY | 0xFF | This is used to retrieve the current mode of operation. |

6.1.8. CCID_ESC_EMV_LOOPBACK

This escape message ID lets the host force the library to perform an EMV Loop-back application.

The input buffer shall contain the escape function value. The output buffer field shall be NULL.

6.1.9. CCID_ESC_APDU_TRANSFER

This escape message ID exchanges an APDU with the smart card.

The input buffer shall contain the escape function value, followed by the APDU. The output buffer field shall point to user allocated buffer to the maximum size of 259 bytes.

The maximum number of bytes that can be received / sent is given below.

Transmit:

Case 1, 2, 3 APDU: Max of 256 bytes per APDU

Case 4 APDU: Max of 255 bytes per APDU

Receive:

Max of 259 bytes per APDU

6.1.10. CCID_ESC_CLK_FREQUENCY

In case when an application wants to change the clock frequency, this escape ID is used to inform the reader about the change in clock. The first byte of the input buffer will contain the escape function value; the next byte will contain the clock divisor value. The output buffer field shall be NULL.

The possible clock divisor values and hence the clock frequency is given in the table below

| CLOCK DIVISOR VALUE | CLK FREQUENCY (in MHz) | |
|---------------------|------------------------|-------|
| | USB | RS232 |
| 6 | 4 | 3.686 |
| 4 | 6 | 5.53 |
| 3 | 8 | 7.37 |
| 2 | 12 | 11.06 |

6.1.11. CCID_ESC_GET_SET_ETU

This function code shall be used by the reader to get and set the current ETU. The ETU is specified in terms of smart card clock cycles.

The input buffer shall point to the escape code followed by a DWORD specifying the value to be set. The output buffer shall point to NULL.

6.1.12. CCID_ESC_GET_SET_WAITTIME

This function code shall be used to set and get the character / block waiting time of the reader. The wait time is specified in terms of smart card clock cycles.

The buffer shall point to the escape code followed by the wait time structure.

```
typedef struct _WAIT_TIME
{
    uint8  byGetSetIdentifier;
    uint8  byWaitTimeIdentifier;
    uint32 dwWaitTime;
} WAIT_TIME;
```

6.1.13. CCID_ESC_GET_SET_GUARDTIME

This function code shall be used to set and get the character / block guard time of the reader. The guard time is specified in terms of etus.

The buffer shall point to the escape code followed by the guard time structure.

```
typedef struct _GUARD_TIME
{
    uint8  byGetSetIdentifier;
    uint8  byGuardTimeIdentifier;
    uint32 dwGuardTime;
} GUARD_TIME;
```

6.1.14. CCID_ESC_GET_SET_EGT

This function code shall be used to set and get the extra guard time of the reader. The guard time is specified in terms of etus.

The buffer shall point to the escape code followed by a structure given below.

```
typedef struct _EGT
{
    uint8  byGetSetIdentifier;
    uint32 dwEGT;
} EGT;
```

6.1.15. CCID_ESC_GET_SET_ATR_TIMEOUT

This function is used to change the delay (in ms) between the power-up attempts in the class-A, class-B, and class-C sequence;

The buffer shall point to the escape code followed by a BYTE containing the timeout value.

6.1.16. CCID_ESC_POWER

This function code shall be used to change the VCC level for the reader. The input buffer shall point to the escape code followed by a WORD containing VCC value. The ICC will be reset to the value given in the buffer.

The supported smart card voltages are:

```
#define Vcc_AUTOMATIC    0x00
#define Vcc_5V          0x01
#define Vcc_3V          0x02
#define Vcc_1_8V        0x03
```

6.1.17. CCID_ESC_ROUGH_TANSFER

This command is used to perform raw exchange of data with the card. The input buffer for this command will contain the escape function value, low byte of the send length, high byte of the send length, low byte of the expected length, high byte of the expected length and the command. The output buffer field shall point to user allocated buffer.

6.1.18. CCID_ESC_GET_SET_PROTOCOL

This function code shall be used to set and get the transmission protocol for the reader.

The buffer shall point to the escape code followed by a BYTE.

The supported protocols are:

```
#define PROTOCOL_T0          0x00
#define PROTOCOL_T1          0x01
#define PROTOCOL_UNDEFINED  0xFF
```

6.1.19. CCID_ESC_GET_SET_TA1_RFU

Since the reader is compliant to ISO 7816-3 (1997), it will reject cards having a TA1 value which is RFU in respect to this version of the specification. Using this escape function the application can introduce such a card to the reader. If a card having this particular value of TA1 in the ATR is detected, it will be accepted by the reader and the maximum communication speed (i.e. 115200 bps) will be applied. The buffer shall point to the escape code followed by a BYTE.

7. Annexes

7.1. Annex A – Status words table

| SW1 | SW2 | Description |
|------|------|--------------------------|
| 0x90 | 0x00 | NO ERROR |
| 0x67 | 0x00 | LENGTH INCORRECT |
| 0x6D | 0x00 | INVALID INSTRUCTION BYTE |
| 0x6E | 0x00 | CLASS NOT SUPPORTED |
| 0x6F | 0x00 | UNKNOWN COMMAND |
| 0x63 | 0x00 | NO INFORMATION GIVEN |
| 0x65 | 0x81 | MEMORY FAILURE |
| 0x68 | 0x00 | CLASS BYTE INCORRECT |
| 0x6A | 0x81 | FUNCTION NOT SUPPORTED |
| 0x6B | 0x00 | WRONG PARAMETER P1-P2 |

7.2. Annex B – Sample code using escape commands through Escape IOCTL

File Name : T_hbr.H

```

#ifdef __cplusplus
extern "C" {
#endif

#define IOCTL_CCID_ESCAPE                SCARD_CTL_CODE (0xDAC)

#define CCID_GET_848KBPS_STATUS          0xFF9D
#define CCID_SET_848KBPS_ON              0x019D
#define CCID_SET_848KBPS_OFF             0x009D

#define MINTIMEOUT                        300

#ifdef __cplusplus
}
#endif

```

File Name : T_hbr.CPP

```
#include <windows.h>
```

```
#include <winbase.h>
#include <stdio.h>
#include <conio.h>
#include "winscard.h"
#include "winerror.h"
#include "T_hbr.H"

VOID main(VOID)
{

    SCARDCONTEXT          ContextHandle;
    SCARDHANDLE           CardHandle;
    BYTE                  OutByte;
    WORD                  InWord,i;
    DWORD                 ActiveProtocol;          /* ICC protocol */
    ULONG                 InBufLen,ResLen;
    ULONG                 ret;
    SCARD_READERSTATE     Reader[1];

    // please add the name of the used reader here or use SCardListReaders
    // to find the right reader name
    char *ReaderName[] = {"SCM Microsystems Inc. @MAXX lite Contactless Reader 0",
                          NULL};

    /*****
    *****/

    ContextHandle = -1;

    ret = SCardEstablishContext(SCARD_SCOPE_USER, NULL, NULL, &ContextHandle);

    if (ret == SCARD_S_SUCCESS)
    {
        ret = SCardConnect( ContextHandle,
                           ReaderName[0],
                           SCARD_SHARE_SHARED,
                           SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
                           &CardHandle,
                           &ActiveProtocol);

        if (ret == SCARD_S_SUCCESS)
        {
```

```
/* get actual 848kbps status: ON/OFF */

InBufLen = 2;
InWord = CCID_GET_848KBPS_STATUS;
ret = SCardControl (CardHandle,
                    IOCTL_CCID_ESCAPE,
                    &InWord,
                    InBufLen,
                    &OutByte,
                    1,
                    &ResLen);

printf ("\n Get 848kbps status: %lx: %.2x", ret, OutByte);

Reader[0].dwCurrentState = SCARD_STATE_UNAWARE;
Reader[0].dwEventState = SCARD_STATE_UNAWARE;
Reader[0].szReader = ReaderName[0];

ret = SCardGetStatusChange( ContextHandle,
                            MINTIMEOUT,
                            Reader,
                            1);

printf ("\nATR: ");
for (i=0; i<Reader->cbAtr; i++)
{
    printf (".2x ", Reader->rgbAtr[i]);
}
printf ("\n-----\n");

/* enable 848KBPS: ON */

printf ("\nEnable 848kbps ");
InBufLen = 2;
InWord = CCID_SET_848KBPS_ON;
ret = SCardControl (CardHandle,
                    IOCTL_CCID_ESCAPE,
                    &InWord,
                    InBufLen,
                    &OutByte,
                    1,
                    &ResLen);

ret = SCardDisconnect (CardHandle, SCARD_RESET_CARD);
```



```

ret = SCardConnect (ContextHandle,
                    ReaderName[0],
                    SCARD_SHARE_SHARED,
                    SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
                    &CardHandle,
                    &ActiveProtocol);

/* get actual 848KBPS status: ON/OFF */

InBufLen = 2;
InWord = CCID_GET_848KBPS_STATUS;
ret = SCardControl (CardHandle,
                    IOCTL_CCID_ESCAPE,
                    &InWord,
                    InBufLen,
                    &OutByte,
                    1,
                    &ResLen);

printf ("\n Get 848kbps status: %lx: %.2x", ret, OutByte);

Reader[0].dwCurrentState = SCARD_STATE_UNAWARE;
Reader[0].dwEventState = SCARD_STATE_UNAWARE;
Reader[0].szReader = ReaderName[0];

ret = SCardGetStatusChange (ContextHandle,
                            MINTIMEOUT,
                            Reader,
                            1);

printf ("\nATR: ");
for (i=0; i<Reader->cbAtr; i++)
{
    printf ("%2x ", Reader->rgbAtr[i]);
}
printf ("\n-----\n");

/* Disable 848Kbps: OFF */
printf ("\nDisable 848KBPS ");
InBufLen = 2;
InWord = CCID_SET_848KBPS_OFF;
ret = SCardControl(CardHandle, IOCTL_CCID_ESCAPE,
                    &InWord, InBufLen,
                    &OutByte, 1, &ResLen);

```

```
ret = SCardDisconnect(CardHandle, SCARD_RESET_CARD);
ret = SCardConnect(ContextHandle,
                    ReaderName[0],
                    SCARD_SHARE_SHARED,
                    SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1,
                    &CardHandle,
                    &ActiveProtocol);

/* get actual 848KBPS status: ON/OFF */
InBufLen = 2;
InWord = CCID_GET_848KBPS_STATUS;
ret = SCardControl(CardHandle, IOCTL_CCID_ESCAPE,
                  &InWord, InBufLen,
                  &OutByte, 1, &ResLen);
printf ("\n Get 848KBPS status: %lx: %.2x", ret, OutByte);

Reader[0].dwCurrentState = SCARD_STATE_UNAWARE;
Reader[0].dwEventState = SCARD_STATE_UNAWARE;
Reader[0].szReader = ReaderName[0];
ret = SCardGetStatusChange(ContextHandle, MINTIMEOUT, Reader, 1);
printf ("\nATR: ");
for (i=0; i<Reader->cbAtr; i++)
{
    printf (".%.2x ", Reader->rgbAtr[i]);
}
printf ("\n-----\n");

ret = SCardDisconnect(CardHandle, SCARD_RESET_CARD);
}
else
{
    printf("\n SCardConnect failed with 0x%.8lX",ret);
}
ret = SCardReleaseContext(ContextHandle);
}
else
{
    printf("\n SCardEstablishContext failed with %.8lX",ret);
}

printf("\npress any key to close the test tool\n");
getch();
}
```