Advanced Card Systems Ltd.
Card & Reader Technologies

ACR39 Series
PC-linked Smart Card Readers

Reference Manual V1.04

info@acs.com.hk
www.acs.com.hk

Revision History

Release Date	Revision Description	Version Number
2013-09-06	● Initial Release	1.00
2014-03-25	● Updated Section 2.0: Features ● Updated Title from ACR39U to ACR39x ● Added brand trademark attributions ● Moved USB Interface section above Contact Smart Card Interface	1.01
2015-05-13	● Updated Section 2.0: Features	1.02
2015-12-14	● Updated Section 2.0: Features ● Re-arranged format structure ● Updated Section 9.1: Other Commands accessed via PC_to_RDR_Xfrblock ● Removed Appendix A – Supported Card Types	1.03
2017-10-10	● Updated Document Title ● Renamed all ACR39x to ACR39 ● Updated Section 2.0: Features	1.04

Advanced Card Systems Ltd.
Card & Reader Technologies

Table of Contents

List of Tables

1.0. Introduction

The ACR39 PC-linked Smart Card Reader acts as an interface for the communication between a computer and a smart card. Different types of smart cards have different commands and different communication protocols, which, in most cases, prevent direct communication between a smart card and a computer. The ACR39 Smart Card Reader establishes a uniform interface from the computer to the smart card for a wide variety of cards. By taking care of the card's particulars, it releases the computer software programmer from being responsible with smart card operations' technical details, which in many cases, are not relevant to the implementation of a smart card system.

1.1. Reference Documents

The following related documents are available from www.usb.org

- Universal Serial Bus Specification 2.0 (also referred to as the USB specification), April 27, 2000
- Universal Serial Bus Common Class Specification 1.0, December 16, 1997
- Universal Serial Bus Device Class: Smart Card CCID Specification for Integrated Circuit(s) Cards Interface Devices, Revision 1.1, April 22, 2005

The following related documents can be ordered through www.ansi.org

- ISO/IEC 7816-1; Identification Cards – Integrated circuit(s) cards with contacts - Part 1: Physical Characteristics
- ISO/IEC 7816-2; Identification Cards – Integrated circuit(s) cards with contacts - Part 2: Dimensions and Locations of the contacts
- ISO/IEC 7816-3; Identification Cards – Integrated circuit(s) cards with contacts - Part 3: Electronic signals and transmission protocols

1.2. Symbols and Abbreviations

Abbreviation	Description
ATR	Answer-To-Reset
CCID	Chip/Smart Card Interface Device
ICC	Integrated Circuit Cards
IFSC	Information Field Sized for ICC for protocol T=1
IFSD	Information Field Sized for CCID for protocol T=1
NAD	Node Address
PPS	Protocol and Parameters Selection
RFU	Reserved for future use[1]
TPDU	Transport Protocol Data Unit
USB	Universal Serial Bus

Table 1: Symbols and Abbreviations

[1] *Must be set to zero unless stated differently.*

2.0. Features

2.1. ACR39U

- USB Full Speed Interface*
- Plug-and-Play – CCID support brings utmost mobility
- Smart Card Reader:
  - Contact Interface:
    - Supports ISO 7816 Class A, B and C (5 V, 3 V, 1.8 V) cards
    - Supports CAC (Common Access Card)
    - Supports SIPRNET Card
    - Supports J-LIS Card
    - Supports microprocessor cards with T=0 and T=1 protocol
    - Supports memory cards
    - Supports PPS (Protocol and Parameters Selection)
    - Features Short Circuit Protection
- Application Programming Interface:
  - Supports PC/SC
  - Supports CT-API (through wrapper on top of PC/SC)
- Supports Android™ 3.1 and later[2]
- Compliant with the following standards:
  - EN60950/IEC 60950
  - ISO 7816
  - EMV™ Level 1 (Contact)
  - PC/SC
  - CCID
  - CE
  - FCC
  - WEEE
  - RoHS 2
  - REACH
  - FIPS 201 (USA)
  - TAA (USA)
  - J-LIS (Japan)
  - VCCI (Japan)
  - PBOC (China)
  - Microsoft® WHQL

[2] *Uses an ACS Defined Android Library*

**Available in USB Type A and USB Type C Connectors*

2.2. ACR39U PocketMate II

- USB Full Speed Interface*
- Plug-and-Play – CCID support brings utmost mobility
- Swivel Motion Design
- Smart Card Reader:
  - Contact Interface:
    - Supports ISO 7816 Class A, B and C (5 V, 3 V, 1.8 V) cards
    - Supports CAC (Common Access Card)
    - Supports SIPRNET Card
    - Supports J-LIS Card
    - Supports microprocessor cards with T=0 and T=1 protocol
    - Supports memory cards
    - Supports PPS (Protocol and Parameters Selection)
    - Features Short Circuit Protection
- Application Programming Interface:
  - Supports PC/SC
  - Supports CT-API (through wrapper on top of PC/SC)
- Supports Android™ 3.1 and later[3]
- Compliant with the following standards:
  - EN60950/IEC 60950
  - ISO 7816
  - EMV™ Level 1 (Contact)
  - PC/SC
  - CCID
  - CE
  - FCC
  - WEEE
  - RoHS 2
  - REACH
  - FIPS 201 (USA)
  - TAA (USA)
  - J-LIS (Japan)
  - VCCI (Japan)
  - PBOC (China)
  - Microsoft® WHQL

[3] *Uses an ACS Defined Android Library*

**Available in USB Type A, USB Micro-B, and USB Type C Connectors*

2.3. ACR39T

- USB Full Speed Interface*
- Plug-and-Play – CCID support brings utmost mobility
- Includes protective USB cap
- Smart Card Reader:
  - Contact Interface:
    - Supports ISO 7816 Class A, B and C (5 V, 3 V, 1.8 V) cards
    - Supports microprocessor cards with T=0 and T=1 protocol
    - Supports memory cards
    - Supports PPS (Protocol and Parameters Selection)
    - Features Short Circuit Protection
- Application Programming Interface:
  - Supports PC/SC
  - Supports CT-API (through wrapper on top of PC/SC)
- Supports Android™ 3.1 and later[4]
- Compliant with the following standards:
  - EN60950/IEC 60950
  - ISO 7816
  - PC/SC
  - CCID
  - CE
  - FCC
  - WEEE
  - RoHS 2
  - REACH
  - VCCI (Japan)
  - Microsoft® WHQL

[4] *Uses an ACS Defined Android Library*

**Available in USB Micro-B and USB Type C Connectors*

3.0. Smart Card Support

3.1. MCU Cards

The ACR39 is a PC/SC–compliant smart card reader that supports ISO 7816 Class A, B and C (5 V, 3 V, and 1.8 V) smart cards. It also works with MCU cards following either the T=0 and T=1 protocol.

The card ATR indicates the specific operation mode (TA2 present; bit 5 of TA2 must be 0) and when that particular mode is not supported by the ACR39, it will reset the card to negotiable mode. If the card cannot be set to negotiable mode, the reader will then reject the card.

When the card ATR indicates the negotiable mode (TA2 not present) and communication parameters other than the default parameters, the ACR39 will execute the PPS and try to use the communication parameters that the card suggested in its ATR. If the card does not accept the PPS, the reader will use the default parameters (F=372, D=1).

Note: *For the meaning of the aforementioned parameters, please refer to ISO 7816-3.*

3.2. Memory-based Smart Cards

ACR39 works with several memory-based smart cards such as:

- Cards following the I2C bus protocol (free memory cards) with maximum 128 bytes page with capability, including:
    - Atmel®: AT24C01/02/04/08/16/32/64/128/256/512/1024
    - SGS-Thomson: ST14C02C, ST14C04C
    - Gemplus: GFM1K, GFM2K, GFM4K, GFM8K
- Cards with intelligent 1 KB EEPROM with write-protect function, including:
    - Infineon®: SLE4418, SLE4428, SLE5518 and SLE5528
- Cards with intelligent 256-byte EEPROM with write-protect function, including:
    - Infineon®: SLE4432, SLE4442, SLE5532 and SLE5542

4.0. USB Interface

4.1. Communication Parameters

The ACR39 is connected to a computer through USB as specified in the USB Specification 2.0. The ACR39 is working in full speed mode, i.e. 12 Mbps.

Pin	Signal	Function
1	**V_{BUS}**	+5 V power supply for the reader
2	**D-**	Differential signal transmits data between ACR39 and computer
3	**D+**	Differential signal transmits data between ACR39 and computer
4	**GND**	Reference voltage level for power supply

Table 2: USB Interface Wiring

4.2. Endpoints

The ACR39 uses the following endpoints to communicate with the host computer:

Control Endpoint	For setup and control purpose.
Bulk-OUT	For command to be sent from host to ACR39. (Data packet size is 64 bytes)
Bulk-IN	For response to be sent from ACR39 to host. (Data packet size is 64 bytes)
Interrupt-IN	For card status message to be sent from ACR39 to host. (Data packet size is 8 bytes)

5.0. Contact Smart Card Interface

The interface between the ACR39 and the inserted smart card follows the specification of ISO 7816-3 with certain restrictions or enhancements to increase the practical functionality of ACR39.

5.1. Smart Card Power Supply VCC (C1)

The current consumption of the inserted card must not be higher than 50 mA.

5.2. Programming Voltage VPP (C6)

According to ISO 7816-3, the smart card contact C6 (VPP) supplies the programming voltage to the smart card. Since all common smart cards in the market are EEPROM-based and do not require the provision of an external programming voltage, the contact C6 (VPP) has been implemented as a normal control signal in the ACR39. The electrical specifications of this contact are identical to those of the signal RST (at contact C2).

5.3. Card Type Selection

The controlling computer must always select the card type through the proper command sent to the ACR39 prior to activating the inserted card. This includes both the memory cards and MCU-based cards.

For MCU-based cards, the reader allows to select the preferred protocol, T=0 or T=1. However, this selection is only accepted and carried out by the reader through the PPS when the card inserted in the reader supports both protocol types. Whenever an MCU-based card supports only one protocol type, T=0 or T=1, the reader automatically uses that protocol type, regardless of the protocol type selected by the application.

5.4. Interface for Microcontroller-based Cards

For microcontroller-based smart cards, only the contacts C1 (VCC), C2 (RST), C3 (CLK), C5 (GND) and C7 (I/O) are used. A frequency of 4.8 MHz is applied to the CLK signal (C3).

5.5. Card Tearing Protection

The ACR39 provides a mechanism to protect the inserted card when it is suddenly withdrawn while it is powered up. The power supply to the card and the signal lines between the ACR39 and the card is immediately deactivated when the card is being removed. However, as a rule to avoid any electrical damage, a card should only be removed from the reader while it is powered down.

Note: *The ACR39 never switches on the power supply to the inserted card by itself. The controlling computer through the proper command sent to the reader must explicitly do this.*

6.0. Power Supply

The ACR39 requires a voltage of 5 V DC, 100 mA, regulated, power supply. The ACR39 gets power supply from the computer (through the cable supplied along with each type of reader).

6.1. Status LED

The LED indicates the activation status of the smart card interface:

- **Flashing slowly (turns on 200 ms every 2 seconds)**

  Indicates ACR39 is powered up and in the standby state. Either the smart card has not been inserted or the smart card has not been powered up (if it is inserted).

- **Lighting up**

  Indicates power supply to the smart card is switched on, i.e., the smart card is activated.

- **Flashing quickly**

  Indicates there are communications between ACR39 and smart card.

7.0. USB Communication Protocol

ACR39 shall interface with the host through the USB connection. A specification, namely CCID, has been released within the industry defining such a protocol for the USB chip-card interface devices. CCID covers all the protocols required for operating smart cards.

The configurations and usage of USB endpoints on ACR39 shall follow CCID Rev 1.0 Section 3.

An overview is summarized below:

1. *Control Commands* are sent on control pipe (default pipe). These include class-specific requests and USB standard requests. Commands that are sent on the default pipe report information back to the host on the default pipe.
2. *CCID Events* are sent on the interrupt pipe.
3. *CCID Commands* are sent on BULK-OUT endpoint. Each command sent to ACR39 has an associated ending response. Some commands can also have intermediate responses.
4. *CCID Responses* are sent on BULK-IN endpoint. All commands sent to ACR39 have to be sent synchronously (e.g., *bMaxCCIDBusySlots* is equal to 01h for ACR39).

The ACR39 supported CCID features are indicated in its Class Descriptor:

Offset	Field	Size	Value	Description
0	*bLength*	1		Size of this descriptor, in bytes.
1	*bDescriptorType*	1		CCID Functional Descriptor type.
2	*bcdCCID*	2		CCID Specification Release Number in binary-coded decimal.
4	*bMaxSlotIndex*	1		One slot is available on ACR39.
5	*bVoltageSupport*	1		ACR39 can supply 1.8 V, 3 V, and 5 V to its slot.
6	*dwProtocols*	4		ACR39 supports T=0 and T=1 protocol.
10	*dwDefaultClock*	4		Default ICC clock frequency is 4.8 MHz.
14	*dwMaximumClock*	4		Maximum supported ICC clock frequency is 4.8 MHz.
18	*bNumClockSupported*	1		Does not support manual setting of clock frequency.
19	*dwDataRate*	4		Default ICC I/O data rate is 12918 bps.
23	*dwMaxDataRate*	4		Maximum supported ICC I/O data rate is 826 Kbps.
27	*bNumDataRatesSupported*	1		Does not support manual setting of data rates.
28	*dwMaxIFSD*	4		Maximum IFSD supported by ACR39 for protocol T=1 is 247.
32	*dwSynchProtocols*	4		ACR39 does not support synchronous card.
36	*dwMechanical*	4		ACR39 does not support special mechanical characteristics.

Offset	Field	Size	Value	Description
40	*dwFeatures*	4		ACR39 supports the following features: • Automatic ICC clock frequency change according to parameters • Automatic baud rate change according to frequency and FI, DI parameters • TPDU level change with ACR39
44	*dwMaxCCIDMessageLength*	4		Maximum message length accepted by ACR39 is 271 bytes.
48	*bClassGetResponse*	1		Insignificant for TPDU level exchanges.
49	*bClassEnvelope*	1		Insignificant for TPDU level exchanges.
50	*wLCDLayout*	2		No LCD.
52	*bPINSupport*	1		With PIN Verification.
53	*bMaxCCIDBusySlots*	1		Only 1 slot can be simultaneously busy.

7.1. CCID Bulk-OUT Messages

ACR39 shall follow the CCID Bulk-OUT Messages as specified in CCID Rev 1.0 Section 4.1. In addition, this specification defines some extended commands for operating additional features.

This section lists the CCID Bulk-OUT Messages to be supported by ACR39.

7.1.1. PC_to_RDR_IccPowerOn

This command activates the card slot and returns the ATR from the card.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	62h	
1	*dwLength*	4	00000000h	Size of extra bytes of this message.
2	*bSlot*	1		Identifies the slot number for this command.
5	*bSeq*	1		Sequence number for command.
6	*bPowerSelect*	1		Voltage that is applied to the ICC: 00h – Automatic Voltage Selection 01h – 5 V 02h – 3 V
7	*abRFU*	2		Reserved for future use.

The response to this message is the *RDR_to_PC_DataBlock* message and the data returned is the Answer-to-Reset (ATR) data.

7.1.2. PC_to_RDR_IccPowerOff

This command deactivates the card slot.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	63h	
1	*dwLength*	4	00000000h	Size of extra bytes of this message.
5	*bSlot*	1		Identifies the slot number for this command.
6	*bSeq*	1		Sequence number for command.
7	*abRFU*	3		Reserved for future use.

The response to this message is the *RDR_to_PC_SlotStatus* message.

7.1.3. PC_to_RDR_GetSlotStatus

This command gets the current status of the slot.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	65h	
1	*dwLength*	4	00000000h	Size of extra bytes of this message.
5	*bSlot*	1		Identifies the slot number for this command.
6	*bSeq*	1		Sequence number for command.
7	*abRFU*	3		Reserved for future use.

The response to this message is the *RDR_to_PC_SlotStatus* message.

7.1.4. PC_to_RDR_XfrBlock

This command transfers data block to the ICC.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	6Fh	
1	*dwLength*	4		Size of *abData* field of this message.
5	*bSlot*	1		Identifies the slot number for this command.
6	*bSeq*	1		Sequence number for command.
7	*bBWI*	1		Used to extend the CCIDs Block Waiting Timeout for this current transfer. The CCID will timeout the block after “this number multiplied by the Block Waiting Time” has expired.
8	*wLevelParameter*	2	0000h	RFU (TPDU exchange level).
10	*abData*	Byte array		Data block sent to the CCID. Data is sent “as is” to the ICC (TPDU exchange level).

The response to this message is the *RDR_to_PC_DataBlock* message.

7.1.5. PC_to_RDR_GetParameters

This command gets the slot parameters.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	6Ch	
1	*dwLength*	4	00000000h	Size of extra bytes of this message.
5	*bSlot*	1		Identifies the slot number for this command.
6	*bSeq*	1		Sequence number for command.
7	*abRFU*	3		Reserved for future use.

The response to this message is the *RDR_to_PC_Parameters* message.

7.1.6. PC_to_RDR_ResetParameters

This command resets the slot parameter to default value.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	6Dh	
1	*dwLength*	4	00000000h	Size of extra bytes of this message.
5	*bSlot*	1		Identifies the slot number for this command.
6	*bSeq*	1		Sequence number for command.
7	*abRFU*	3		Reserved for future use.

The response to this message is the *RDR_to_PC_Parameters* message.

7.1.7. PC_to_RDR_SetParameters

This command sets the slot parameters.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	61h	
1	*dwLength*	4		Size of extra bytes of this message.
5	*bSlot*	1		Identifies the slot number for this command.
6	*bSeq*	1		Sequence number for command.
7	*bProtocolNum*	1		Specifies what protocol data structure follows: 00h – Structure for protocol T=0 01h – Structure for protocol T=1 The following values are reserved for future use: 80h – Structure for 2-wire protocol 81h – Structure for 3-wire protocol 82h – Structure for I2C protocol
8	*abRFU*	2		Reserved for future use.
10	*abProtocolDataStructure*	Byte array		Protocol Data Structure.

Protocol Data Structure for Protocol T=0 (*dwLength*=00000005h)

Offset	Field	Size	Value	Description
10	*bmFindexDindex*	1		B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor. B3-0 – DI – Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor.
11	*bmTCCKST0*	1		B0 – 0b, B7-2 – 000000b B1 – Convention used (b1=0 for direct, b1=1 for inverse) ***Note:*** *The CCID ignores this bit.*
12	*bGuardTimeT0*	1		Extra guard time between two characters. Add 0 to 254 etu to the normal guard time of 12 etu. FFh is the same as 00h.
13	*bWaitingIntegerT0*	1		WI for T=0 used to define WWT.

Offset	Field	Size	Value	Description
14	*bClockStop*	1		ICC Clock Stop Support: 00h – Stopping the Clock is not allowed 01h – Stop with Clock signal Low 02h – Stop with Clock signal High 03h – Stop with Clock signal either High or Low

Protocol Data Structure for Protocol T=1 (*dwLength*=00000007h)

Offset	Field	Size	Value	Description
10	*bmFindexDindex*	1		B7-4 – FI – Index into the Table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor. B3-0 – DI – Index into the Table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor.
11	*bmTCCKST1*	1		B7-2 – 000100b B0 – Checksum type (b0=0 for LRC, b0=1 for CRC) B1 – Convention used (b1=0 for direct, b1=1 for inverse) ***Note:*** *The CCID ignores this bit.*
12	*bGuardTimeT1*	1		Extra guard time (0 to 254 etu between two characters). If value is FFh, then the guard time is reduced by 1 etu.
13	*bWaitingIntegerT1*	1		B7-4 – BWI values 0-9h valid B3-0 – CWI values 0-Fh valid
14	*bClockStop*	1		ICC Clock Stop Support: 00h – Stopping the Clock is not allowed 01h – Stop with Clock signal Low 02h – Stop with Clock signal High 03h – Stop with Clock signal either High or Low
15	*bIFSC*	1		Size of negotiated IFSC.
16	*bNadValue*	1	00h	Only support NAD=00h.

The response to this message is the *RDR_to_PC_Parameters* message.

7.2. CCID Bulk-IN Messages

The Bulk-IN Messages are used in response to the Bulk-OUT Messages. ACR39 shall follow the CCID Bulk-IN Messages as specified in CCID Rev 1.0 Section 4.

This section lists the CCID Bulk-IN Messages to be support by ACR39.

7.2.1. RDR_to_PC_DataBlock

This message is sent by ACR39 in response to the *PC_to_RDR_IccPowerOn*, and *PC_to_RDR_XfrBlock* messages.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	80h	Indicates that a data block is being sent from the CCID.
1	*dwLength*	4		Size of extra bytes of this message.
5	*bSlot*	1		Same value as in Bulk-OUT message.
6	*bSeq*	1		Same value as in Bulk-OUT message.
7	*bStatus*	1		Slot status register as defined in CCID Rev 1.0 Section 4.2.1.
8	*bError*	1		Slot status register as defined in CCID Rev 1.0 Section 4.2.1.
9	*bChainParameter*	1	00h	RFU (TPDU exchange level).
10	*abData*	Byte array		This field contains the data returned by the CCID.

7.2.2. RDR_to_PC_SlotStatus

This message is sent by the ACR39 in response to *PC_to_RDR_IccPowerOff*, and *PC_to_RDR_GetSlotStatus* messages.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	81h	
1	*dwLength*	4	00 00 00 00h	Size of extra bytes of this message.
5	*bSlot*	1		Same value as in Bulk-OUT message.
6	*bSeq*	1		Same value as in Bulk-OUT message.
7	*bStatus*	1		Slot status register as defined in CCID Rev 1.0 Section 4.2.1.
8	*bError*	1		Slot status register as defined in CCID Rev 1.0 Section 4.2.1.
9	*bClockStatus*	1		Value: 00h – Clock running 01h – Clock stopped in state L 02h – Clock stopped in state H 03h – Clock stopped in an unknown state All other values are RFU.

7.2.3. RDR_toPC_Parameters

This message is sent by ACR39 in response to *PC_to_RDR_GetParameters*, *PC_to_RDR_ResetParameters*, and *PC_to_RDR_SetParameters* messages.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	82h	
1	*dwLength*	4		Size of extra bytes of this message.
5	*bSlot*	1		Same value as in Bulk-OUT message.
6	*bSeq*	1		Same value as in Bulk-OUT message.
7	*bStatus*	1		Slot status register as defined in CCID Rev 1.0 Section 4.2.1.
8	*bError*	1		Slot error register as defined in CCID Section 4.2.1 and
9	*bProtocolNum*	1		Specifies what protocol data structure follows: 00h – Structure for protocol T=0 01h – Structure for protocol T=1 The following values are reserved for future use: 80h – Structure for 2-wire protocol 81h – Structure for 3-wire protocol 82h – Structure for I2C protocol
10	*abProtocolDataStructure*	Byte array		Protocol Data Structure.

8.0. Memory Card Command Set

Memory cards can be accessed via *PC_to_RDR_XfrBlock* command. All functions of memory cards are mapped into pseudo-APDUs.

8.1. Memory Card – 1, 2, 4, 8, and 16 kilobit I2C Card

8.1.1. SELECT_CARD_TYPE

This command is used to power down and up the selected card in the card reader, and then performs a card reset.

Note: *This command can only be used after the logical smart card reader communication has been established using the SCardConnect() API. For details of SCardConnect() API, please refer to PC/SC Specifications.*

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	01h

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.1.2. SELECT_PAGE_SIZE

This command is used to select the page size to read the smart card. The default value is 8-byte page write. It will reset to default value whenever the card is removed or the reader is powered off.

Command format (*abdata* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Page Size
FFh	01h	00h	00h	01h	

Where:

Page size	= 03h for 8-byte page write
	= 04h for 16-byte page write
	= 05h for 32-byte page write
	= 06h for 64-byte page write
	= 07h for 128-byte page write

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.1.3. READ_MEMORY_CARD

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh	B0h			

Where:

Byte Address	Memory address location of the memory card.
MEM_L	Length of data to be read from the memory card.

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

BYTE 1	…	BYTE N	SW1	SW2

Where:

BYTE x	Data read from the memory card.
SW1 SW2	= 90 00h if no error.

8.1.4. WRITE_MEMORY_CARD

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU							
CLA	INS	Byte Address		MEM_L	BYTE 1	…	BYTE N
		MSB	LSB				
FFh	D0h						

Where:

Byte Address	Memory address location of the memory card.
MEM_L	Length of data to be read from the memory card.
BYTE X	Data to be written to the memory card.

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.2. Memory Card – 32, 64, 128, 256, 512, and 1024 kilobit I2C Card

8.2.1. SELECT_CARD_TYPE

This command is used to power down and up the selected card in the card reader, and then performs a card reset.

Note: *This command can only be used after the logical smart card reader communication has been established using the SCardConnect() API. For details of SCardConnect() API, please refer to PC/SC Specifications.*

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	02h

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.2.2. SELECT_PAGE_SIZE

This command is used to select the page size to read the smart card. The default value is 8-byte page write. It will reset to default value whenever the card is removed or the reader is powered off.

Command format (*abdata* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Page Size**
FFh	01h	00h	00h	01h	

Where:

Data	TPDU to be sent to the card.
Page size	= 03h for 8-byte page write
	= 04h for 16-byte page write
	= 05h for 32-byte page write
	= 06h for 64-byte page write
	= 07h for 128-byte page write

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.2.3. READ_MEMORY_CARD

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh				

Where:

INS	= B0h for 32, 64, 128, 256, and 512 kilobit iic card
	= 1011 000*b for 1024 kilobit iic card
	where * is the MSB of the 17 bit addressing
Byte Address	Memory address location of the memory card.
MEM_L	Length of data to be read from the memory card.

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

BYTE 1	...	BYTE N	SW1	SW2

Where:

BYTE x	Data read from memory card.
SW1 SW2	= 90 00h if no error.

8.2.4. WRITE_MEMORY_CARD

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU							
CLA	INS	Byte Address		MEM_L	BYTE 1	...	BYTE N
		MSB	LSB				
FFh							

Where:

INS	= D0h for 32, 64, 128, 256, and 512 kilobit iic card
	= 1101 000*b for 1024 kilobit iic card
	where * is the MSB of the 17 bit addressing
Byte Address	Memory address location of the memory card.
MEM_L	Length of data to be read from the memory card.

BYTE X	Data to be written to the memory card.

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.3. Memory Card – SLE4418/SLE4428/SLE5518/SLE5528

8.3.1. SELECT_CARD_TYPE

This command is used to power down and up the selected card in the card reader, and then performs a card reset.

Note: *This command can only be used after the logical smart card reader communication has been established using the SCardConnect() API. For details of SCardConnect() API, please refer to PC/SC Specifications.*

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	05h

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.3.2. READ_MEMORY_CARD

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	**INS**	**Byte Address**		**MEM_L**
		MSB	**LSB**	
FFh	B0h			

Where:

MSB Byte Address	= 0000 00A_9A_8b is the memory address location of the memory card.
LSB Byte Address	**=** $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b is the memory address location of the memory card.
MEM_L	Length of data to be read from the memory card.

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

BYTE 1	...	BYTE N	SW1	SW2

Where:

BYTE x	Data read from memory card.
SW1 SW2	= 90 00h if no error.

8.3.3. READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (Only SLE4428 and SLE5528)

This command is used to read the presentation error counter for the secret code.

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	B1h	00h	00h	03h

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

ERRCNT	DUMMY 1	DUMMY 2	SW1	SW2

Where:

ERRCNT	The value of the presentation error counter. FFh indicates the last verification is correct. 00h indicates the password is locked (exceeded maximum number of retries). Other values indicate the last verification failed.
DUMMY	Two bytes dummy data read from the card.
SW1 SW2	= 90 00h if no error.

8.3.4. READ_PROTECTION_BIT

Command format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh	B2h			

Where:

MSB Byte Address	= 0000 00A_9A_8b is the memory address location of the memory card.
LSB Byte Address	**=** $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b is the memory address location of the memory card
MEM_L	Length of data to be read from the memory card (in multiple of 8 bits; maximum of 32).

MEM_L = 1 + INT [(number of bits – 1)/8]

For example: To read eight protection bits starting from memory 0010h, the following pseudo-APDU should be issued as:

FF B1 00 10 01h

Response data format (*abData* field in the *RDR_to_PC_DataBlock*)

PROT 1	…	PROT L	SW1	SW2

Where:

PROT y	Bytes containing the protection bits.
SW1 SW2	= 90 00h if no error.

The arrangement of the protection bits in the PROT bytes is as follows:

PROT 1								PROT 2								…							
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	P18	P17

Where:

Px is the protection bit of BYTE x in the response data.

'0' byte is write protected.

'1' byte can be written.

8.3.5. WRITE_MEMORY_CARD

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	....	....	Byte N
		MSB	LSB					
FFh	D0h							

Where:

MSB Byte Address	= 0000 00A_9A_8b is the memory address location of the memory card.
LSB Byte Address	= $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b is the memory address location of the memory card.
MEM_L	Length of data to be written to the memory card.
Byte x	Data to be written to the memory card.

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.3.6. WRITE_PROTECTION_MEMORY_CARD

Each byte specified in the command is used in the card to compare the byte stored in a specified address location. If the data match, the corresponding protection bit is irreversibly programmed to ‘0’.

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	....	....	Byte N
		MSB	LSB					
FFh	D1h							

Where:

MSB Byte Address	= 0000 00A_9A_8b is the memory address location of the memory card.
LSB Byte Address	= $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b is the memory address location of the memory card.
MEM_L	Length of data to be written to the memory card.
Byte x	Byte values to be compared with the data in the card starting at *Byte Address*. BYTE 1 is compared with the data at Byte Address; BYTE N is compared with the data at (Byte Address + N - 1).

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.3.7. PRESENT_CODE_MEMORY_CARD (Only SLE4428 and SLE5528)

This command is used to submit the secret code to the memory card to enable the write operation with the SLE4428 and SLE5528 card, the following actions are executed:

1. Search a ‘1’ bit in the presentation error counter and write the bit to ‘0’.
2. Present the specified code to the card.
3. Try to erase the presentation error counter.

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU						
CLA	INS	P1	P2	MEM_L	CODE	
					Byte 1	Byte 2
FFh	20h	00h	00h	02h		

Where:

CODE Two bytes secret code (PIN).

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2 ErrorCnt
90h	

Where:

SW1	= 90h
SW2 (ErrorCnt)	= Error Counter. FFh indicates successful verification. 00h indicates that the password is locked (or exceeded the maximum number of retries). Other values indicate that current verification has failed.

8.4. Memory Card – SLE4432/SLE4442/SLE5532/SLE5542

8.4.1. SELECT_CARD_TYPE

This command is used to power down and up the selected card in the card reader and performs a card reset.

Note: *This command can only be used after the logical smart card reader communication has been established using the SCardConnect() API. For details of SCardConnect() API, please refer to PC/SC specifications.*

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	06h

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.4.2. READ_MEMORY_CARD

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	B0h	00h		

Where:

Byte Address	= $A_7A_6A_5A_4$ $A_3A_2A_1A_0b$ is the memory address location of the memory card.
MEM_L	Length of data to be read from the memory card.

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

BYTE 1	…	BYTE N	PROT 1	PROT 2	PROT 3	PROT 4	SW1	SW2

Where:

BYTE x	Data read from memory card.
PROT y	Bytes containing the protection bits from protection memory.
SW1 SW2	= 90 00h if no error.

The arrangement of the protection bits in the PROT bytes is as follows:

PROT 1								PROT 2								…							
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	P18	P17

Where:

Px is the protection bit of BYTE x in the response data.

‘0’ byte is write protected.

‘1’ byte can be written.

8.4.3. READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (Only SLE4442 and SLE5542)

This command is used to read the presentation error counter for the secret code.

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	B1h	00h	00h	04h

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

ERRCNT	DUMMY 1	DUMMY 2	DUMMY 3	SW1	SW2

Where:

ERRCNT	The value of the presentation error counter. 07h indicates that the last verification is correct. 00h indicates that the password is locked (exceeded the maximum number of retries). Other values indicate that the last verification has failed.
DUMMY	Three bytes dummy data read from the card.
SW1 SW2	= 90 00h if no error.

8.4.4. READ_PROTECTION_BITS

This command is used to read the protection bits for the first 32 bytes.

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	B2h	00h	00h	04h

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

PROT 1	PROT 2	PROT 3	PROT 4	SW1	SW2

Where:

PROT y	Bytes containing the protection bits from protection memory.
SW1 SW2	= 90 00h if no error.

The arrangement of the protection bits in the PROT bytes is as follows:

PROT 1								PROT 2								…							
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	P18	P17

Where:

Px is the protection bit of BYTE x in the response data.

'0' byte is write protected.

'1' byte can be written.

8.4.5. WRITE_MEMORY_CARD

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	....	....	Byte N
FFh	D0h	00h						

Where:

Byte Address	= $A_7A_6A_5A_4$ $A_3A_2A_1A_0b$ is the memory address location of the memory card.
MEM_L	Length of data to be written to the memory card.
Byte x	Data to be written to the memory card.

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.4.6. WRITE_PROTECTION_MEMORY_CARD

Each byte specified in the command is internally in the card compared with the byte stored at the specified address and if the data match, the corresponding protection bit is irreversibly programmed to '0'.

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	....	....	Byte N
FFh	D1h	00h						

Where:

Byte Address	= 000A_4 $A_3A_2A_1A_0$b (00h to 1Fh) is the protection memory address location of the memory card.
MEM_L	Length of data to be written to the memory card.
Byte x	Byte values to be compared with the data in the card starting at Byte Address. BYTE 1 is compared with the data at Byte Address; BYTE N is compared with the data at (Byte Address + N - 1).

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

8.4.7. PRESENT_CODE_MEMORY_CARD (Only SLE4442 and SLE5542)

This command is used to submit the secret code to the memory card to enable the write operation with the SLE4442 and SLE5542 card, the following actions are executed:

1. Search a '1' bit in the presentation error counter and write the bit to '0'.
2. Present the specified code to the card.
3. Try to erase the presentation error counter.

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU							
CLA	INS	P1	P2	MEM_L	CODE		
					Byte 1	Byte 2	Byte 3
FFh	20h	00h	00h	03h			

Where:

CODE Three bytes secret code (PIN).

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2 ErrorCnt
90h	

Where:

SW1	= 90h
SW2 (ErrorCnt)	= Error Counter. 07h indicates that the verification is correct. 00h indicates the password is locked (exceeded the maximum number of retries). Other values indicate that the current verification has failed.

8.4.8. CHANGE_CODE_MEMORY_CARD (Only SLE4442 and SLE5542)

This command is used to write the specified data as new secret code in the card. The current secret code must have been presented to the card with the *PRESENT_CODE* command prior to the execution of this command.

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU							
CLA	INS	P1	P2	MEM_L	CODE		
					Byte 1	Byte 2	Byte 3
FFh	D2h	00h	01h	03h			

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error.

9.0. Other commands accessed via PC_to_RDR_XfrBlock

9.1. GET_READER_INFORMATION

This command is used to return the firmware revision number of the ACR39 reader.

Note: *This command can only be used after the logical smart card reader communication has been established using the SCardConnect() API. For details of SCardConnect() API, please refer to PC/SC specifications.*

Command Format (*abData* field in the *PC_to_RDR_XfrBlock*)

Pseudo-APDU				
CLA	**INS**	**P1**	**P2**	**Le**
FFh	09h	00h	00h	11h

Response Data Format (*abData* field in the *RDR_to_PC_DataBlock*)

FIRMWARE										

Where:

FIRMWARE	11 bytes data for firmware version.

Appendix A. Response Error Codes

The following table summarizes the possible error codes returned by the ACR39:

Error Code	Status
FFh	SLOTERROR_CMD_ABORTED
FEh	SLOTERROR_ICC_MUTE
FDh	SLOTERROR_XFR_PARITY_ERROR
FCh	SLOTERROR_XFR_OVERRUN
FBh	SLOTERROR_HW_ERROR
F8h	SLOTERROR_BAD_ATR_TS
F7h	SLOTERROR_BAD_ATR_TCK
F6h	SLOTERROR_ICC_PROTOCOL_NOT_SUPPORTED
F5h	SLOTERROR_ICC_CLASS_NOT_SUPPORTED
F4h	SLOTERROR_PROCEDURE_BYTE_CONFLICE
F3h	SLOTERROR_DEACTIVATED_PROTOCOL
F2h	SLOTERROR_BUSY_WITH_AUTO_SEQUENCE
E0h	SLOTERROR_CMD_SLOT_BUSY

Table 3: Response Error Codes