SCM Microsystems

Reference Manual – version 1.01

SDI011

Dual interface (contactless and contact) stationary reader

Reference manual

SDI011 Dual Interface (Contactless and Contact) Stationary Reader

© 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
08/09/2010	1.0	Initial Version
11/10/2010	1.01	Add FCC warning Typo corrections

Contact information

http://www.scmmicro.com/products-services/smart-card-readers-terminals/contactless-dual-interface-readers.html

For sales information, please email sales@scmmicro.com

Table of Contents

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. FCC

1.2.1. Section 15.21 Information to user

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment

1.2.2. Section 15.105 (b)

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

--Reorient or relocate the receiving antenna.

--Increase the separation between the equipment and receiver.

--Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

--Consult the dealer or an experienced radio/TV technician for help.

1.3. 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.

1.4. Trademarks

MIFARE is a registered trademark of NXP Semiconductors BV.

Windows is a trademark of Microsoft 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 SDI011 dual interface (contactless and contact) reader, hereafter referred to as "SDI011".

This manual describes in details interfaces and supported commands available for developers using SDI011 in their applications.

2.2. Target audience

This document describes the technical implementation of SDI011.

The manual targets software developers. It assumes knowledge about 13.56 MHz contactless technologies like ISO/IEC 14443 and commonly used engineering terms.

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

2.3. Product version corresponding to the manual

Item	Version
Hardware	1.0
Firmware	7.36
Windows Contact Driver	5.19
Windows Contactless Driver	5.20
MAC driver	5.0.18
LINUX Driver	5.0.18

2.4. Definition of various terms and acronyms

Term	Expansion
APDU	Application Protocol Data Unit
ATR	Answer to Reset, defined in ISO7816
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	The ISO14443 Type A with extensions for security (NXP)
NA	Not applicable
NAD	Node Address
Nibble	Group of 4 bits. 1 digit of the hexadecimal representation of a byte. *Example:* 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 till ~10 cm.
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 $\in\{0,1\}$
0xYY	The byte value YY is represented in hexadecimal

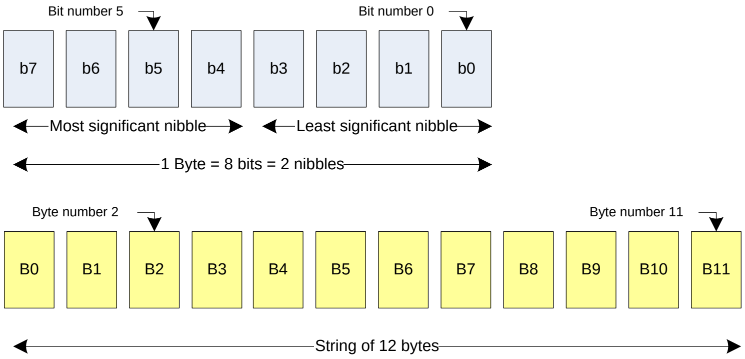
2.5. References

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 anti-collision	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 SDI011

3.1. SDI011 key benefits

With its combination of a modern slim design and its state of the art feature set, SDI011 is the perfect desktop reader choice for environments where both contact and contactless smart card support is required. Such environments may be corporate where physical and logical access control is implemented.

As for all SCM Microsystems products, SDI011 is designed to offer best in class interoperability with various formats of tokens: cards, dongles, watches or NFC mobile phones.

Its infield upgradeable firmware makes SDI011 a secure and future-proof investment providing both flexibility and fast time to market for new applications as well as minimum risk linked to contactless technology standards evolution.

3.2. SDI011 key features

- 13.56MHz contactless reader:
  - ISO14443 type A & B,
  - MIFARE
- ISO7816 compliant contact smart card reader
- PC/SC v2.0 compliant
- In field upgradeable firmware
- Unique serial number which enables that SDI011 can be plugged into any USB slot on a PC without having to re-install the driver.

3.3. SDI011 ordering information

Item	Part number	
SDI011	905214	
Contactless SDK	905124	
Contact SDK	905129	

3.4. SDI011 customization options

Upon request, SCM can customize:

- The color of the casing
- The logo
- The product label
- The USB strings

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

3.5. Contactless communication principles and SDI011 usage recommendations

SDI011 is a dual interface reader capable of reading both contact smart cards and contactless user tokens. The following paragraph focuses on a few specifics of the contactless communication to outline usage recommendations in order to ensure best user experience.

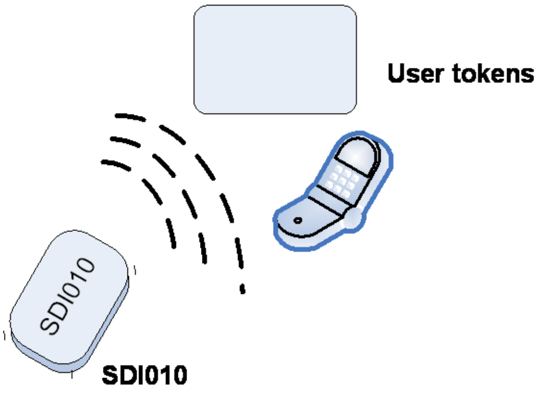
SDI011 is a contactless reader[1] designed to communicate with user tokens.

User tokens[2] are made of a contactless integrated circuit card connected to an antenna

User tokens can take several form factors:

- Credit card sized smart card
- Key fob
- NFC mobile phone etc…

Communication between SDI011 and user tokens uses magnetic field inductive coupling.



The magnetic field generated by SDI011 has a carrier frequency of 13.56MHz.

3.5.1. Power supply

When the user token is put in the magnetic field of the reader, its 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 the user token.

3.5.2. Data exchange

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 refer to the standard for further details.

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

[1] In the ISO/IEC 14443 standard, the reader is called the proximity coupling device (PCD)

[2] In the ISO/IEC 14443 standard, the user token is called proximity integrated chip card (PICC)

3.5.3. Recommendations

The communication between the reader and the user token 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 severely 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.

It is recommended for proper communication to avoid putting SDI011 in close proximity of conductive materials.

The presence of multiple user tokens in the field also interferes with the communication. When several user tokens are in the field of the reader, load of the field increases which implies that less energy is available for each of them and that the system is detuned. For this reason, SCM Microsystems has implemented in its driver only 1 slot by default. This means that in the event several user tokens are in the field of the SDI011, only one will be active. It is possible using INF configuration to enable up to 4 slots – i.e. to activate up to 4 user tokens nevertheless depending on the power consumption of the user tokens communication cannot be guaranteed.

It is recommended to present only one user credential at a time in front of SDI011.

Please note that multiple contactless slots feature is supported but is kept disabled by default. The SDI011 driver on configuration allows the presence and use of several PICCs (maximum 4) at the same time. The driver can support multiple logical connections and present each of them as a slot logical device to the Resource Manager and higher components. Also the simultaneous working of multiple Contactless cards is not guaranteed and depends on the antenna size and the power requirements of the card.

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

SDI011 was primarily designed and optimized to function with user credentials of various technologies having the size of a credit card.

It may happen that SDI011 is not capable of communicating with extremely large or extremely small antennas.

In order to optimize the coupling between the reader and the user token, it is recommended to put both antennas as parallel as possible

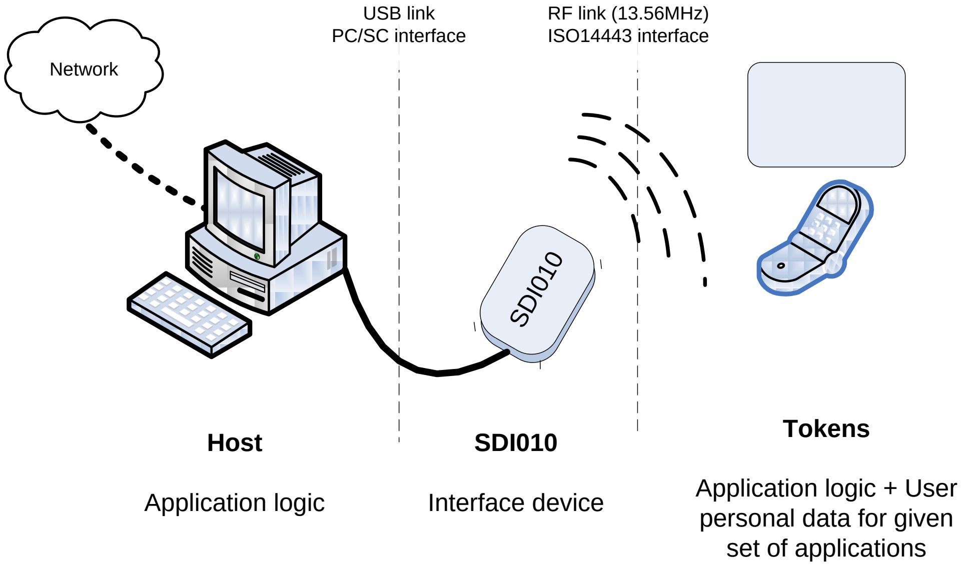
In order to optimize transaction speed between the reader and the card it is recommended to place the user token as close as possible to the reader. This will increase the amount of energy supplied to the user credential which will then be able to use its microprocessor at higher speeds

3.6. Applications

3.6.1. General

SDI011 is a transparent reader designed to interface a personal computer host supporting PC/SC interface with 13.56MHz user tokens like public transport cards, contactless banking cards, electronic identification documents – e.g. e-passports, e-ID cards, driving licenses etc.

Those user tokens can have several form factors like credit cards, key fobs, NFC mobile phones or USB dongles like SCT3511 that SCM Microsystems markets.



SDI011 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 SDI011. There are many tools provided; here are two of them:

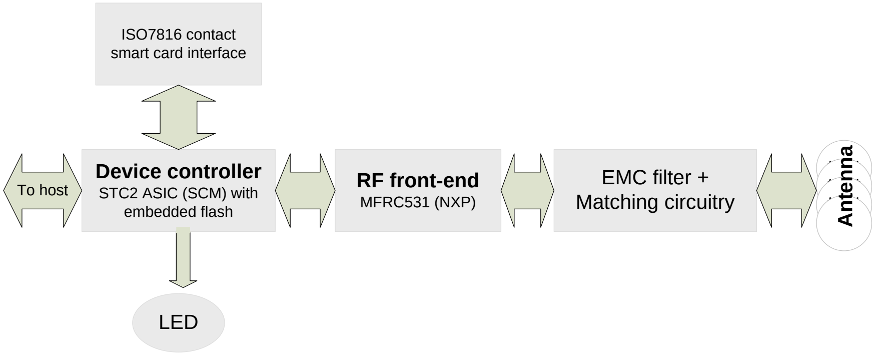
- The NFC forum tag reader/writer is a standalone application that enables the user to read and write NFC forum compliant records into NFC forum compatible tags. It is an easy to use tool to configure rapidly NFC forum tag demonstrations. Note: SDI011 supports NFC forum tag type 2 and 4, only.
- Smart 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 SDI011.

4. SDI011 characteristics

4.1. SDI011 high level architecture

4.1.1. Block diagram

The link between SDI011 and the host to which it is connected is the USB interface providing both the power and the communication channel.


ISO7816 contact smart card interface
To host
Device controller
STC2 ASIC (SCM) with embedded flash
RF front-end
MFRC531 (NXP)
EMC filter + Matching circuitry
Antenna
LED

SDI011 has a device controller which is SCM's STCII ASIC. This ASIC has several interfaces available. In SDI011 implementation 3 peripherals are connected to the device controller:

- LED for reader status indication
- A contact smart card interface
- An RF front-end that handles the RF communication

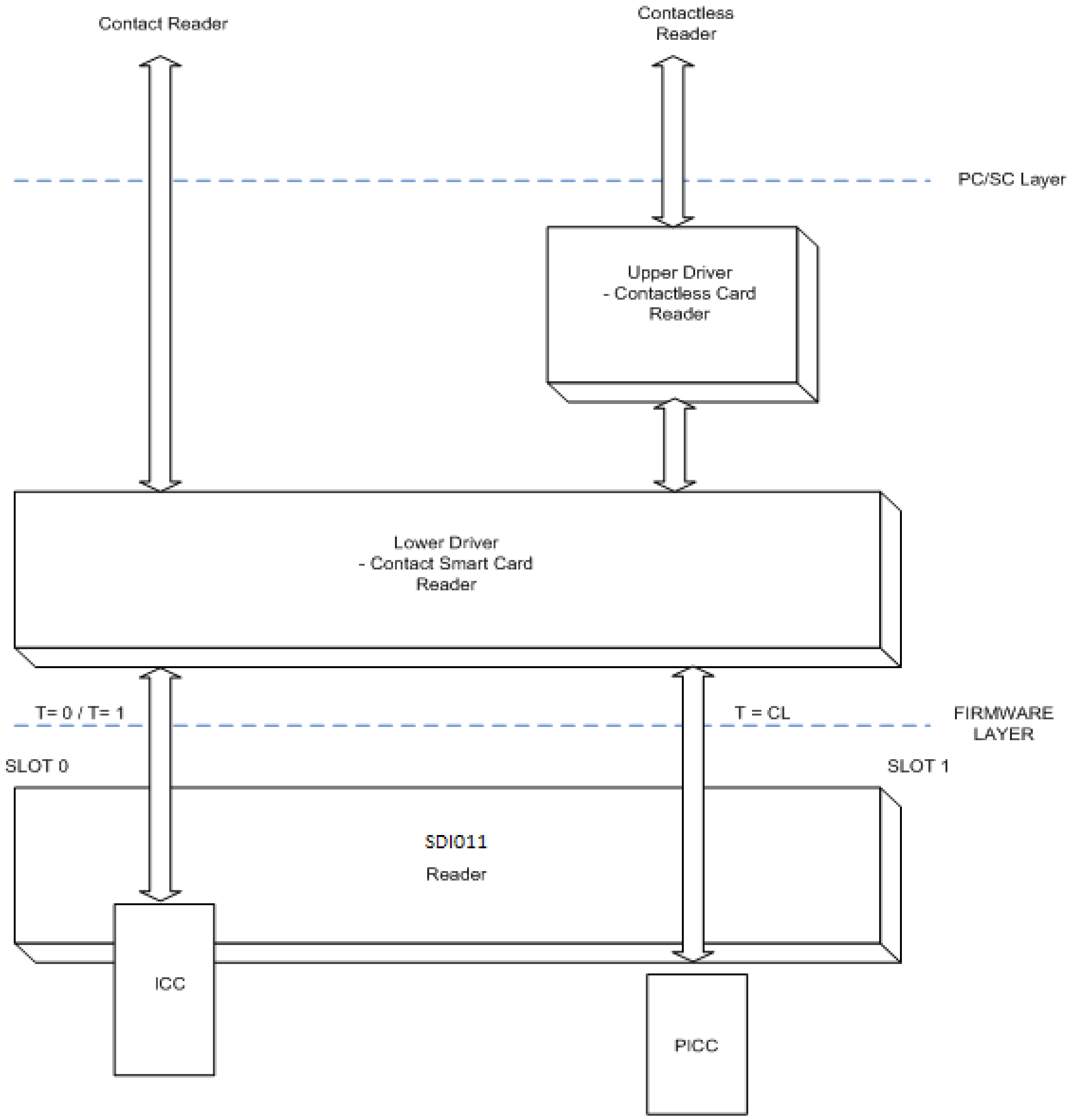
The ASIC embeds flash memory. The flash is programmed during the manufacturing of SDI011 devices. This flash contains the firmware developed by SCM Microsystems to handle all the ISO7816 contact protocol, the RF communication protocols and the PC/SC communication protocol with the host. The flash can be upgraded once the device is deployed in the field, hence enabling firmware upgrades to add and potentially patch features.

The RF front-end ensures the coding/decoding/framing modulation/demodulation required for the RF communication. It is controlled by the device controller through registers.

The matching circuitry provides the transmission and receiver paths adaptation for the antenna to function properly.

4.1.2. Software architecture

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

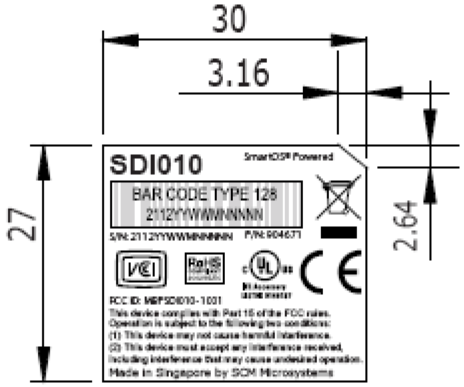


The SDI011 driver implements PC/SC v2.0 API towards upper layers and uses SCM firmware commands encapsulated in CCID-like protocol for the contactless slot and full CCID for the contact slot.

The SDI011 contactless driver handles all the contactless-related intelligence – i.e. ISO/IEC 14443 and the SDI011 firmware handles the raw transport of data to and from the contactless cards.

4.2. Quick reference data

4.2.1. SDI011 dimensions

Item	Characteristic	Value
SDI011	Weight	128 Grams
	External dimensions	L 118 mm × W 78mm × H 22mm
	Cable length	1.5 meter long with USB type A connector
	Default color	Black with metallic silver
	Default label	 30 3.16 27 2.64 SDI010 SmartOS® Powered BAR CODE TYPE 128 2112YYWWNNNNN S/N: 2112YYWWNNNNNN P/N: 904671 FCC ID: MBPSDI010-1001 This device complies with Part 15 of the FCC rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference. (2) This device must accept any interference received, including interference that may cause undesired operation. Made in Singapore by SCM Microsystems

Drawing with dimensions of the SDI011 and accessories can be found in annex.

4.2.2. LED behavior

SDI011 is equipped with a bicolor LED. Its behavior is described in the table below.

SDI011 states	LED1 Indication (GREEN)	LED2 Indication (RED)
Just after plug-in (with drivers already installed)	ON	OFF
Just after DFU operation	ON	OFF
Suspend / standby	OFF	OFF
Reader powered, Contact card IN, but not powered (98/ME – issue power down using the Testresman utility)	500ms ON 500ms OFF	OFF
Reader powered, Contact card IN, but not powered (2K/XP – power down takes place)	ON	OFF
Reader powered, Contactless card IN, but not powered (98/ME - issue power down using the Testresman utility)	500ms ON 500ms OFF	500ms ON 500ms OFF
Reader powered, Contactless card IN, but not powered (2K/XP - power down takes place)	ON	ON
Contact card powered / communication	500ms ON 500ms OFF	OFF
Contactless card powered / communication	500ms ON 500ms OFF	500ms ON 500ms OFF
Reader / card errors	OFF	100ms ON 100ms OFF
Firmware upgrade running	OFF	ON
Combi[3] card powered in contact Slot	500ms ON 500ms OFF	OFF
Combi card powered using RF field	500ms ON 500ms OFF	500ms ON 500ms OFF

[3] A combi card is a smart card which has both a contact and a contactless interface. Some of those cards have one controller with two interfaces. Data can be accessed through the contact or the contactless interface. For those when the contact interface is powered up the contactless interface is disabled. There are nevertheless in the market combi cards with 1 contact chip and 1 contactless chip. Those cards can be seen at the same time as a contact and a contactless card when inserted in the contact interface of SDI011.

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
Operating temperature range	0º to 50ºC
Operating humidity range	Up to 95%RH non condensing
Certifications	USB CE FCC VCCI WEEE RoHS WHQL

4.2.3.2. USB

Parameter	Value/Description
DC characteristics	High bus powered (SDI011 draws power from USB bus) Voltage: 5V Max. Current : 200mA Suspend current : 380uA
USB specification	USB 2.0 FS Device
USB Speed	Full Speed Device (12Mbit/s)
Device Class	Vendor
PID	0x5121
VID	0x04E6

4.2.3.3. Contactless interface

Parameter	Value/Description
RF carrier frequency	13.56 MHz +/- 50ppm
Modulation	12 to 14 %
ID1 format tokens supported	ISO/IEC 14443-4 PICC type A and type B MIFARE Type B memory card PICC through SCM-proprietary APDU
Maximum baud rate	424Kbps (848 Kbps is available as configurable option)
Multiple PICC in field	Supported and is kept disabled by default. Allows the presence and use of several PICC's (Maximum 4) at the same time. The driver can support multiple logical connections and present each of them as a slot logical device to the Resource Manager and higher components. Also the simultaneous working of multiple Contactless cards is not guaranteed and depends on the antenna size and the power requirements of the card.

4.2.3.4. Contact interface

Parameter	Value/Description
Smart card operating frequency	4MHz
Maximum supported card baud-rate	500Kbps
Cards supported	Class A and Class AB smart cards (Class B only cards not supported) Synchronous smart cards
ISO-7816 compliant	Yes
EMV'2000 compliant	Not validated
CT-API compliant	Yes
Number of slots	Single smart card slot
Ejection mechanism	Manual

5. Software modules

5.1. Installation

SCM provides an installer for Windows and for Mac

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

5.2. Utilities

The following utilities are available:

- A tool for device firmware upgrade (DFU)
- 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

The DFU utility comes with a specific driver for dynamic Device Firmware Upgrade (DFU) through the USB interface.

Operating systems supported by DFU tool:

- Windows 98
- Windows ME
- Windows 2000
- Windows 2003 Server (32 & 64 bit)
- Windows XP (32 & 64 bit)
- Windows Vista (32 & 64 bit)
- Windows Server 2008 (32 & 64 bit)

5.3. Driver

5.3.1. SDI011 listing

SDI011 is listed by PC/SC applications as

- *SCM Microsystems Inc. SDI011 Smart Card Reader* for the contact reader
- *SCM Microsystems Inc. SDI011 Contactless Reader* for the contactless reader

5.3.2. Supported operating systems

Operating systems supported by the driver:

- Windows 98
- Windows ME
- Windows 2000
- Windows 2003 Server (32 & 64 bit)
- Windows XP (32 & 64 bit)
- Windows Vista (32 & 64 bit)
- Windows Server 2008 (32 & 64 bit)

5.3.3. PC/SC 2.0 compliant ATR for contactless interface

When a user token is placed on the reader, initialization, anti-collision is done. The user token is automatically activated and an ATR is built as defined in the PC/SC specification.

5.3.3.1. ATR for contactless storage user tokens

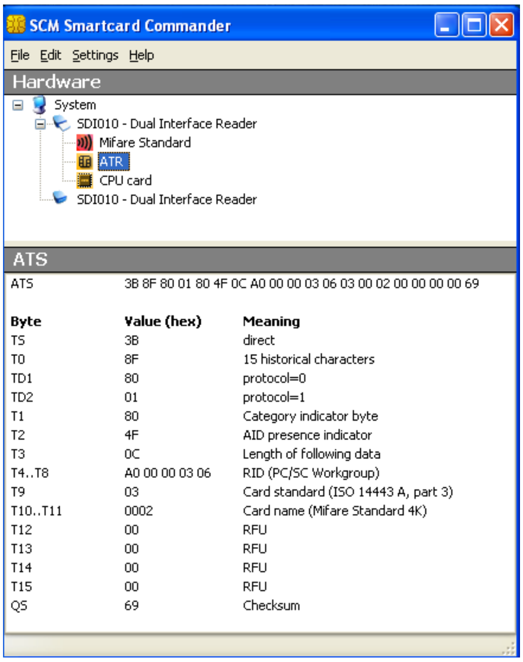
The ATR of the user token is composed as described in the table below. In order to allow the application to identify the storage card properly, it's Standard and Card name describing bytes must be interpreted according to the Part 3 Supplemental Document, maintained by PC/SC.

Tokens using technology like MIFARE are examples of such user tokens.

Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8n	T0	n indicates the number of historical bytes in following ATR
2	0x80	TD1	Nibble8 indicates no TA2, TB2, TC2 Nibble 0 means T=0
3	0x01	TD2	Nibble8 indicates no TA3, TB3, TC3 Nibble 1 means T=1
4...3+n	0x80		A status indicator may be present in an optional TLV data object
	0x4F	Optional TLV data object	Tag: Application identifier
	Length		1 byte
	RID		Registered identifier on 5 bytes
	PIX		Proprietary identifier extension on 3 bytes
	0x00 0x00 0x00 0x00		4 RFU bytes
4+n		TCK	XOR of all previous bytes

Example of the ATR built for contactless storage tokens:

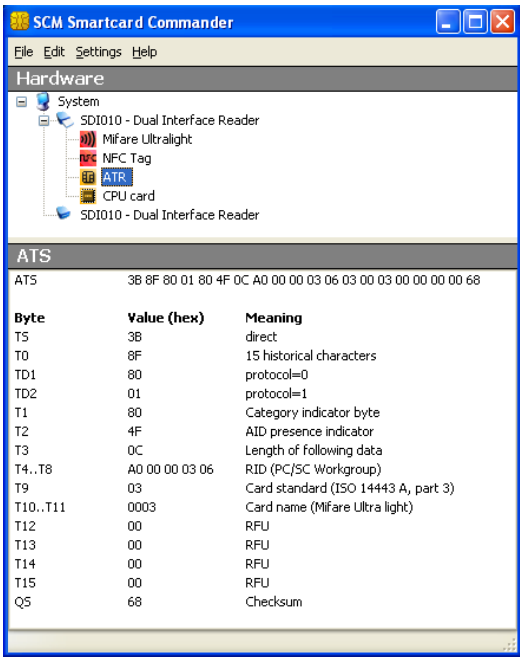
MIFARE Classic 4K



```
ATS         3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 02 00 00 00 00 69

Byte        Value (hex)      Meaning
TS          3B               direct
T0          8F               15 historical characters
TD1         80               protocol=0
TD2         01               protocol=1
T1          80               Category indicator byte
T2          4F               AID presence indicator
T3          0C               Length of following data
T4..T8      A0 00 00 03 06   RID (PC/SC Workgroup)
T9          03               Card standard (ISO 14443 A, part 3)
T10..T11    0002             Card name (Mifare Standard 4K)
T12         00               RFU
T13         00               RFU
T14         00               RFU
T15         00               RFU
QS          69               Checksum
```

MIFARE Ultralight



```
ATS         3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 00 68

Byte        Value (hex)      Meaning
TS          3B               direct
T0          8F               15 historical characters
TD1         80               protocol=0
TD2         01               protocol=1
T1          80               Category indicator byte
T2          4F               AID presence indicator
T3          0C               Length of following data
T4..T8      A0 00 00 03 06   RID (PC/SC Workgroup)
T9          03               Card standard (ISO 14443 A, part 3)
T10..T11    0003             Card name (Mifare Ultra light)
T12         00               RFU
T13         00               RFU
T14         00               RFU
T15         00               RFU
QS          68               Checksum
```

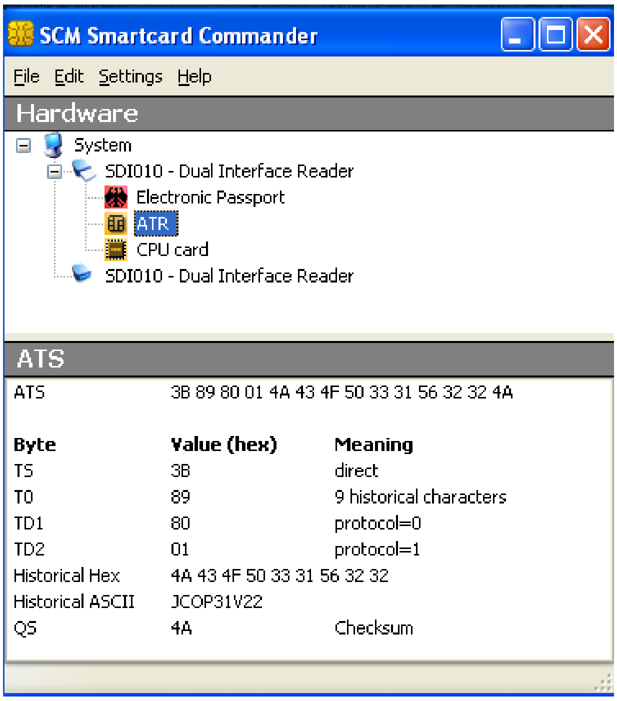
5.3.3.2. ATR for ISO/IEC 14443-4 user tokens

The user token exposes its ATS or application information which is mapped to an ATR. The table describes how this mapping is done.

Byte#	Value	Designation	Description
0	0x3B	Initial header	
1	0x8n	T0	n indicates the number of historical bytes in following ATR
2	0x80	TD1	Nibble8 indicates no TA2, TB2, TC2 Nibble 0 means T=0
3	0x01	TD2	Nibble8 indicates no TA3, TB3, TC3 Nibble 1 means T=1
4...3+n		Historical bytes or application information	Type A: the historical bytes from the ATS (up to 15 bytes) Type B (8 bytes): • Byte 0 through 3: application data from ATQB, • Byte 4 through 6: protocol info byte from ATQB, • Byte 7: highest nibble is the MBLI (maximum buffer length index) from ATTRIB, lowest nibble is 0x0
4+n		TCK	XOR of all previous bytes

Example of the ATR built for an ISO14443-4 user tokens:

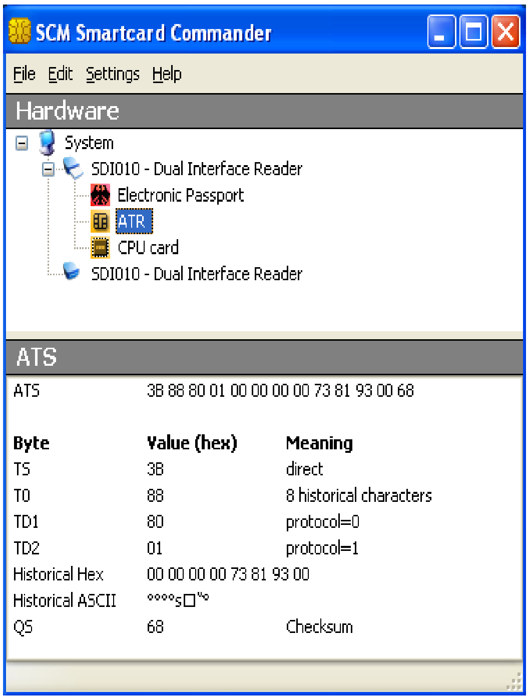
Type A



```
ATS                  3B 89 80 01 4A 43 4F 50 33 31 56 32 32 4A

Byte                 Value (hex)           Meaning
TS                   3B                    direct
T0                   89                    9 historical characters
TD1                  80                    protocol=0
TD2                  01                    protocol=1
Historical Hex       4A 43 4F 50 33 31 56 32 32
Historical ASCII     JCOP31V22
QS                   4A                    Checksum
```

Type B



```
ATS                  3B 88 80 01 00 00 00 00 73 81 93 00 68

Byte                 Value (hex)           Meaning
TS                   3B                    direct
T0                   88                    8 historical characters
TD1                  80                    protocol=0
TD2                  01                    protocol=1
Historical Hex       00 00 00 00 73 81 93 00
Historical ASCII     °°°°s□"°
QS                   68                    Checksum
```

5.4. Firmware

5.4.1. CCID transport protocol

SDI011 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 messages supported

The following CCID messages are supported both for the contact and the contactless interfaces when received through bulk-out endpoint.

- 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_NotifySlotChange

The following CCID messages are NOT implemented and hence fail with command not supported error:

- PC_to_RDR_ResetParameters
- PC_to_RDR_IccClock
- PC_to_RDR_T0APDU
- PC_to_RDR_Secure
- PC_to_RDR_Mechanical
- PC_to_RDR_SetDataRateAndClockFrequency

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. XFR_PARITY_ERROR

This error code is returned when a parity error condition is detected. This error will be reported in the response to a PC_to_RDR_XfrBlock message. This error code has been defined in the error code table 6.2-2 of the CCID specification.

5.4.1.3.3. 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.

5.4.2. Automatic PPS for the contactless interface

Automatic PPS is implemented in SDI011's driver. This means that by default SDI011 switches to the maximum communication speed indicated by the card during its selection. Automatic PPS can be disabled using escape messages as explained later in this manual.

When Auto PPS is disabled (discussed in escape messages section) the reader works at the default baud rate of 106kbps. An escape command has been introduced in the driver to force the required baud rate.

The maximum speed supported by SDI011 is 424Kbps by default. Using escape messages as explained later in this manual it is possible to change this.

6. Commands description

6.1. Generic APDU

6.1.1. Get UID Command

6.1.1.1. Description

GET UID will retrieve the UID or SNR or PUPI of the user token. This command can be used for all supported technologies.

6.1.1.2. Format

CLA	INS	P1	P2	Lc	Data in	Le
0xFF	0xCA	0x00	0x00	-	-	XX

Setting Le = 0x00 can be used to request the full UID or PUPI is sent back.(e.g. for ISO14443A single 4 bytes, double 7 bytes, triple 10 bytes, for ISO14443B 4 bytes PUPI).

6.1.1.3. Response

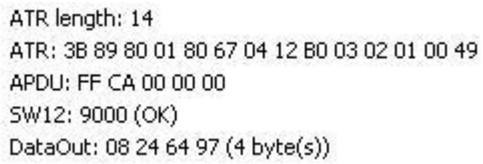
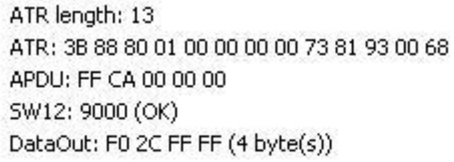
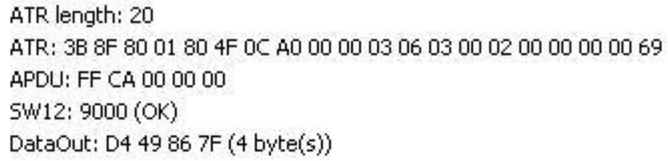
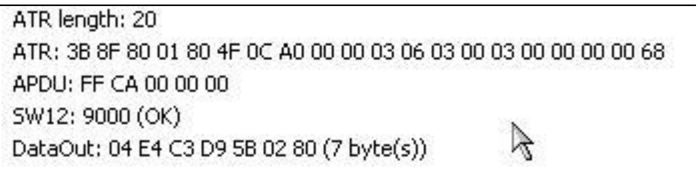
Data Out
UID + SW1 + SW2

6.1.1.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x62	0x82	End of UID reached before Le bytes (Le is greater than UID length)
0x6C	0xXX	Wrong Length. 0xXX is the exact value for Le

Further error codes can be found in annex

6.1.1.5. Examples

ISO14443-4A	 ATR length: 14 ATR: 3B 89 80 01 80 67 04 12 B0 03 02 01 00 49 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: 08 24 64 97 (4 byte(s))
ISO14443-4B	 ATR length: 13 ATR: 3B 88 80 01 00 00 00 00 73 81 93 00 68 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: F0 2C FF FF (4 byte(s))
MIFARE 4K	 ATR length: 20 ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 02 00 00 00 00 69 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: D4 49 86 7F (4 byte(s))
MIFARE Ultralight	 ATR length: 20 ATR: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 00 68 APDU: FF CA 00 00 00 SW12: 9000 (OK) DataOut: 04 E4 C3 D9 5B 02 80 (7 byte(s))

6.1.2. Escape command APDU

6.1.2.1. Description

This command can be used to send escape commands to SDI011. For description of escape commands please refer to the dedicated chapter in this manual.

6.1.2.2. Format

CLA	INS	P1	P2	P3	Data in
0xFF	0xCC	0x00	0x00	Lc	Input buffer of escape command

Lc is the length of the escape command's input buffer. See escape commands description later in this manual

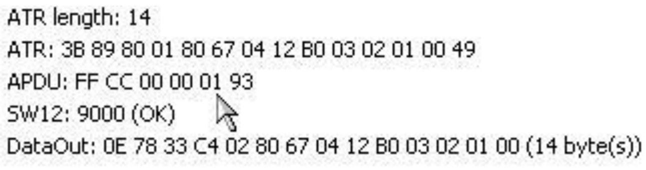
6.1.2.3. Response

Output buffer of the escape command

6.1.2.4. Example

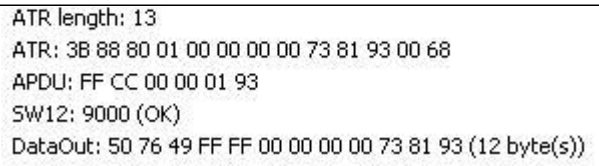
To get the ATS or ATQB of the ISO14443-4 based user token, you can use this APDU to send the READER_CNTLESS_GET_ATS_ATQB (0x93) escape command

Type A passport



```
ATR length: 14
ATR: 3B 89 80 01 80 67 04 12 B0 03 02 01 00 49
APDU: FF CC 00 00 01 93
SW12: 9000 (OK)
DataOut: 0E 78 33 C4 02 80 67 04 12 B0 03 02 01 00 (14 byte(s))
```

Type B passport

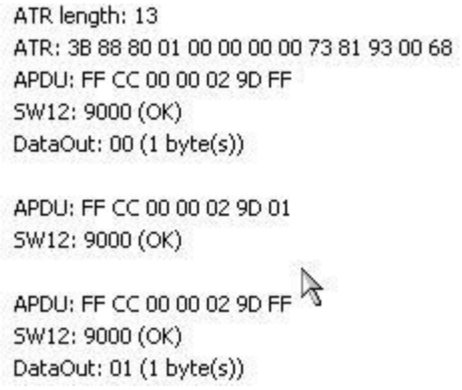


```
ATR length: 13
ATR: 3B 88 80 01 00 00 00 00 73 81 93 00 68
APDU: FF CC 00 00 01 93
SW12: 9000 (OK)
DataOut: 50 76 49 FF FF 00 00 00 00 73 81 93 (12 byte(s))
```

To get the reader status about support of 848Kbps, you can use this APDU to send the READER_CNTLESS_848KBPS (0x9D) escape command.

By default the SDI011 doesn't have 848Kbps enabled on its contactless interface, the following sequence

- Checks the status (0x00 as response, means 848Kbps is disabled)
- Enables 848Kbps
- Checks the status again and the answer 0x01 indicates 848Kbps is enabled



```
ATR length: 13
ATR: 3B 88 80 01 00 00 00 00 73 81 93 00 68
APDU: FF CC 00 00 02 9D FF
SW12: 9000 (OK)
DataOut: 00 (1 byte(s))

APDU: FF CC 00 00 02 9D 01
SW12: 9000 (OK)

APDU: FF CC 00 00 02 9D FF
SW12: 9000 (OK)
DataOut: 01 (1 byte(s))
```

6.2. Set of APDU for contactless storage user tokens

6.2.1. STORAGE_CARD_CMDS_READ_BINARY

6.2.1.1. Description

Using this APDU, application can read a memory block on user tokens based on technologies like MIFARE Classic 1K or 4K (block size 0x10 bytes) or MIFARE Ultra light (block size 0x04 bytes).

6.2.1.2. Format

CLA	INS	P1	P2	Le
0xFF	0xB0	Address MSB	Address LSB	0xXX

Where:

- P2 indicates the block number from where to read
- Le can be a short (maximum value 255) or extended (maximum value 65535). If Le=0x00, then all the bytes until the end of the block are read (0x10 bytes for MIFARE Classic 1K or 4K cards and 0x04 bytes for MIFARE Ultra Light cards).

6.2.1.3. Response

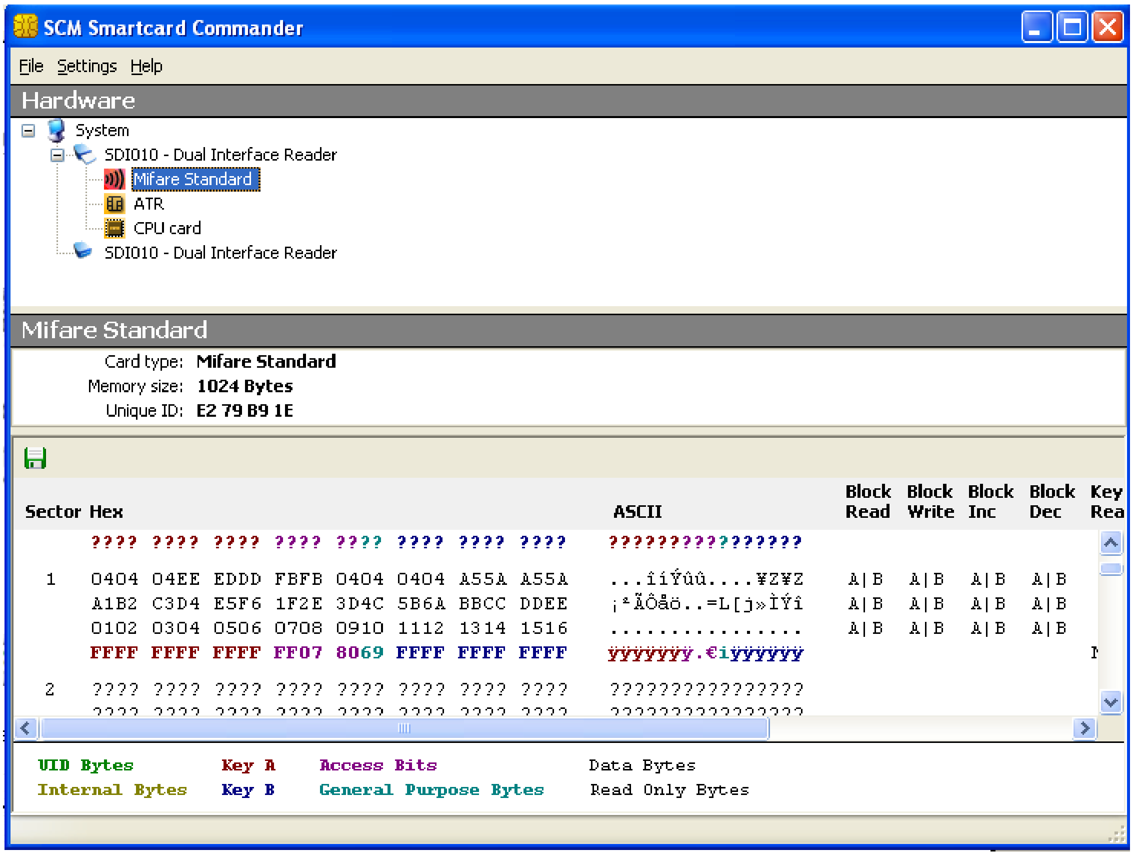
Data Out
Data + SW1 + SW2

6.2.1.4. Status words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x62	0x81	WARNING: part of the returned data may be corrupted
	0x82	WARNING: end of file reached before Le bytes where read
0x67	0x00	Length incorrect
0x68	0x00	CLA byte incorrect
0x69	0x81	Command not supported
	0x82	Security status not satisfied
	0x86	Command not allowed
0x6A	0x81	Function not supported
	0x82	File not found, addressed blocks or bytes do not exist
0x6B	0x00	Wrong P1, P2 parameters
0x6C	0xXX	Wrong Le, 0xXX is the correct value

6.2.1.5. Example

For a MIFARE Classic 1K card which has the following memory content:



Card type: **Mifare Standard**
Memory size: **1024 Bytes**
Unique ID: **E2 79 B9 1E**

```
                                                                    Block Block Block Block Key
Sector Hex                                      ASCII               Read  Write Inc   Dec   Rea
       ???? ???? ???? ???? ???? ???? ???? ????  ????????????????

  1    0404 04EE EDDD FBFB 0404 0404 A55A A55A  ...îiÝûû....¥Z¥Z    A|B   A|B   A|B   A|B
       A1B2 C3D4 E5F6 1F2E 3D4C 5B6A BBCC DDEE  ¡ªÃÔåö..=L[j»ÌÝî    A|B   A|B   A|B   A|B
       0102 0304 0506 0708 0910 1112 1314 1516  ................    A|B   A|B   A|B   A|B
       FFFF FFFF FFFF FF07 8069 FFFF FFFF FFFF  ÿÿÿÿÿÿÿ.€iÿÿÿÿÿÿ

  2    ???? ???? ???? ???? ???? ???? ???? ????  ????????????????
       ???? ???? ???? ???? ???? ???? ???? ????  ????????????????
```

UID Bytes   Key A   Access Bits   Data Bytes
Internal Bytes   Key B   General Purpose Bytes   Read Only Bytes

To read the seventh block, you have to issue the following command and get the following response:

```
APDU: FF B0 00 06 10
SW12: 9000 (OK)
DataOut: 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 (16 byte(s))
```

6.2.2. STORAGE_CARD_CMDS_WRITE_BINARY

6.2.2.1. Description

This APDU writes data to a memory address

6.2.2.2. Format

CLA	INS	P1	P2	Lc	Data in
0xFF	0xD6	Address MSB	Address LSB	0xXX	Data

Where:

- P2 indicate the memory block number where data should be written
- Lc=0x10 for MIFARE Classic 1K/4K. Lc=0x04 for MIFARE Ultralight

6.2.2.3. Response

Data Out
SW1 + SW2

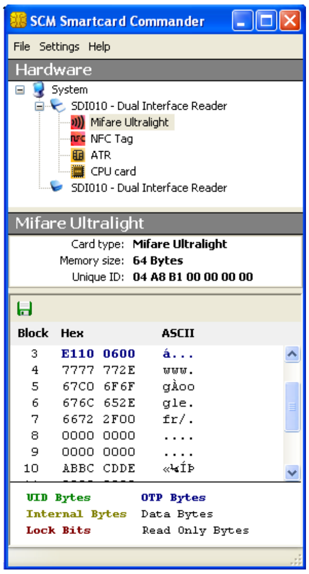
6.2.2.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x69	0x81	Command not supported
0x64	0x00	State of the non-volatile memory unchanged

6.2.2.5. Example

For a MIFARE Classic Ultralight card which has the following memory content:

Issuing the command



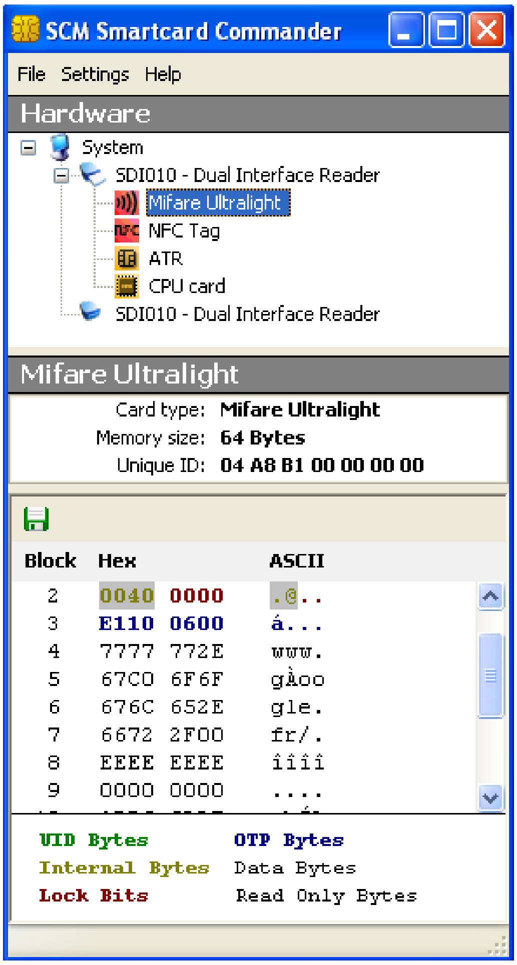
Card type: Mifare Ultralight
Memory size: 64 Bytes
Unique ID: 04 A8 B1 00 00 00 00

```
Block  Hex          ASCII
  3    E110 0600    á...
  4    7777 772E    www.
  5    67C0 6F6F    gÀoo
  6    676C 652E    gle.
  7    6672 2F00    fr/.
  8    0000 0000    ....
  9    0000 0000    ....
 10    ABBC CDDE    «¼ÍÞ
```

UID Bytes OTP Bytes
Internal Bytes Data Bytes
Lock Bits Read Only Bytes

```
APDU: FF D6 00 08 04 EE EE EE EE
SW12: 9000 (OK)
```

Results into the following memory mapping



Card type: **Mifare Ultralight**
Memory size: **64 Bytes**
Unique ID: **04 A8 B1 00 00 00 00**

```
Block  Hex        ASCII
  2    0040 0000  .@..
  3    E110 0600  á...
  4    7777 772E  www.
  5    67C0 6F6F  gÀoo
  6    676C 652E  gle.
  7    6672 2F00  fr/.
  8    EEEE EEEE  îîîî
  9    0000 0000  ....
```

```
UID Bytes         OTP Bytes
Internal Bytes    Data Bytes
Lock Bits         Read Only Bytes
```

6.2.3. STORAGE_CARD_CMDS_LOAD_KEYS

6.2.3.1. Description

Some type of user tokens like MIFARE Classic may require that an authentication happens before any data can be read or written. To perform this authentication, keys need to be loaded in the reader's memory using this command.

6.2.3.2. Format

CLA	INS	P1	P2	Lc	Data in
0xFF	0x82	0x00	Key Type	Key Length	Key value

Where P2 can have the following values (please refer to MIFARE documentation from NXP for further details on what is key A and Key B):

- 0x60 to use the Key A
- 0x61 to use the Key B

6.2.3.3. Response

Data Out
SW1 + SW2

6.2.3.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x69	0x83	Reader key not supported
	0x85	Secured transmission not supported
	0x87	Non volatile memory not available
	0x88	Key number not valid
	0x89	Key length not correct

6.2.4. STORAGE_CARD_CMDS_AUTHENTICATE

6.2.4.1. Description

This command enables to perform authentication for user tokens based on MIFARE Classic 1K or 4K. Before this command can be successfully executed, the STORAGE_CARD_CMDS_LOAD_KEY command must have been executed.

6.2.4.2. Format

CLA	INS	P1	P2	Lc	Data in
0xFF	0x86	0x00	0x00	0x05	Data

Where the data field is structured as follow

Byte #	Value	Description
B0	0x01	Version
B1		Address MSB
B2		Address LSB
B3	0x60	Key A
	0x61	Key B
B4		Number of the key to be used for authentication

Information about memory structure of MIFARE Classic must be requested from NXP Semiconductors.

6.2.4.3. Response

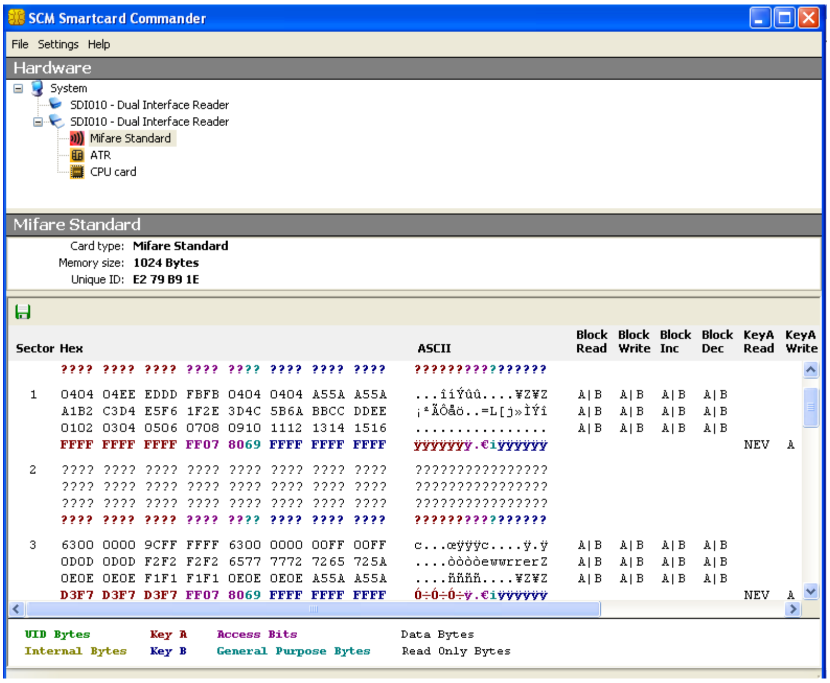
Data Out
SW1 + SW2

6.2.4.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x63	0x00	WARNING no further info
0x69	0x82	Security status not satisfied
	0x84	Referenced key not usable
	0x86	Key type not known

6.2.4.5. Example

For a MIFARE Classic 1K card which has the following memory mapping:



Card type: Mifare Standard
Memory size: 1024 Bytes
Unique ID: E2 79 B9 1E

```
Sector Hex                                      ASCII             Block Block Block Block KeyA KeyA
                                                                  Read  Write Inc   Dec   Read Write
       ???? ???? ???? ???? ???? ???? ???? ????  ????????????????
  1    0404 04EE EDDD FBFB 0404 0404 A55A A55A  ...îíÝûû....¥Z¥Z  A|B   A|B   A|B   A|B
       A1B2 C3D4 E5F6 1F2E 3D4C 5B6A BBCC DDEE  ¡²ÃÔåö..=L[j»ÌÝî  A|B   A|B   A|B   A|B
       0102 0304 0506 0708 0910 1112 1314 1516  ................  A|B   A|B   A|B   A|B
       FFFF FFFF FFFF FF07 8069 FFFF FFFF FFFF  ÿÿÿÿÿÿÿ.€iÿÿÿÿÿÿ                          NEV  A
  2    ???? ???? ???? ???? ???? ???? ???? ????  ????????????????
       ???? ???? ???? ???? ???? ???? ???? ????  ????????????????
       ???? ???? ???? ???? ???? ???? ???? ????  ????????????????
       ???? ???? ???? ???? ???? ???? ???? ????  ????????????????
  3    6300 0000 9CFF FFFF 6300 0000 00FF 00FF  c...œÿÿÿc....ÿ.ÿ  A|B   A|B   A|B   A|B
       0D0D 0D0D F2F2 F2F2 6577 7772 7265 725A  ....òòòòewwrrerZ  A|B   A|B   A|B   A|B
       0E0E 0E0E F1F1 F1F1 0E0E 0E0E A55A A55A  ....ññññ....¥Z¥Z  A|B   A|B   A|B   A|B
       D3F7 D3F7 D3F7 FF07 8069 FFFF FFFF FFFF  Ó÷Ó÷Ó÷ÿ.€iÿÿÿÿÿÿ                          NEV  A
```

UID Bytes   Key A   Access Bits   Data Bytes
Internal Bytes   Key B   General Purpose Bytes   Read Only Bytes

Reading sector 0 or sector 1 of this card requires authentication with key A or key B.

The following example:

- authenticates with key A of sector 1
- reads block #6
- authenticates against sector 3
- reads block #E

```
APDU: FF 82 00 60 06 FF FF FF FF FF FF
SW12: 9000 (OK)

APDU: FF 86 00 00 05 01 00 06 60 00
SW12: 9000 (OK)

APDU: FF B0 00 06 10
SW12: 9000 (OK)
DataOut: 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 (16 byte(s))

APDU: FF 82 00 60 06 D3 F7 D3 F7 D3 F7
SW12: 9000 (OK)

APDU: FF 86 00 00 05 01 00 0E 60 00
SW12: 9000 (OK)

APDU: FF B0 00 0E 10
SW12: 9000 (OK)
DataOut: 0E 0E 0E 0E F1 F1 F1 F1 0E 0E 0E 0E A5 5A A5 5A (16 byte(s))
```

6.2.5. STORAGE_CARD_CMDS_VALUE_BLOCK

6.2.5.1. Description

This APDU is used to interact with MIFARE Classic e-purse applications. Please refer to MIFARE Classic documentation available from NXP Semiconductors for further details on MIFARE classic memory mapping and commands.

6.2.5.2. Format

CLA	INS	P1	P2	Lc	Data in
0xFF	0xF0	0x00	Block#	Lc	Data

Where P2 code the address of the block number addressed

Where the data field is structured as follow

Byte #	Value	Description
B0	0xC0	Increment
	0xC1	Decrement
B1		Block number
B2-B5		Value (LSB first)

6.2.5.3. Response

Data Out
SW1 + SW2

6.2.5.4. Status Words

SW1	SW2	Description
0x90	0x00	NO ERROR
0x67	0x00	Length incorrect
0x68	0x00	CLA byte incorrect
0x6A	0x81	Function not supported
0x6B	0x00	Wrong P1, P2 parameters

6.2.5.5. Example

CLA	INS	P1	P2	Lc	Data in
0xFF	0xF0	0x00	0x1E	0x06	0xC0 0x1E 0x01 0x00 0x00 0x00

The above APDU will increment the value in block number 0x1E of a MIFARE Classic-based user token by a value of 0x01.

6.3. Set of APDU for ISO/IEC14443-4 user tokens

6.3.1. T=CL Command

Description

SDI011 can transfer directly ISO/IEC7816-4 APDU to the PICC.

SDI011 supports user tokens that have both the MIFARE and T=CL partitions. Depending on the APDU sent by the host, the reader switches to the corresponding mode (MIFARE or T=CL) automatically and the command is processed accordingly.

6.3.1.1. Format

CLA	INS	P1	P2	P3	Data

Description of the APDU commands can be found in ISO/IEC 7816-4 specification.

6.3.1.2. Response

Data Out
PICC answer as defined in ISO/IEC 7816-4+ SW1 + SW2

As defined in ISO/IEC 7816-4.

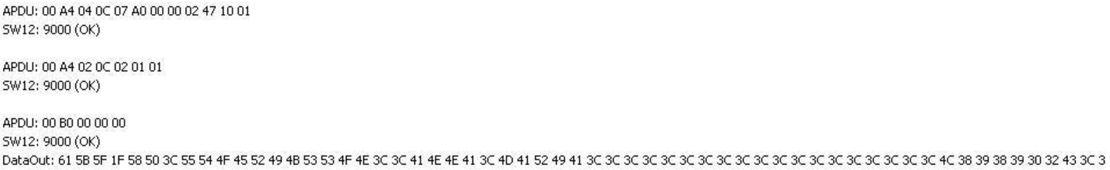
6.3.1.3. Status Words

SW1	SW2	Description
		See ISO/IEC 7816-4

As defined in ISO/IEC 7816-4.

6.3.1.4. Example

The following APDU sequence reads the first 256 bytes of the data group 1 as specified in ICAO LDS (logical data structure) for machine readable travel documents with open access. It first selects the issuer application using its AID (0xA0 0x00 0x00 0x02 0x47 0x10 0x01), then selects the DG1 file (0x01 0x01) and then does a read binary.



```
APDU: 00 A4 04 0C 07 A0 00 00 02 47 10 01
SW12: 9000 (OK)

APDU: 00 A4 02 0C 02 01 01
SW12: 9000 (OK)

APDU: 00 B0 00 00 00
SW12: 9000 (OK)
DataOut: 61 5B 5F 1F 58 50 3C 55 54 4F 45 52 49 4B 53 53 4F 4E 3C 3C 41 4E 4E 41 3C 4D 41 52 49 41 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 4C 38 39 38 39 30 32 43 3C 3
```

6.3.2. T=CL user command

Description

This command can be used to send raw data to the user token.

6.3.2.1. Format

CLA	INS	P1	P2	P3	Data
0xFF	0xFE	0x00	0x00	Lraw_data	Raw_data

6.3.2.2. Response

Data Out
PICC response data+ SW1 + SW2

6.3.2.3. Status Words

SW1	SW2	Description

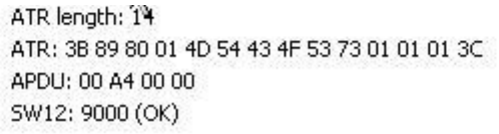
User should refer to the status words defined by the PICC manufacturer for a description of the status words

6.3.2.4. Example

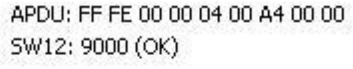
Let’s consider the Select command defined in ISO7816-4. This command being ISO can be sent to the user token in 2 different ways:

- Using the T=CL command
- Using the T=CL user command

Here are the 2 answers for the select command:



```
ATR length: 14
ATR: 3B 89 80 01 4D 54 43 4F 53 73 01 01 01 3C
APDU: 00 A4 00 00
SW12: 9000 (OK)
```



```
APDU: FF FE 00 00 04 00 A4 00 00
SW12: 9000 (OK)
```

The T=CL command is nevertheless more useful for sending commands which are not defined in ISO7816.

6.4. Set of APDU defined by SCM Microsystems

6.4.1. MIFARE DESFire Commands

Description

This command can be used to send commands to DESFire-based user tokens.

For a description of DESFire commands please contact NXP Semiconductors.

6.4.1.1. Format

CLA	INS	P1	P2	P3	Data
0xFF	0xDE	0x00	0x00	Lcommand	Command

Response

Data Out
DESFire response data+ 9000 if the DESFire response data is of single byte DESFire response data if the DESFire response data is more than 1 byte

6.5. Escape commands for the contactless interface

6.5.1. Sending escape commands to SDI011

A developer can use 2 methods to send escape commands to SDI011 to the contactless interface

- SCardControl method defined in PC/SC API
- SCardTransmit method defined in PC/SC API in conjunction with the escape command APDU defined earlier in this manual

6.5.2. Escape command codes

Escape commands can be used by an application to configure SDI011 to function in a mode that is not its default configured mode or to get specific information. To put the SDI011 back into its default mode, either the SDI011 has to be unplugged and plugged again or the application can send again the same escape command.

The following escape commands are supported by SDI011 for the contactless interface.

Escape command	Code
READER_GETCARDINFO	0x11
READER_LEDCONTROL	0x19
READER_CNTLESS_GET_MFRC_REV	0x92
READER_CNTLESS_GET_ATS_ATQB	0x93
READER_CNTLESS_GET_TYPE	0x94
READER_CNTLESS_SET_TYPE	0x95
READER_CNTLESS_RF_SWITCH	0x96
READER_CNTLESS_RAW_CFG	0x97
READER_CNTLESS_RAW_XMIT_EX	0xAE
READER_ CNTLESS_DISABLE_PPS	0x99
READER_SWITCH_RF_ON_OFF	0x9C
READER_CNTLESS_848KBPS	0x9D
READER_CNTLESS_BAUDRATE	0x9E
READER_CNTLESS_FORCE_BAUDRATE_PCSC_REV2	0xAD
READER_LEDCTRL_BY_FW	0xB2

Sample code to send escape commands can be found in annex.

6.5.3. READER_GETCARDINFO

This escape command is used to get information about the card placed on the reader. The SDI011 returns an error if no card is placed on it.

The input buffer shall contain the escape command code

Input buffer
0x11

The output buffer contents are described below.

Output buffer	Value	Description
B0	0x01	Contactless card present
B1	0xNN	Baud rate of card-reader communication
B2	0xXY	X – Upper nibble indicates 0 - memory card 1 - T=CL card 2 - Dual mode card Y – Lower nibble indicates 0 - Type A card 1 - Type B card

The Baud rate of card-reader communication 0xNN shall indicate a BYTE as follows

b8	b7	b6	b5	b4	b3	b2	b1
				0			

b1 – 212kbps supported (direction reader to card)

b2 – 424kbps supported (direction reader to card)

b3 – 848kbps supported (direction reader to card)

b5 – 212kbps supported (direction card to reader)

b6 – 424kbps supported (direction card to reader)

b7 – 848kbps supported (direction card to reader)

b8 – 1 – indicates same baud rate in both directions
0 – indicates different baud rates in opposite directions

For Example:

If 0xNN = 0x77, the card supports all baud rates namely 106, 212, 424 and 848 kbps in both directions. This card can be forced to work at different baud rates in the send and receive directions using the escape command READER_CNTLESS_FORCE_BAUDRATE_PCSC_REV2.

If 0xNN = 0xB3, the card supports 106, 212 and 424 kbps in both directions. This card can be made to work only at the same baud rate in the send and receive directions using the escape command READER_CNTLESS_FORCE_BAUDRATE_PCSC_REV2.

6.5.4. READER_LED_CONTROL_BY_FW

This escape command may be used to enable or disable LED control by the firmware.

The input buffer is

Byte #	Value	Description
B0	0xB2	Escape command code
B1	0x00	Disable LED control by FW
	0x01	Enable LED control by FW

The output buffer is

Output buffer
NULL

6.5.5. READER_LEDCONTROL

This escape command is used to turn ON/OFF the LED.

This escape command shall work only if LED control by firmware is disabled.

The input buffer shall contain 3 bytes

Byte #	Value	Description
B0	0x19	Escape command code
B1	0x00	LED number
B2	0x00	LED ON
	0x01	LED OFF

The output buffer is

Output buffer
NULL

6.5.6. READER_CNTLESS_GET_MFRC_REV

This escape message retrieves the revision number of the RF ASIC MFRC531.

The input buffer contains the escape command code

Input buffer
0x92

The output buffer contains the version of the MFRC531 ASIC.

6.5.7. READER_CNTLESS_GET_ATS_ATQB

This escape command enables the host to retrieve the ATS for Type A T= CL or the ATQB for Type B cards.

The input buffer contains the escape command code

Input buffer
0x93

The output buffer contains the ATS bytes or the ATQB bytes depending on the type of PICC placed on the reader.

6.5.8. READER_CNTLESS_GET_TYPE

This escape command retrieves the type of the card which SDI011 is configured to poll for.

The input buffer shall contain the escape command code

Input buffer
0x94

The output buffer shall point to a BYTE buffer which will contain the type value coded as

Value	Description
0x00	Type A
0x01	Type B
0x02	Type A + type B

6.5.9. READER_CNTLESS_SET_TYPE

This escape command configures the type of cards SDI011 will poll for.

Using this command can improve the polling efficiency of SDI011 for applications where only type A or only type B cards are expected.

The default is Type A + type B (0x02).

The input buffer shall contain 2 bytes

Byte #	Value	Description
B0	0x95	Escape command code
B1	0x00	Type A
	0x01	Type B
	0x02	Type A + type B

The output buffer is

Output buffer
NULL

6.5.10. READER_CNTLESS_RF_SWITCH

This escape command can be used to retrieve/set the RF state of SDI011.

The default RF field state is ON.

The input buffer shall contain 2 bytes

Byte #	Value	Description
B0	0x96	Escape command code
B1	0x00	Switch RF Field OFF
	0x01	Switch RF Field ON
	0xFF	Get current field state

After the RF is turned off, to turn the RF ON again, card connect shall be done in direct mode.

If B1 of the input buffer is 0x00 or 0x01 the output buffer is

Output buffer
NULL

If B1 of the input buffer is 0xFF, the output buffer is a BYTE buffer with 2 possible values

Output buffer	Description
0x01	RF field is OFF
0x00	RF field is ON

6.5.11. READER_CNTLESS_RAW_CFG

This escape command switches SDI011 to raw mode.

When SDI011 is in raw mode it only polls for one type of contactless card.

SDI011 is by default not in this mode and therefore READER_CNTLESS_RAW_XMIT_EX would fail.

The input buffer contains 2 bytes

Byte #	Value	Description
B0	0x97	Escape Function code
B1	0x00	Type A will be use for further transmissions in raw mode
	0x01	Type B will be use for further transmissions in raw mode

The output buffer is

Output buffer
NULL

Once SDI011 is in raw mode commands can be sent using READER_CNTLESS_RAW_XMIT_EX escape command.

6.5.12. READER_CNTLESS_RAW_XMIT_EX

This escape command can only be executed by the firmware once SDI011 is put in raw mode using the READER_CNTLESS_RAW_CFG escape command.

This escape command can be used to send commands to smart card when SDI011 is in raw mode

The input buffer is

Byte #	Value	Description
B0	0xAE	Escape Function code
B1		Wait Time
B2		Is CRC required?
B3		No of bits per command
B4		Card Type 0 – Type A 1 – type B
B5		Command length
B6 - Bn		Command

The output buffer contains the response to the command from the offset B6 onwards.

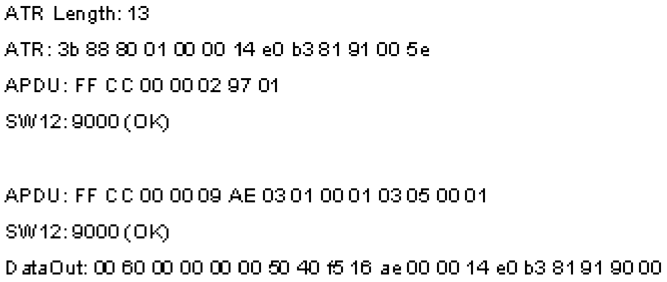
The following example uses the raw mode to send a REQB command

First, we have to switch the SDI011 into raw mode for type B communication

Byte #	Value	Description
B0	0x97	READER_CNTLESS_RAW_CONFIG code
B1	0x01	Type B will be used

Then, we can send the following bytes to obtain the ATQB response of any type B user token in the field

Byte #	Value	Description
B0	0xAE	READER_CNTLESS_RAW_XMIT_EX code
B1	0x03	FWI is set to 3
B2	0x01	Enable CRC (CRC will be calculated by the RF front end of SDI011)
B3	0x00	Number of bits to be sent in the command 0 – Entire byte will be sent
B4	0x01	Type B
B5	0x03	Command length in bytes
B6	0x05	REQB command’s anti-collision prefix byte
B7	0x00	REQB command’s application family identifier
B8	0x01	REQB command parameter with slot number set as 1



```
ATR Length: 13
ATR: 3b 88 80 01 00 00 14 e0 b3 81 91 00 5e
APDU: FF CC 00 00 02 97 01
SW12: 9000 (OK)

APDU: FF CC 00 00 09 AE 03 01 00 01 03 05 00 01
SW12: 9000 (OK)
DataOut: 00 60 00 00 00 00 50 40 f5 16 ae 00 00 14 e0 b3 81 91 90 00
```

6.5.13. READER_ CNTLESS_DISABLE_PPS

By default SDI011 does automatic PPS – i.e. it switches the RF communication speed to the highest possible supported by the card.

This escape command can be used to switch ON/OFF automatic PPS. When automatic PPS is OFF, then 106Kbps only is available.

The input buffer is

Byte #	Value	Description
B0	0x99	Escape command code
B1	0x01	Disable Auto-PPS
	0x00	Enable Auto-PPS

The output buffer is

Output buffer
NULL

6.5.14. READER_SWITCH_RF_ON_OFF

This escape command can be used to switch the RF field ON or OFF when a Contact smart card is inserted into the reader.

By default, the RF field is always in the ON state and when any contact smart card is inserted in the reader, the RF field is turned OFF.

The input buffer shall contain 2 bytes

Byte #	Value	Description
B0	0x9C	Escape command code
B1	0x00	Switch RF Field OFF when contact card is present in the reader
	0x01	Switch RF Field ON when contact card is present in the reader
	0xFF	Get current field state when Contact smart card is present in the reader

After the RF is turned off, to turn the RF ON again, card connect shall be done in direct mode.

If B1 of the input buffer is 0x00 or 0x01 the output buffer is

Output buffer
NULL

If B1 of the input buffer is 0xFF, the output buffer is a BYTE buffer with 2 possible values

Output buffer	Description
0x00	RF field is OFF
0x01	RF field is ON

6.5.15. READER_CNTLESS_848KBPS

This escape command can be used to enable/disable 848kbps support by SDI011 as well as query whether 848kbps is currently enabled or disabled by SDI011.

The RF communication with a user token will only switch to 848Kbps provided the user token supports this baud rate and provided automatic PPS is ON.

The input buffer shall contain 2 bytes

Byte #	Value	Description
B0	0x9D	Escape command code
B1	0x00	Disable 848Kbps support
	0x01	Enable 848Kbps support
	0xFF	Get current status on 848Kbps support

If B1 of the input buffer is 0x00 or 0x01 then the output buffer is

Output buffer
NULL

If B1 of the input buffer is 0xFF, the output buffer is a BYTE buffer with following possible values

Output buffer	Description
0x00	848Kbps is disabled
0x01	848Kbps is enabled

6.5.16. READER_CNTLESS_BAUDRATE

This escape command can be used to get the actual operating baud rate of card-reader communication.

The input buffer shall contain the escape message value.

Input buffer
0x9E

The output buffer shall point to a BYTE buffer with following possible values

Output buffer	Description
0x00	106Kbps in both directions
0x01	106Kbps from PICC to PCD, 212Kbps from PCD to PICC
0x02	106Kbps from PICC to PCD, 424Kbps from PCD to PICC
0x03	106Kbps from PICC to PCD, 848Kbps from PCD to PICC
0x10	212Kbps from PICC to PCD, 106Kbps from PCD to PICC
0x11	212Kbps in both directions
0x12	212Kbps from PICC to PCD, 424Kbps from PCD to PICC
0x13	212Kbps from PICC to PCD, 848Kbps from PCD to PICC
0x20	424Kbps from PICC to PCD, 106Kbps from PCD to PICC
0x21	424Kbps from PICC to PCD, 212Kbps from PCD to PICC
0x22	424Kbps in both directions
0x23	424Kbps from PICC to PCD, 848Kbps from PCD to PICC
0x30	848Kbps from PICC to PCD, 106Kbps from PCD to PICC
0x31	848Kbps from PICC to PCD, 212Kbps from PCD to PICC
0x32	848Kbps from PICC to PCD, 424Kbps from PCD to PICC
0x33	848Kbps in both directions

6.5.17. READER_CNTLESS_FORCE_BAUDRATE_PCSC_REV2

This escape command can be used to force baud rate for Contactless cards.

The input buffer is

Byte #	Value	Description
B0	0xAD	Escape command code
B1	0x00	Apply the baud rate specified by the card
	0x01	Force baud rate specified in B2
B2	b0- DR=2 supported, if bit is set to 1 b1- DR=4 supported, if bit is set to 1 b2- DR=8 supported, if bit is set to 1 b3- shall be set to 0, 1 is RFU b4- DS=2 supported, if bit is set to 1 b5- DS=4 supported, if bit is set to 1 b6- DS=8 supported, if bit is set to 1 b7- 1 if the same D is required for both communication directions b8- 0 if different D is supported for each communication direction	Encoding of the baud rate to be forced if B1 value is 0x01. No need to send this byte in case B1 has the value =x00
	NULL	If B1=0x00

The output buffer is

Output buffer
NULL

6.6. Escape commands for the contact interface

6.6.1. Sending escape commands to SDI011

A developer can use the following method to send escape commands to SDI011 for the contact interface

- SCardControl method defined in PC/SC API

6.6.2. Escape command codes

Escape commands can be used by an application to configure SDI011 to function in a mode that is not its default configured mode or to get specific information. To put the SDI011 back into its default mode, either the SDI011 has to be unplugged and plugged again or the application can send again the same escape command.

The following escape commands are supported by SDI011 for the contact interface

Escape command	**Code**
READER_SETMODE	0x01
READER_GETMODE	0x02
READER_APDU_TRANSFER	0x08
READER_SWITCH_SPEED	0x0A
READER_SWITCH_PROTOCOL	0x0C
READER_DISABLE_PPS	0x0F
READER_GETIFDTYPE	0x12
READER_GETINFO_EXTENDED	0X1E

6.6.3. READER_SETMODE

This escape command may be used to set the mode of the reader. Applications may call this function, to set the desired mode. Typically, this call is used to switch between the EMV, ISO7816 and the memory card modes of operation.

The input buffer is

Byte #	**Value**	**Description**
B0	0x01	Escape command code
B1	0x00	ISO 7816 mode
	0x01	EMV mode
	0x02	Memory card mode

The output buffer is

NULL

6.6.4. READER_GETMODE

This escape command may be used to retrieve the current mode of the reader.

The input buffer is

Byte #	Value	Description
B0	0x02	Escape command code

The output buffer is

0x00	ISO 7816 mode
0x01	EMV mode
0x02	Memory card mode

6.6.5. READER_APDU_TRANSFER

This escape command may be used to exchange an APDU with the smart card.

The input buffer is

Byte #	Value	Description
B0	0x08	Escape command code
B1- Bn		Command APDU

The output buffer contains the response APDU.

The maximum number of bytes that can be transmitted and received 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

6.6.6. READER_SWITCH_SPEED

In case, when the application is capable of switching the card’s speed through APDU (if the card supports such a feature), this escape command is used to inform the reader about the speed change occurred between application and card. The first byte will contain the escape function value; the next two bytes contain Fi and Di respectively. The output buffer field shall be NULL.

The input buffer is

Byte #	Value	Description
B0	0x0A	Escape command code
B1		Fi value
B2		Di value

The output buffer is

NULL

6.6.7. READER_SWITCH_PROTOCOL

In case, when the application is capable of switching the card’s protocol through APDU (if the card support such a feature), this escape command is used to inform the reader about the protocol change occurred between application and card.

The input buffer is

Byte #	Value	Description
B0	0x0C	Escape command code
B1	0x00	T0_PROTOCOL
	0X01	T1_PROTOCOL

The output buffer is

NULL

6.6.8. READER_DISABLE_PPS

This escape command disables the automatic PPS done by the firmware.

The input buffer is

Byte #	Value	Description
B0	0x0F	Escape command code
B1	0x00	Enable PPS
	0X01	Disable PPS

The output buffer is

NULL

6.6.9. READER_GETIFDTYPE

This escape command is used to get the current IFD type from the reader. The first byte of the input buffer contains the escape id value. The reader gets the value from the reader capability structure, which is implemented in the reader as a configurable item. The output buffer shall point to a WORD buffer. The IFD type of SDI011-Generic is 0x010E,

The input buffer is

Byte #	Value	Description
B0	0x12	Escape command code

6.6.10. READER_GETINFO_EXTENDED

This escape command is used to get the information of the reader like the major and minor version of the firmware, capabilities of the reader and the Unicode serial number. The output buffer shall point to an application allocated SCARD_READER_GETINFO_PARAMS_EX structure mentioned below.

The input buffer is

Byte #	Value	Description
B0	0x1E	Escape command code

```
typedef struct __SCARD_READER_GETINFO_PARAMS_EX
{
        OUT     BYTE    byMajorVersion;
        OUT     BYTE    byMinorVersion;
        OUT     BYTE    bySupportedModes; // 0 – ISO7816, 1 – EMV, 2 – Memory card
        OUT     WORD    wSupportedProtocols; // 1 – T=0; 2 - T=1; 3 – T=0 & T=1
        OUT     WORD    winputDevice;
        OUT     BYTE    byPersonality;
        OUT     BYTE    byMaxSlots;
        OUT     BYTE    bySerialNoLength;
        OUT     BYTE[28] bySerialNumber;
}SCARD_READER_GETINFO_PARAMS_EX, *PSCARD_READER_GETINFO_PARAMS_EX;
```

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
0x69	0x82	Command not allowed, security status not satisfied
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. SDI011 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();
}
```

7.3. Annex C - SCM Proprietary CLA bytes

0xF0	Contact Memory cards
0xFF	MIFARE-TCL Switching
	T=CL User command
	Escape command APDU

The second SCM Proprietary APDU is blocked for the application layer. This is used for internal communication i.e. between the driver and the firmware.

Function	CLA byte – PC/SC1.0	CLA byte – PC/SC2.0
T=CL User Command APDU	0xFC	0xFF
MIFARE DESFire APDU	0xFC	0xFF
Escape Command APDU	0xFD	0xFF

In order to maintain compatibility with some customer applications which use the CLA bytes of PC/SC 1.0 architecture, the following switching mechanism can be used.

Option1: Use the CLA byte of PC/SC1.0 architecture

Option2: Use the CLA byte of PC/SC2.0 architecture

The above two options can be controlled by

- Configuring the firmware
- INF/registry entry
- Vendor IOCTL.

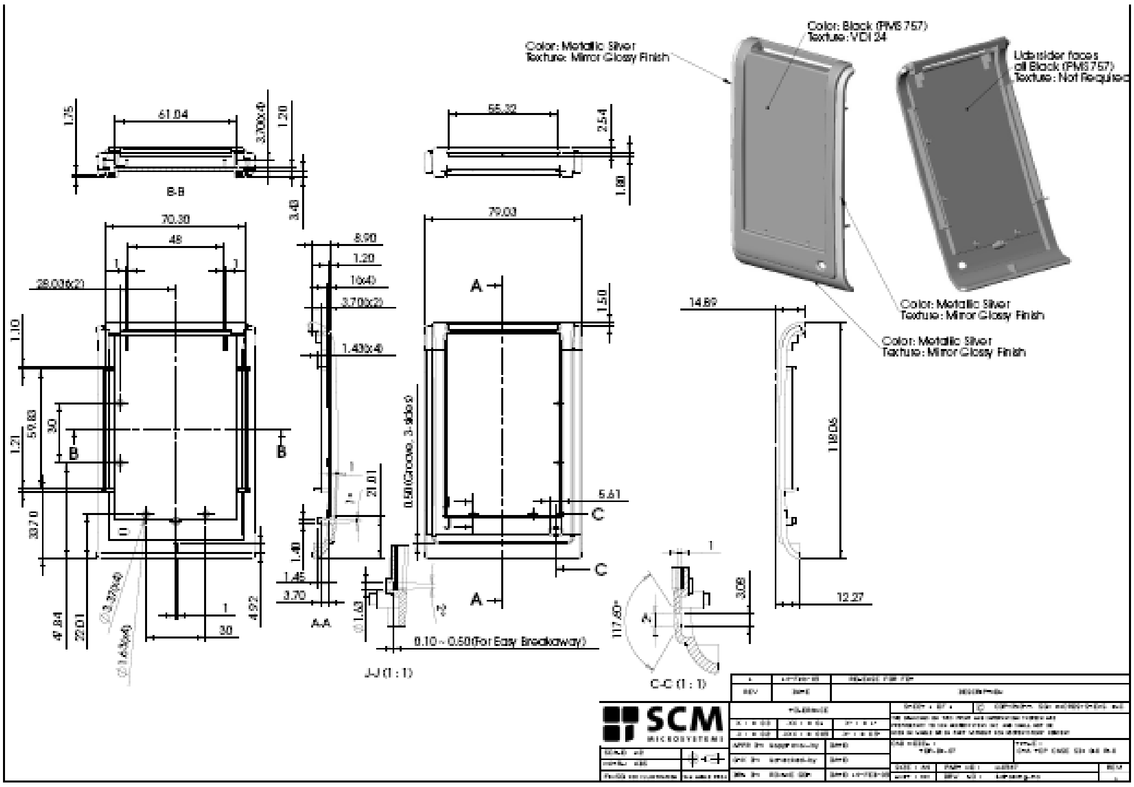
By default, option #2 is set in the firmware binary.

The entry System\CurrentControlSet\Services\SCM\ProprietaryAPDUOption is kept disabled in the INF by default. Enabling this key and setting this entry to 0 will override the firmware setting with option #1. Enabling this key and setting this entry to 1 will override the firmware setting with option #2.

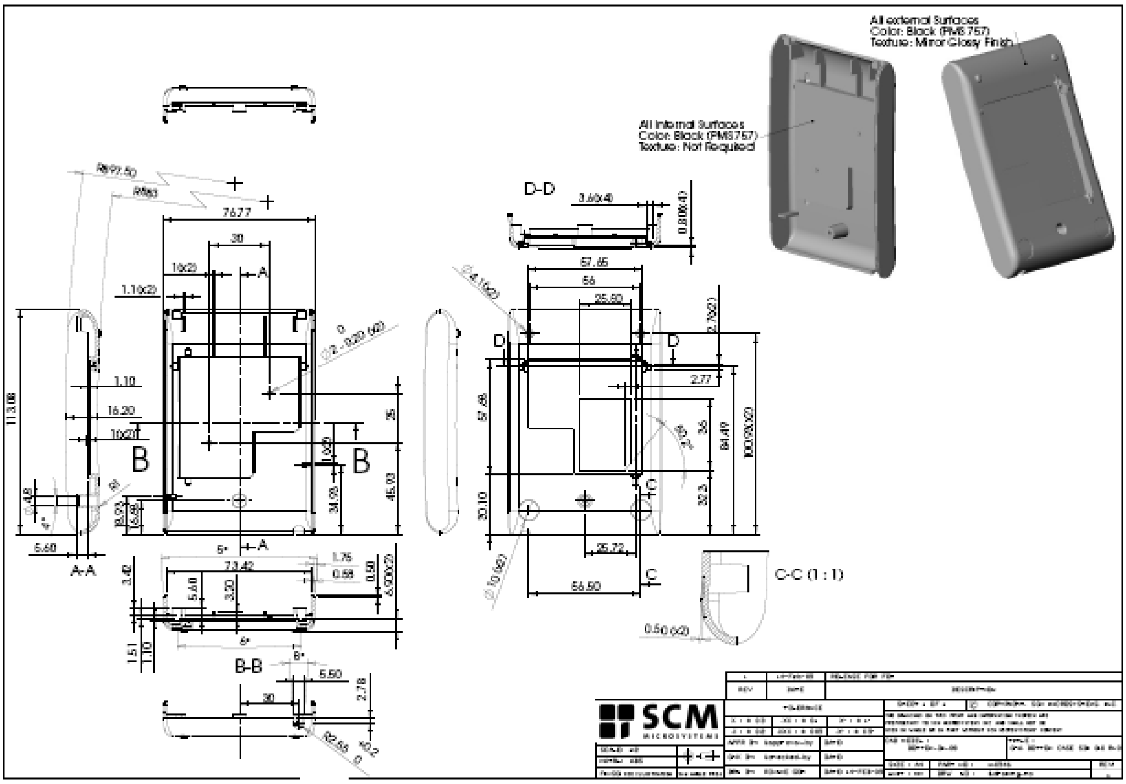
Vendor IOCTL_SWITCH_PROPRIETARY_APDU_OPTION (0x856) can also be used to switch between the two options. Input buffer with value 0x00 will switch to option#1 and input buffer with value 0x01 will switch to option#2.

7.4. Annex D – Mechanical drawings

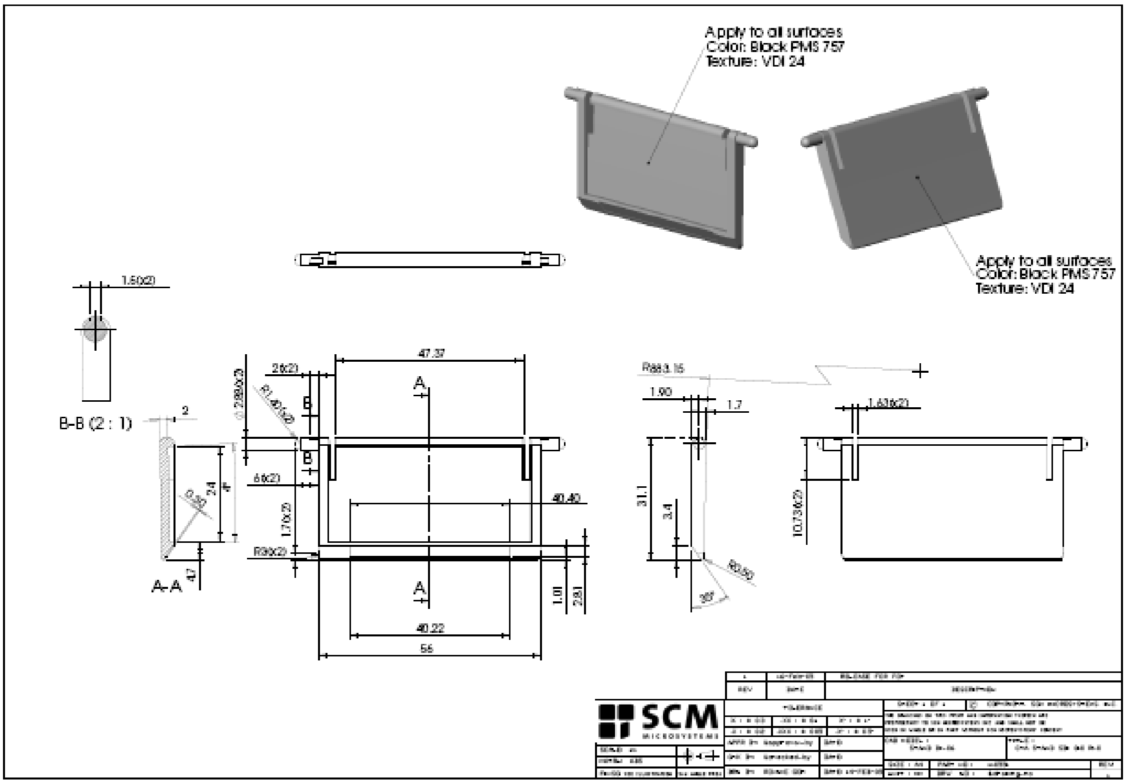
7.4.1. Top Casing



7.4.2. Bottom Casing



7.4.3. Stand


Apply to all surfaces
Color: Black PMS 757
Texture: VDI 24
Apply to all surfaces
Color: Black PMS 757
Texture: VDI 24
1.60(2)
B-B (2 : 1)
2
24
4
0.35
A-A
47
2.28(x2)
2(x2)
R1.40(x2)
47.37
A
B
B
6(x2)
1.70(2)
R30(x2)
40.40
A
1.01
2.81
40.22
56
R883.16
1.90
1.7
31.1
3.4
R0.50
30°
1.63(x2)
10.73(x2)
REV
DATE
DESCRIPTION
TOLERANCE
SCM
MICROSYSTEMS