Advanced Card Systems Ltd.
Card & Reader Technologies

ACR3x Mobile Card Reader

Reference Manual V1.01

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

Table of Contents

List of Figures

List of Tables

1.0. Introduction

The ACR3x Mobile Card Reader serves as an interface for the communication between a mobile device and a magnetic/contact/contactless card. Different types of cards have different commands and communication protocols, and the ACR3x establishes a uniform interface from the mobile device to the card.

The ACR3x is connected to the mobile device through a 3.5 mm audio jack interface. Through this, the ACR3x will read information from the cards through the decoder on the reader which will be sent to the mobile device, such as smartphone or tablet. Furthermore, as a way to enhance security, the card information is encrypted using the AES-128 encryption algorithm before it will be sent to the backend server.

This document describes the hardware and software design of the ACR3x and the list of commands it uses to communicate with the mobile device.

1.1. Definitions of Terms

Abbreviation	Description
ACS Secret Key	Key used to perform Master Reset authentication. This key cannot be modified through command messages and is hard coded in the firmware. This key must be kept securely by ACS only.
AES	Advanced Encryption Standard
AES Key	The key used to encrypt the magnetic stripe track data using AES-128 CBC cipher mode. This key can be modified by the customer.
Custom ID	10 bytes of identification code set by customer. This ID can be modified by the customer.
Customer Master Key	The key being kept by the customer to perform authentication with ACR3x before the injection of AES Key, new Customer Master Key, Custom ID and DUKPT option, as well as performing DUKPT initialization. This key can be modified by the customer
Device ID	8 bytes of unique identification code of the MCU used in ACR3x. Customer can use this ID to derive the Custom ID or DUKPT initialization data. This ID cannot be modified by any means and is hard coded inside the MCU by the MCU manufacturer.
Master Reset	This term is equivalent to factory reset. By performing a Master Reset, all the data stored in the flash memory will be erased and set to default values
MReset Session Key	Key being created uniquely after each success mutual authentication for Master Reset
Session Key	Key being created uniquely after each success mutual authentication for sensitive data injection
TDES	Triple Data Encryption Standard

Table 1: Definitions of Terms

2.0. Features

2.1. ACR31

- 3.5 mm Audio Jack Interface
- Powered by a CR2016 battery
- Reads up to two tracks of card data
- Capable of bi-directional reading
- Supports AES-128 encryption algorithm
- Supports DUKPT Key Management System
- Magnetic Stripe Card Reader:
  - Supports ISO 7810/7811 magnetic cards
  - Supports Hi-coercivity and Low-coercivity magnetic cards
  - Supports JIS1 and JIS2
- Supports Android™ 2.3 and above*
- Supports iOS 5.0 and above*
- Compliant with the following standards:
  - CE
  - FCC
  - VCCI
  - RoHS
  - REACH

***Visit** www.acs.com.hk for the list of supported devices.*

2.2. ACR32

- 3.5 mm Audio Jack Interface
- USB Powered (PC-linked Mode):
  - USB 2.0 Full Speed Interface
  - CCID Compliance
  - Application Programming Interface
    - Supports PC/SC*
    - Supports CT-API (through wrapper on top of PC/SC)
- Battery Powered:
  - Powered by a Lithium-ion battery (rechargeable through PC-linked mode)
- Smart Card Reader:
  - 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
- Magnetic Stripe Card Reader:
  - Reads up to two tracks of card data
  - Capable of bi-directional reading
  - Supports AES-128 encryption algorithm
  - Supports DUKPT Key Management System
  - Supports ISO 7810/7811 magnetic cards
  - Supports Hi-coercivity and Low-coercivity magnetic cards
  - Supports JIS1 and JIS2
- Supports Android™ 2.0 and above**
- Supports iOS 5.0 and above**
- Compliant with the following standards:
  - EN60950/IEC 60950
  - ISO 7816
  - CE
  - FCC
  - VCCI
  - PC/SC
  - CCID
  - EMV™ Contact Level 1
  - Microsoft® WHQL
  - RoHS 2
  - REACH

** Applicable under PC-linked mode.*
*** Visit www.acs.com.hk for the list of supported devices.*

2.3. ACR35

- 3.5 mm Audio Jack Interface
- Powered by a Lithium-ion battery (rechargeable through USB cable)
- Smart Card Reader:
  - Built-in antenna for contactless tag access, with reading distance of up to 50 mm (depending on tag type)
  - Supports ISO 14443 Part 4 Type A and B cards
  - Supports MIFARE®
  - Supports FeliCa
  - Supports ISO 18092 Tags (NFC Tags)*
  - Built-in anti-collision feature (only one tag is accessed at any time)
  - NFC Support:
    - Card reader/writer mode
- Magnetic Stripe Card Reader:
  - Reads up to two tracks of card data
  - Capable of bi-directional reading
  - Supports AES-128 encryption algorithm
  - Supports DUKPT Key Management System
  - Supports ISO 7810/7811 magnetic cards
  - Supports Hi-coercivity and Low-coercivity magnetic cards
  - Supports JIS1 and JIS2
- Supports Android™ 2.0 and above**
- Supports iOS 5.0 and above**
- Compliant with the following standards:
  - ISO 18092
  - ISO 14443
  - CE
  - FCC
  - VCCI
  - RoHS 2
  - REACH

**Topaz type is excluded. Please contact ACS for more details.*
*** Visit www.acs.com.hk for the list of supported devices.*

3.0. Supported Cards

3.1. Magnetic Cards

ACR3x operates with ISO 7810/7811 magnetic cards with Hi-coercivity and Low-coercivity.

3.2. MCU Cards

ACR32 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 ACR32, 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 ACR32 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).

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

3.3. Memory-based Smart Cards

ACR32 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 secure memory IC with password and authentication, including:
  - Atmel®: AT88SC153 and AT88SC1608
- Cards with intelligent 1 kilobytes EEPROM with write-protect function, including:
  - Infineon®: SLE4418, SLE4428, SLE5518 and SLE5528
- Cards with intelligent 256 bytes EEPROM with write-protect function, including:
  - Infineon®: SLE4432, SLE4442, SLE5532 and SLE5542
- Cards with ‘104’ type EEPROM non-reloadable token counter cards, including:
  - Infineon®: SLE4406, SLE4436, SLE5536 and SLE6636
- Cards with Intelligent 416-Bit EEPROM with internal PIN check, including:
  - Infineon®: SLE4404
- Cards with Security Logic with Application Zone(s), including:
  - Atmel®: AT88SC101, AT88SC102 and AT88SC1003

3.4. Contactless Cards

ACR35 works with various contactless cards and tags such as:

- ISO 14443 Type A cards
- ISO 14443 Type B cards
- ISO/IEC 18092 (NFC) cards
- MIFARE® Classic 1K/4K
- FeliCa
- MIFARE Ultralight®
- MIFARE Ultralight® C
- MIFARE® DESFire® EV1

4.0. System Block Design

4.1. ACR31

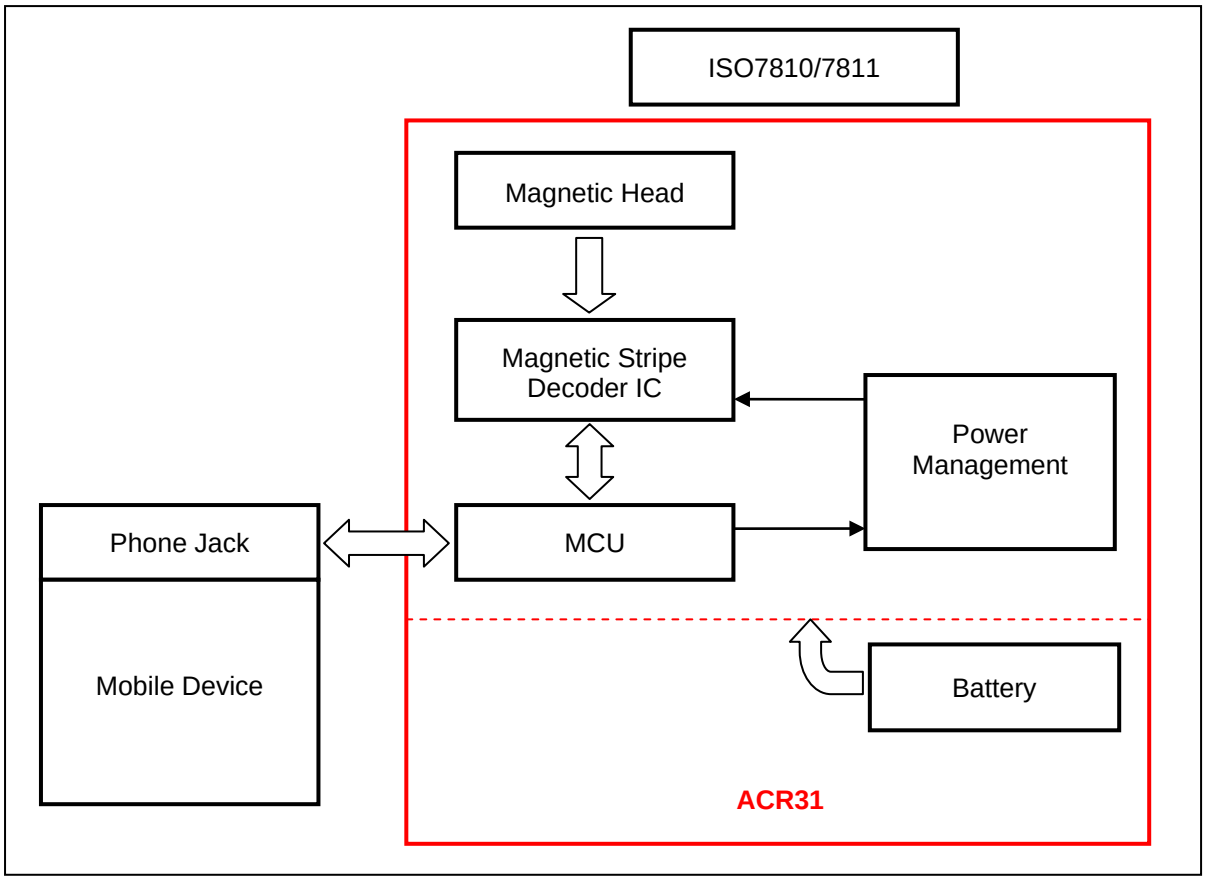

Figure 1: ACR31 Architecture

4.2. ACR32

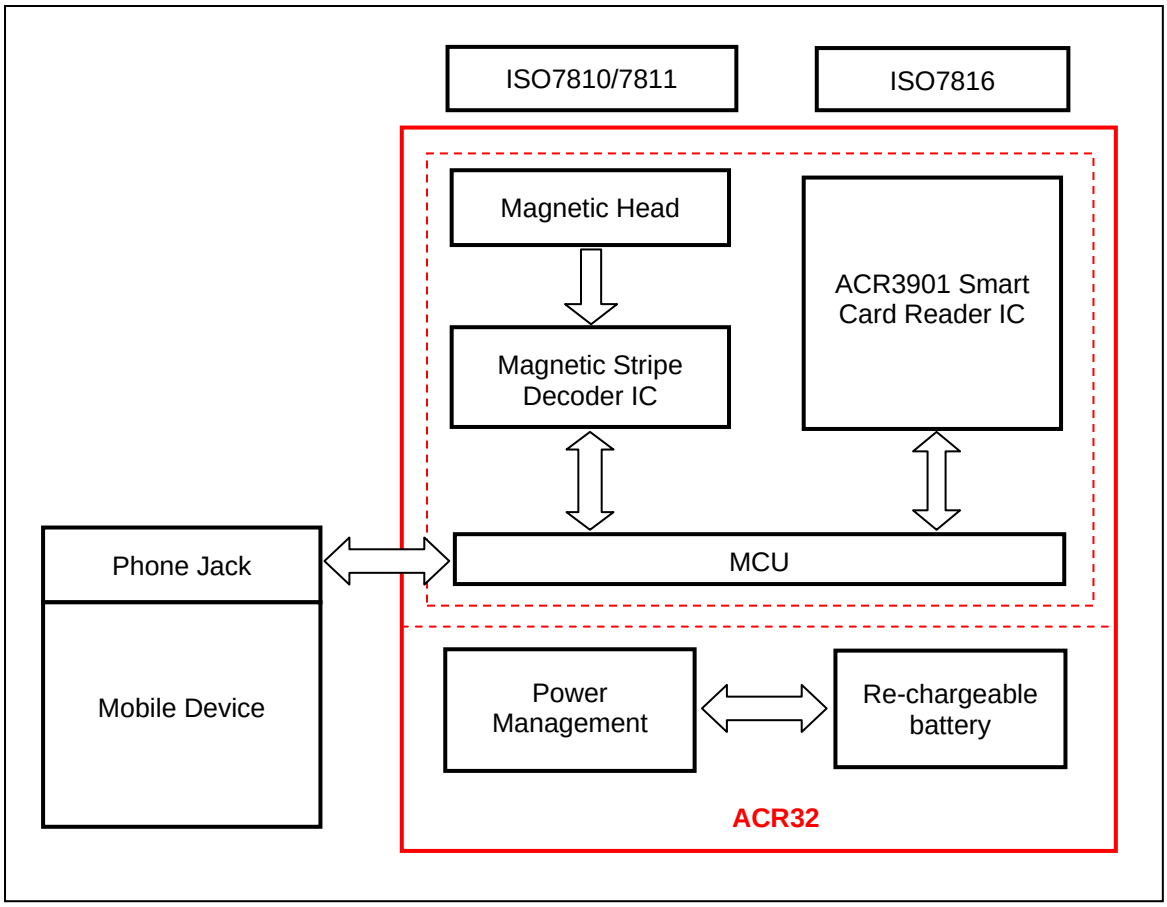

Figure 2: ACR32 Architecture

4.3. ACR35

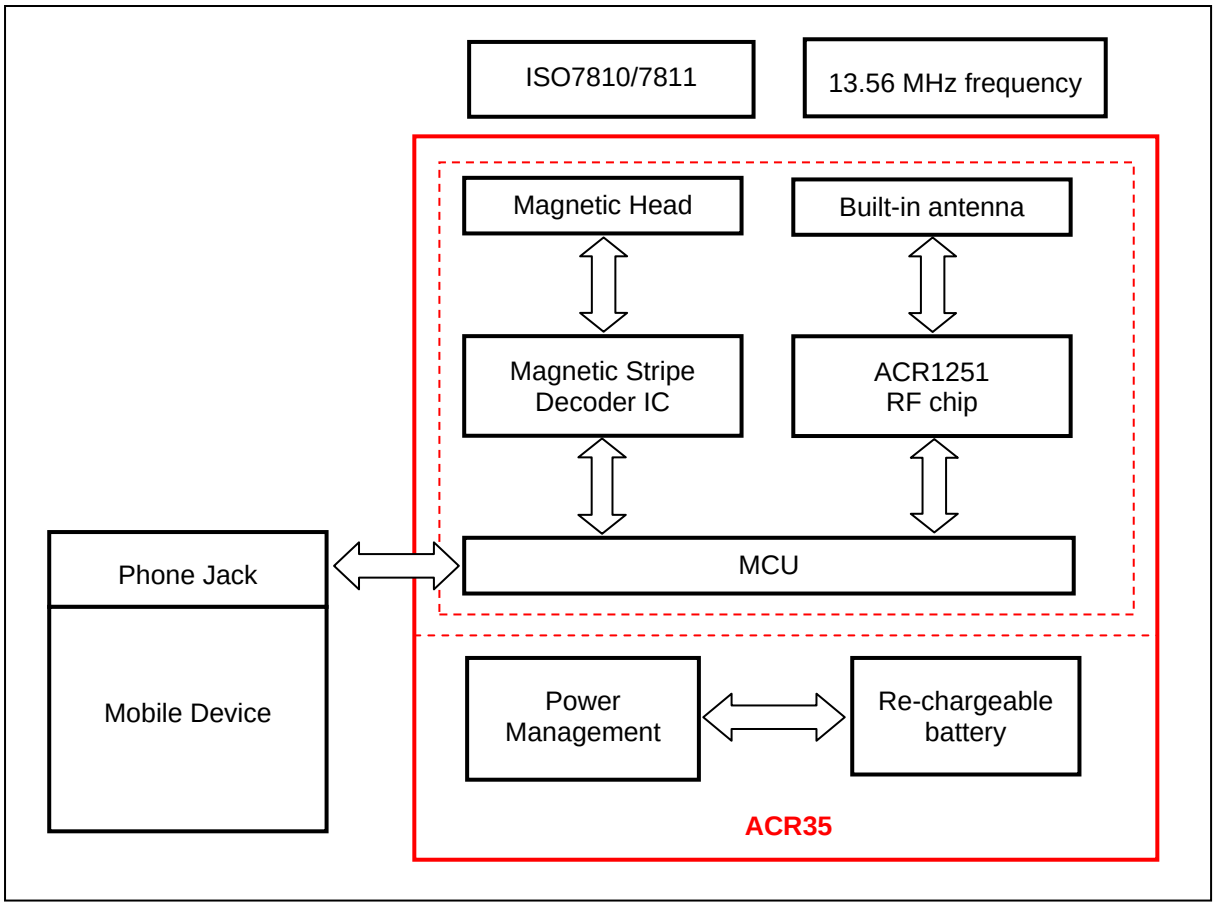

Figure 3: ACR35 Architecture

5.0. Hardware Design

5.1. Battery

ACR31 is using a CR2016 battery which has a capacity of 90 mAh. On the other hand, ACR32 and ACR35 are using a rechargeable Lithium-ion battery which has a capacity of 200 mAh.

5.2. Status LED

The different LED colors indicate the different states of ACR32 and ACR35, where:

- **Green LED**

  Operational

- **Red LED**

  Battery status

5.3. Micro USB Interface

The Micro USB port is to be used to connect the ACR32 and ACR35 to the computer as battery charging port. This port is also to be used in order for the ACR32 to act as a PC-linked reader.

5.4. Audio Channel

5.4.1. Communication Parameters

ACR3x is connected to a mobile device through Audio Channel.

Pin	Signal	Function
1	Left	Transmit the data to ACR3x
2	Right	Wake up device signal
3	GND	GND
4	MIC	Transmit the data to smart phone

Table 2: 3.5 mm Audio Socket Wiring

5.5. Magnetic Stripe Card Interface

ACR3x can read any magnetic stripe card that conforms to ISO 7810/7811 standards. ISO 7810 specifies the physical characteristics of the card, while ISO 7811 specifies the recording technique used in identification cards.

High-coercivity (Hi-Co) magnetic stripes are typically black in color and are encoded with a stronger magnetic field (2750 Oersted). This makes Hi-Co cards more durable because the data encoded on the stripes are less likely to be unintentionally erased when exposed to an outside magnetic field. When swiped across the magnetic head, Hi-Co magnetic stripes can induce larger signal pulses and are more easily detected and decoded.

Low-coercivity (Lo-Co) magnetic stripes are generally brown in color and are encoded with lower magnetic field intensity (300 Oersted). They will induce small signal pulses compared to Hi-Co cards when swiped across the magnetic head. As a result, the Signal-Noise (S/N) ratio is relatively low and they are more vulnerable to noise interference. A more sophisticated hardware support and signal processing algorithm are needed to decode the signal correctly.

Since the magnetic fields of Hi-Co and Lo-Co cards are different, a magnetic stripe decoder IC with automatic gain control can be used in the design to cater these two types of cards.

5.6. Smart Card Interface

The interface between the ACR32 and the inserted smart card follows the specification of ISO 7816-3 with certain restrictions or enhancements to increase the practical functionality of ACR32

5.6.1. Smart Card Power Supply VCC (C1)

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

5.6.2. 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 ACR32. The electrical specifications of this contact are identical to those of the signal RST (at contact C2).

5.6.3. Card Type Selection

The controlling PC must always select the card type through the proper command sent to the ACR32 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.

5.6.4. 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 MHz is applied to the CLK signal (C3).

5.6.5. Card Tearing Protection

The ACR32 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 ACR32 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 ACR32 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. Communication Protocol

ACR3x is a slave device and almost all operations are initiated by the mobile device. The mobile device that sends the command is carried out in the form of successive command request-response exchange. Additionally, the new request message should wait until previous response message has been received

ACR3x will communicate with a mobile device through its audio jack interface. The communication channel is bi-directional, with the reader sending data to the mobile device through the MIC pin of the audio jack while the mobile device sending commands to the reader through the Right-Channel of the audio jack.

While it is not operating, the ACR3x will remain in deep sleep mode. Upon receiving a wake up signal from the mobile device through the Left-Channel of the audio jack, the ACR3x will wake up and send back an acknowledgement signal to the mobile device. ACR3x will then wait for the swipe of the magnetic stripe card within a timeout limit. After successfully obtaining the data from the swiped card, the ACR3x will perform AES-128 encryption on the received card data and send back the encrypted data to the mobile device in communication. If the reader fails to obtain a card swipe or command message within the timeout limit, the ACR3x will send back the corresponding status to the mobile device. After which, ACR3x will go back to deep sleep mode to save battery power.

Before the communication protocol between the ACR3x and the mobile device employ a direct signal feeding, the signal received from the ACR3x will be passing through a DC offset cancellation filter. The data to be transmitted will be encoded using the Manchester coding scheme (conforms to IEEE 802.3), with the clock frequency used in the Manchester coding scheme being set at 10 kHz. Since the data transmission speed in the Manchester coding scheme always matches the clock speed, a maximum baud rate of around 10 Kbps could be achieved.

The signal interpretation on the mobile device and ACR3x is based on sampling the corresponding input waveforms. The sampling frequency should be at least double the clock frequency (Nyquist rate) used in the Manchester coding scheme. After sampling the signals, the data encoded in the signals could be received by determining the logical zero-crossing time.

7.0. Application Programming Interface

Please refer to the HTML files included in the *ACR3x Android Library* or *ACR3x iOS Library*.

The libraries may be downloaded from ACS website.

8.0. Contact Card Commands

This section contains the memory card command set for ACR32.

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

8.1.1. SELECT_CARD_TYPE

This command powers down and up the selected card inserted 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 specification.*

Command Format

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

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.1.2. SELECT_PAGE_SIZE

This command chooses 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

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

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.1.3. READ_MEMORY_CARD

Command Format

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

BYTE 1	...	BYTE N	SW1	SW2

Where:

BYTE x Data read from memory card

SW1 SW2 = 90 00h if no error

8.1.4. WRITE_MEMORY_CARD

Command Format

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 written to the memory card

Byte x Data to be written to the memory card

Response Data Format

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 powers down and up the selected card that is inserted 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

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

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.2.2. SELECT_PAGE_SIZE

This command chooses 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

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

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.2.3. READ_MEMORY_CARD

Command Format

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh				

Where:

INS = B0h for 32, 64, 128, 256, 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

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

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

Where:

INS = D0h for 32, 64, 128, 256, 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 written to the memory card
Byte x	Data to be written to the memory card

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.3. Memory Card – Atmel® AT88SC153

8.3.1. SELECT_CARD_TYPE

This command powers up and down the selected card that is inserted in the card reader and performs a card reset. It will also select the page size to be 8-byte page write.

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

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	03h

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.3.2. READ_MEMORY_CARD

Command Format

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

Where:

INS = B0h for reading zone 00b
= B1h for reading zone 01b
= B2h for reading zone 10b
= B3h for reading zone 11b
= B4h for reading fuse

Byte Address Memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format

BYTE 1	...	BYTE N	SW1	SW2

Where:

BYTE x Data read from memory card

SW1 SW2 = 90 00h if no error

8.3.3. WRITE_MEMORY_CARD

Command Format

Pseudo-APDU							
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**	**Byte 1**	**....**	**Byte n**
FFh		00h					

Where:

INS = D0h for writing zone 00b
= D1h for writing zone 01b
= D2h for writing zone 10b
= D3h for writing zone 11b
= D4h for writing fuse

Byte Address Memory address location of the memory card

MEM_L Length of data to be written to the memory card

MEM_D Data to be written to the memory card

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.3.4. VERIFY_PASSWORD

Command Format

Pseudo-APDU							
CLA	**INS**	**P1**	**P2**	**Lc**	**Pw(0)**	**Pw(1)**	**Pw(2)**
FFh	20h	00h		03h			

Where:

Pw(0),Pw(1),Pw(2) Passwords to be sent to memory card

P2 = 0000 00rpb

where the two bits “rp” indicate the password to compare

r = 0: Write password,

r = 1: Read password,

p: Password set number,

rp = 01 for the secure code.

Response Data Format

SW1	SW2 ErrorCnt
90h	

Where:

SW1 = 90h

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

8.3.5. INITIALIZE_AUTHENTICATION

Command Format

Pseudo-APDU								
CLA	**INS**	**P1**	**P2**	**Lc**	**Q(0)**	**Q(1)**	**…**	**Q(7)**
FFh	84h	00h	00h	08h				

Where:

Q(0),Q(1)…Q(7) Host random number, 8 bytes

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.3.6. VERIFY_AUTHENTICATION

Command Format

Pseudo-APDU								
CLA	**INS**	**P1**	**P2**	**Lc**	**Ch(0)**	**Ch(1)**	**…**	**Ch(7)**
FFh	82h	00h	00h	08h				

Where:

Ch(0),Ch(1)…Ch(7) Host challenge, 8 bytes

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.4. Memory Card – Atmel® AT88C1608

8.4.1. SELECT_CARD_TYPE

This command powers down and up the selected card that is inserted in the card reader and performs a card reset. It will also select the page size to be 16-byte page write.

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

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	04h

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.4.2. READ_MEMORY_CARD

Command Format

Pseudo-APDU				
CLA	**INS**	**Zone Address**	**Byte Address**	**MEM_L**
FFh				

Where:

INS = B0h for reading user zone

= B1h for reading configuration zone or reading fuse

Zone Address = 0000 $0A_{10}A_9A_8$b where A_{10} is the MSB of zone address

= Don't care for reading fuse

Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b is the memory address location of the memory card

= 1000 0000b for reading fuse

MEM_L Length of data to be read from the memory card

Response Data Format

BYTE 1	…	BYTE N	SW1	SW2

Where:

BYTE x Data read from memory card

SW1 SW2 = 90 00h if no error

8.4.3. WRITE_MEMORY_CARD

Command Format

Pseudo-APDU							
CLA	**INS**	**Zone Address**	**Byte Address**	**MEM_L**	**Byte 1**	**…**	**Byte n**
FFh							

Where:

INS = D0h for writing user zone

= D1h for writing configuration zone or writing fuse

Zone Address = 0000 $0A_{10}A_9A_8$b where A_{10} is the MSB of zone address

= Don't care for writing fuse

Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b is the memory address location of the memory card

= 1000 0000b for writing fuse

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

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.4.4. VERIFY_PASSWORD

Command Format

Pseudo-APDU								
CLA	**INS**	**P1**	**P2**	**Lc**	**Data**			
FFh	20h	00h	00h	04h	RP	Pw(0)	Pw(1)	Pw(2)

Where:

Pw(0),Pw(1),Pw(2) Passwords to be sent to memory card

RP = 0000 $rp_2p_1p_0b$

where the four bits "$rp_2p_1p_0$" indicate the password to compare:

r = 0 : Write password,

r = 1 : Read password,

$p_2p_1p_0$: Password set number.

($rp_2p_1p_0$ = 0111 for the secure code)

Response Data Format

SW1	SW2 ErrorCnt
90h	

Where:

SW1 = 90h

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

8.4.5. INITIALIZE_AUTHENTICATION

Command Format

Pseudo-APDU								
CLA	**INS**	**P1**	**P2**	**Lc**	**Q(0)**	**Q(1)**	**…**	**Q(7)**
FFh	84h	00h	00h	08h				

Where:

Byte Address Memory address location of the memory card

Q(0),Q(1)…Q(7) Host random number, 8 bytes

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.4.6. VERIFY_AUTHENTICATION

Command Format

Pseudo-APDU								
CLA	**INS**	**P1**	**P2**	**Lc**	**Q1(0)**	**Q1(1)**	**…**	**Q1(7)**
FFh	82h	00h	00h	08h				

Where:

Byte Address Memory address location of the memory card

Q1(0),Q1(1)…Q1(7) Host challenge, 8 bytes

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

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

8.5.1. SELECT_CARD_TYPE

This command powers down and up the selected card that is inserted 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

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

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.5.2. READ_MEMORY_CARD

Command Format

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

BYTE 1	…	BYTE N	SW1	SW2

Where:

BYTE x Data read from memory card

SW1 SW2 = 90 00h if no error

8.5.3. READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (SLE4428 and SLE5528)

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

Command Format

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

Response Data Format

ERRCNT	DUMMY 1	DUMMY 2	SW1	SW2

Where:

ERRCNT Error Counter. FFh 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 Two bytes dummy data read from the card

SW1 SW2 = 90 00h if no error

8.5.4. READ_PROTECTION_BIT

Command Format

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 protection bits to be read from the card, in multiples of 8 bits. Maximum value is 32.

MEM_L = 1 + INT((number of bits - 1)/8)

For example, to read 8 protection bits starting from memory 0010h, the following pseudo-APDU should be issued:

FF B2 00 10 01h

Response Data Format

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

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

'0' byte is write protected

'1' byte can be written

8.5.5. WRITE_MEMORY_CARD

Command Format

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

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.5.6. 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

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

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.5.7. PRESENT_CODE_MEMORY_CARD (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

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

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.

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

8.6.1. SELECT_CARD_TYPE

This command powers down and up the selected card that is inserted 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

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

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.6.2. READ_MEMORY_CARD

Command Format

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

Where:

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

BYTE 1	…	BYTE N	SW1	SW2

Where:

BYTE x Data read from memory card

SW1 SW2 = 90 00h if no error

8.6.3. READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (SLE4442 and SLE5542)

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

Command Format

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

Response Data Format

ERRCNT	DUMMY 1	DUMMY 2	DUMMY 3	SW1	SW2

Where:

ERRCNT 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

8.6.4. READ_PROTECTION_BITS

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

Command Format

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

Response Data Format

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

8.6.5. WRITE_MEMORY_CARD

Command Format

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

Where:

Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$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

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.6.6. 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

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

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.6.7. PRESENT_CODE_MEMORY_CARD (SLE4442 and SLE5542)

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

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

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.

8.6.8. CHANGE_CODE_MEMORY_CARD (SLE4442 and SLE5542)

This command is used to write the specified data as new secret code in the card.

The current secret code must be presented to the card with the *PRESENT_CODE* command prior to the execution of this command.

Command Format

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

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.7. Memory Card – SLE4406/SLE4436/SLE5536/SLE6636

8.7.1. SELECT_CARD_TYPE

This command powers down and up the selected card that is inserted 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

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	07h

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.7.2. READ_MEMORY_CARD

Command Format

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

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

BYTE 1	…	BYTE N	SW1	SW2

Where:

BYTE x Data read from memory card

SW1 SW2 = 90 00h if no error

8.7.3. WRITE_ONE_BYTE_MEMORY_CARD

This command is used to write one byte to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

Four different WRITE modes are available for this card type, which are distinguished by a flag in the command data field:

a) **Write**

The byte value specified in the command is written to the specified address. This command can be used for writing personalization data and counter values to the card.

b) **Write with carry**

The byte value specified in the command is written to the specified address and the command is sent to the card to erase the next lower counter stage. Thus, this write mode can only be used for updating the counter value in the card.

c) **Write with backup enabled** (SLE4436, SLE5536 and SLE6636 only)

The byte value specified in the command is written to the specified address. This command can be used for writing personalization data and counter values to the card. Backup bit is enabled to prevent data loss when card tearing occurs.

d) **Write with carry and backup enabled** (SLE4436, SLE5536 and SLE6636 only)

The byte value specified in the command is written to the specified address and the command is sent to the card to erase the next lower counter stage. Thus, this write mode can only be used for updating the counter value in the card. Backup bit is enabled to prevent data loss when card tearing occurs.

With all write modes, the byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

The backup mode available in the SLE4436 and SLE5536 card can be enabled or disabled in the write operation.

Command Format

Pseudo-APDU						
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**	**MODE**	**BYTE**
FFh	D0h	00h		02h		

Where:

Byte Address = Memory address location of the memory card

MODE Specifies the write mode and backup option

00h: Write

01h: Write with carry

02h: Write with backup enabled (SLE4436, SLE5536 and SLE6636 only)

03h: Write with carry and with backup enabled (SLE4436, SLE5536 and SLE6636 only)

BYTE Byte value to be written to the card

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.7.4. PRESENT_CODE_MEMORY_CARD

To submit the secret code to the memory card to enable the card personalization mode, the following actions are executed:

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

ACR3901x does not try to erase the presentation counter after the code submission. This must be done by the application software through a separate 'Write with carry' command.

Command Format

Pseudo-APDU								
CLA	INS	P1	P2	MEM_L	CODE			
					ADDR	Byte 1	Byte 2	Byte 3
FFh	20h	00h	00h	04h	09h			

Where:

ADDR Byte address of the presentation counter in the card

CODE Three bytes secret code (PIN)

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.7.5. AUTHENTICATE_MEMORY_CARD (SLE4436, SLE5536 and SLE6636)

To read a card authentication certificate from a SLE5536 or SLE6636 card, the ACR3901x executes the following actions:

1. Select Key 1 or Key 2 in the card as specified in the command.
2. Present the challenge data specified in the command to the card.
3. Generate the specified number of CLK pulses for each bit of authentication data computed by the card.
4. Read 16 bits of authentication data from the card.
5. Reset the card to normal operation mode.

The authentication has to be performed in two steps. The first step is to send the Authentication Certificate to the card. The second step is to get back two bytes of authentication data calculated by the card.

Step 1: Send Authentication Certificate to the Card

Command Format

Pseudo-APDU											
CLA	INS	P1	P2	MEM_L	CODE						
					KEY	CLK_CNT	Byte 1	Byte 2	……	Byte 5	Byte 6
FFh	84h	00h	00h	08h							

Where:

KEY Key to be used for the computation of the authentication certificate:

00h: Key 1 with no cipher block chaining

01h: Key 2 with no cipher block chaining

80h: Key 1 with cipher block chaining (SLE5536 and SLE6636 only)

81h: Key 2 with cipher block chaining (SLE5536 and SLE6636 only)

CLK_CNT Number of CLK pulses to be supplied to the card for the computation of each bit of the authentication certificate. Typical value is 160 clocks (A0)

BYTE 1...6 Card challenge data

Response Data Format

SW1	SW2
61h	02h

Where:

SW1 SW2 = 61 02h if no error, meaning two bytes of authentication data are ready. The authentication data can be retrieved by *Get_Response* command.

Step 2: Get back the Authentication Data (Get_Response)

Command Format

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	C0h	00h	00h	02h

Response Data Format

CERT		SW1	SW2

Where:

CERT 16 bits of authentication data computed by the card. The LSB of BYTE 1 is the first authentication bit read from the card.

SW1 SW2 = 90 00h if no error

8.8. Memory Card – SLE 4404

8.8.1. SELECT_CARD_TYPE

This command powers down and up the selected card that is inserted 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

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	08h

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.8.2. READ_MEMORY_CARD

Command Format

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

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

BYTE 1	…	BYTE N	SW1	SW2

Where:

BYTE x Data read from memory card

SW1 SW2 = 90 00h if no error

8.8.3. WRITE_MEMORY_CARD

This command is used to write data to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

The byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

Command Format

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

Where:

Byte Address = Memory address location of the memory card

MEM_L Length of data to be written to the memory card

BYTE Byte value to be written to the card

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.8.4. ERASE_SCRATCH_PAD_MEMORY_CARD

This command is used to erase the data of the scratch pad memory of the inserted card. All memory bits inside the scratch pad memory will be programmed to the state of '1'.

To erase error counter or user area, please use the *VERIFY_USER_CODE* command as specified in the **Section 8.8.5**.

Command Format

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	D2h	00h		00h

Where:

Byte Address Memory byte address location of the scratch pad

Typical value is 02h

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.8.5. VERIFY_USER_CODE

This command is used to submit User Code (2 bytes) to the inserted card. User Code is to enable the memory access of the card.

The following actions are executed:

1. Present the specified code to the card.
2. Search a '1' bit in the presentation error counter and write the bit to '0'.
3. Erase the presentation error counter. The User Error Counter can be erased when the submitted code is correct.

Command Format

Pseudo-APDU						
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE	
					Byte 1	Byte 2
FFh	20h	04h	08h	02h		

Where:

Error Counter LEN Length of presentation error counter in bits

Byte Address Byte address of the key in the card

CODE 2 bytes User Code

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

= 63 00h if there are no more retries

Note: *After SW1 SW2 = 90 00h has been received, read back the User Error Counter to check if the VERIFY_USER_CODE is correct. If User Error Counter is erased and is equal to "FFh," the previous verification is successful.*

8.8.6. VERIFY_MEMORY_CODE

This command is used to submit Memory Code (4 bytes) to the inserted card. Memory Code is used to authorize the reloading of the user memory, together with the User Code.

The following actions are executed:

1. Present the specified code to the card.
2. Search a '1' bit in the presentation error counter and write the bit to '0'.
3. Erase the presentation error counter. Please note that Memory Error Counter cannot be erased.

Command Format

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	Byte 3	Byte 4
FFh	20h	40h	28h	04h				

Where:

Error Counter LEN Length of presentation error counter in bits

Byte Address Byte address of the key in the card

CODE 4 bytes Memory Code

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

= 63 00h if there are no more retries

Note: *After SW1 SW2 = 90 00h has been received, read back the Application Area can check if the VERIFY_MEMORY_CODE is correct. If all data in Application Area is erased and is equal to "FFh," the previous verification is successful.*

8.9. Memory Card – AT88SC101/AT88SC102/AT88SC1003

8.9.1. SELECT_CARD_TYPE

This command powers down and up the selected card that is inserted 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

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	09h

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.9.2. READ_MEMORY_CARD

Command Format

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

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

BYTE 1	…	BYTE N	SW1	SW2

Where:

BYTE x Data read from memory card

SW1 SW2 = 90 00h if no error

8.9.3. WRITE_MEMORY_CARD

This command is used to write data to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

The byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

Command Format

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

Where:

Byte Address Memory address location of the memory card

MEM_L Length of data to be written to the memory card

BYTE Byte value to be written to the card

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.9.4. ERASE_NON_APPLICATION_ZONE

This command is used to erase the data in Non-Application Zones. The EEPROM memory is organized into 16-bit words. Although erases are performed on single bit, the ERASE operation clears an entire word in the memory. Therefore, performing an ERASE on any bit in the word will clear ALL 16 bits of that word to the state of '1'.

To erase Error Counter or the data in Application Zones, please refer to the following:

1. *ERASE_APPLICATION_ZONE_WITH_ERASE* command as specified in **Section 8.9.5**.
2. *ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE* command as specified in **Section 8.9.6**.
3. *VERIFY_SECURITY_CODE* commands as specified in **Section 8.9.7**.

Command Format

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	D2h	00h		00h

Where:

Byte Address Memory byte address location of the word to be erased.

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

8.9.5. ERASE_APPLICATION_ZONE_WITH_ERASE

This command can be used in the following cases:

1. AT88SC101: To erase the data in Application Zone with EC Function Disabled.
2. AT88SC102: To erase the data in Application Zone 1.
3. AT88SC102: To erase the data in Application Zone 2 with EC2 Function Disabled.
4. AT88SC1003: To erase the data in Application Zone 1.
5. AT88SC1003: To erase the data in Application Zone 2 with EC2 Function Disabled.
6. AT88SC1003: To erase the data in Application Zone 3.

The following actions are executed for this command:

1. Present the specified code to the card
   a. Erase the presentation error counter. The data in corresponding Application Zone can be erased when the submitted code is correct.

Command Format

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	…	Byte N
FFh	20h	00h						

Where:

Error Counter LEN Length of presentation error counter in bits. The value should be 00h always.

Byte Address Byte address of the Application Zone Key in the card. Please refer to the table below for the correct value.

	Byte Address	LEN
AT88SC101: Erase Application Zone with EC function disabled	96h	04h
AT88SC102: Erase Application Zone 1	56h	06h
AT88SC102: Erase Application Zone 2 with EC2 function disabled	9Ch	04h
AT88SC1003: Erase Application Zone 1	36h	06h

	Byte Address	LEN
AT88SC1003: Erase Application Zone 2 with EC2 function disabled	5Ch	04h
AT88SC1003: Erase Application Zone 3	C0h	06h

MEM_L Length of the Erase Key. Please refer to the table above for the correct value.

CODE N bytes of Erase Key

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

Note: *After SW1 SW2 = 90 00h has been received, read back the data in Application Zone to check if the ERASE_APPLICATION_ZONE_WITH_ERASE is correct. If all data in Application Zone is erased and is equal to "FFh," the previous verification is successful.*

8.9.6. ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE

This command can be used in the following cases:

1. AT88SC101: To erase the data in Application Zone with EC Function Enabled.
2. AT88SC102: To erase the data in Application Zone 2 with EC2 Function Enabled.
3. AT88SC1003: To erase the data in Application Zone 2 with EC2 Function Enabled.

With EC or EC2 Function Enabled (that is, ECEN or EC2EN Fuse is undamaged and in "1" state), the following actions are executed:

1. Present the specified code to the card.
2. Search a '1' bit in the presentation error counter and write the bit to '0'.
3. Erase the presentation error counter. The data in corresponding Application Zone can be erased when the submitted code is correct.

Command Format

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	Byte 3	Byte 4
FFh	20h	80h		04h				

Where:

Error Counter LEN Length of presentation error counter in bits. The value should be 80h always.

Byte Address Byte address of the Application Zone Key in the card

	Byte Address
AT88SC101	96h
AT88SC102	9Ch
AT88SC1003	5Ch

CODE 4 bytes Erase Key

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

= 63 00h if there are no more retries

Note: *After SW1 SW2 = 90 00h has been received, read back the data in Application Zone can check whether the ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE is correct. If all data in Application Zone is erased and is equal to "FFh," the previous verification is successful.*

8.9.7. VERIFY_SECURITY_CODE

This command is used to submit Security Code (2 bytes) to the inserted card. Security Code is to enable the memory access of the card.

The following actions are executed:

1. Present the specified code to the card.
2. Search a '1' bit in the presentation error counter and write the bit to '0'.
3. Erase the presentation error counter. The Security Code Attempts Counter can be erased when the submitted code is correct.

Command Format

Pseudo-APDU						
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE	
					Byte 1	Byte 2
FFh	20h	08h	0Ah	02h		

Where:

Error Counter LEN Length of presentation error counter in bits

Byte Address Byte address of the key in the card

CODE 2 bytes Security Code

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

= 63 00h if there are no more retries

Note: *After SW1 SW2 = 90 00h has been received, read back the Security Code Attempts Counter (SCAC) to check whether the VERIFY_USER_CODE is correct. If SCAC is erased and is equal to "FFh," the previous verification is successful.*

8.9.8. BLOWN_FUSE

This command is used to blow the fuse of the inserted card. The fuse can be EC_EN Fuse, EC2EN Fuse, Issuer Fuse or Manufacturer's Fuse.

Note: *The blowing of fuse is an irreversible process.*

Command Format

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Fuse Bit Addr (High)	Fuse Bit Addr (Low)	State of FUS Pin	State of RST Pin
FFh	05h	00h	00h	04h			01h	00h or 01h

Where:

Fuse Bit Addr (2 bytes) Bit address of the fuse. Please refer to the table below for the correct value.

State of FUS Pin State of the FUS pin. Should always be 01h.

State of RST Pin State of the RST pin. Please refer to below table for the correct value.

		Fuse Bit Addr (High)	Fuse Bit Addr (Low)	State of RST Pin
AT88SC101	Manufacturer Fuse	05h	80h	01h
	EC_EN Fuse	05h	C9h	01h
	Issuer Fuse	05h	E0h	01h
AT88SC102	Manufacturer Fuse	05h	B0h	01h
	EC2EN Fuse	05h	F9h	01h
	Issuer Fuse	06h	10h	01h
AT88SC1003	Manufacturer Fuse	03h	F8h	00h
	EC2EN Fuse	03h	FCh	00h
	Issuer Fuse	03h	E0h	00h

Response Data Format

SW1	SW2

Where:

SW1 SW2 = 90 00h if no error

9.0. Contactless Card Commands

9.1. Pseudo APDU for Contactless Interface

9.1.1. Get Data

This command will return the serial number of the ATS of the PICC card.

Get UID APDU Format (5 bytes)

Command	Class	INS	P1	P2	Le
Get Data	FFh	CAh	00h 01h	00h	00h (Max length)

If P1=00h, Get UID Response Format (UID + 2 bytes)

Response	Data Out					
Result	UID (LSB)	…	…	UID (MSB)	SW1	SW2

If P1 = 01h, Get ATS of an ISO14443 A card (ATS + 2 bytes)

Response	Data Out		
Result	ATS	SW1	SW2

Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully
Warning	62 82h	The end of the UID/ATS reached before LE bytes (Le is greater than UID length)
Error	6C XX	Wrong length (wrong number Le: 'XX' encodes the exact number) if Le is less than the available UID length
Error	63 00h	The operation failed
Error	6A 81h	Function not supported

Examples:

//To get the serial number of the PICC card

UINT8 GET_UID[5]={FF, CA, 00, 00, 00h};

//To get the ATS of the ISO14443 A contactless card

UINT8 GET_ATS[5]={FF, CA, 01, 00, 00h};

9.2. PICC Commands (T=CL Emulation) for MIFARE 1K/4K Memory Cards

9.2.1. Load Authentication Keys

This command loads the authentication keys to the reader. The authentication keys are used to authenticate the particular sector of the MIFARE Classic (1K/4K) memory card. Two kinds of authentication key locations are provided: volatile and non-volatile key locations.

Load Authentication Keys APDU Format (11 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Load Authentication Keys	FFh	82h	Key Structure	Key Number	06h	Key (6 bytes)

Where:

Key Structure 1 byte.

00h = Key is loaded into the reader volatile memory.

Other = Reserved.

Key Number 1 byte.

00h – 01h = Non-volatile memory for storing keys. The keys are permanently stored in the reader and will be retained in the reader's memory even if the reader is disconnected from the PC. It can store up to 32 keys inside the reader non-volatile memory.

Note: *The default value is FF FF FF FF FF FFh.*

Key 6 bytes. The key value loaded into the reader.

Example: FF FF FF FF FF FFh.

Load Authentication Keys Response Format (2 Bytes)

Response	Data Out	
Result	SW1	SW2

Load Authentication Keys Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation is completed successfully.
Error	63 00h	The operation is failed.

Example:

// Load a key {FF FF FF FF FF FFh} into the volatile memory location 00h.

APDU = {FF 82 00 00 06 FF FF FF FF FF FFh}

9.2.2. Authentication for MIFARE 1K/4K

This command uses the keys stored in the ACR3x to do authentication with the MIFARE Classic (1K/4K) card (PICC). Two types of authentication keys are used: TYPE_A and TYPE_B.

Load Authentication Keys APDU Format (10 bytes)

Command	Class	INS	P1	P2	Lc	Data In
Authentication	FFh	86h	00h	00h	05h	Authenticate Data Bytes

Authenticate Data Bytes (5 bytes):

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Version 01h	00h	Block Number	Key Type	Key Number

Where:

Block Number 1 byte

The memory block to be authenticated.

For MIFARE 1K card, it has a total of 16 sectors, wherein each sector consists of 4 consecutive blocks.

e.g., Sector 00h consists of Blocks {00h, 01h, 02h and 03h}; Sector 01h consists of Blocks {04h, 05h, 06h and 07h}; the last sector 0Fh consists of Blocks {3Ch, 3Dh, 3Eh and 3Fh}.

Once the authentication is done successfully, there is no need to do the authentication again provided that the blocks to be accessed belong to the same sector.*

Please refer to the MIFARE 1K/4K specification for more details.

***Note:** Once the block is authenticated successfully, all blocks belonging to the same sector are accessible.*

Key Type 1 byte.

60h = Key is used as TYPE A key for authentication

61h = Key is used as TYPE B key for authentication

Key Number 1 byte.

00h ~ 01h = Volatile memory for storing keys. They keys will be removed when the reader is disconnected from the computer. Two volatile keys are provided. The volatile key can be used as a session key for different sessions.

Load Authentication Keys Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Load Authentication Keys Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully

Results	SW1 SW2	Meaning
Error	63 00h	The operation failed

Sectors (16 sectors, 4 consecutive blocks per sector)	Data Blocks (3 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)
Sector 0	00 ~ 02h	03h
Sector 1	04 ~ 06h	07h
…		
…		
Sector 14	38 ~ 0Ah	3Bh
Sector 15	3C ~ 3E	3Fh

1K bytes

Table 3: MIFARE 1K Memory Map

Sectors (32 sectors, 4 consecutive blocks per sector)	Data Blocks (3 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)
Sector 0	00 ~ 02h	03h
Sector 1	04 ~ 06h	07h
…		
…		
Sector 30	78 ~ 7Ah	7Bh
Sector 31	7C ~ 7Eh	7Fh

2K bytes

Sectors (8 sectors, 16 consecutive blocks per sector)	Data Blocks (15 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)
Sector 32	80 ~ 8Eh	8Fh
Sector 33	90 ~ 9Eh	9Fh
…		
…		
Sector 38	E0 ~ EEh	EFh
Sector 39	F0 ~ FEh	FFh

2K bytes

Table 4: MIFARE 4K Memory Map

Example:

//To authenticate the Block 04h with a {TYPE A, key number 00h}

// PC/SC V2.01, Obsolete

APDU = {FF 88 00 04 60 00h};

Similarly,

// To authenticate the Block 04h with a {TYPE A, key number 00h}.

// PC/SC V2.07

APDU = {FF 86 00 00 05 01 00 04 60 00h}

Note: *MIFARE Ultralight does not need to perform authentication as its memory is freely accessible.*

Byte Number	0	1	2	3	Page
Serial Number	SN0	SN1	SN2	BCC0	0
Serial Number	SN3	SN4	SN5	SN6	1
Internal/Lock	BCC1	Internal	Lock0	Lock1	2
OTP	OPT0	OPT1	OTP2	OTP3	3
Data read/write	Data0	Data1	Data2	Data3	4
Data read/write	Data4	Data5	Data6	Data7	5
Data read/write	Data8	Data9	Data10	Data11	6
Data read/write	Data12	Data13	Data14	Data15	7
Data read/write	Data16	Data17	Data18	Data19	8
Data read/write	Data20	Data21	Data22	Data23	9
Data read/write	Data24	Data25	Data26	Data27	10
Data read/write	Data28	Data29	Data30	Data31	11
Data read/write	Data32	Data33	Data34	Data35	12
Data read/write	Data36	Data37	Data38	Data39	13
Data read/write	Data40	Data41	Data42	Data43	14
Data read/write	Data44	Data45	Data46	Data47	15

512 bits or 64 bytes

Table 5: MIFARE Ultralight Memory Map

9.2.3. Read Binary Blocks

The Read Binary Blocks command is used in retrieving multiple data blocks from the PICC card. The data block/trailer block must be authenticated first before executing the Read Binary Blocks command.

Read Binary APDU Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Read Binary Blocks	FFh	B0h	00h	Block Number	Number of Bytes to Read

Where:

Block Number 1 byte. The starting block.

Number of Bytes to Read 1 byte.

Multiples of16 bytes for MIFARE 1K/4K or Multiples of 4 bytes for MIFARE Ultralight.

Maximum of 48 bytes for MIFARE 1K (Multiple blocks mode; 3 consecutive blocks).

Maximum of 240 bytes for MIFARE 4K (Multiple blocks mode; 15 consecutive blocks).

Maximum of 16 bytes for MIFARE Ultralight.

Example 1: 10h (16 bytes). The starting block only (Single Block Mode)

Example 2: 40h (64 bytes). From the starting block to starting block+3. (Multiple Blocks Mode)

Note: *For security reasons, the Multiple Block Mode is used for accessing Data Blocks only. The Trailer Block is not supposed to be accessed in Multiple Blocks Mode. Please use Single Block Mode to access the Trailer Block.*

Read Binary Block Response Format (Multiples of 4/16 + 2 Bytes)

Response	Data Out		
Result	Data (Multiples of 4/16 Bytes)	SW1	SW2

Read Binary Block Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully.
Error	63 00h	The operation failed.

Examples:

//Read 16 bytes from the binary block 04h (MIFARE 1K or 4K)

APDU = {FF B0 00 04 10}

//Read 240 bytes starting from the binary block 80h (MIFARE 4K)

//Block 80 to Block 8Eh (15 blocks)

APDU = {FF B0 00 80 F0}

9.2.4. Update Binary Blocks

The Update Binary Blocks command is used in writing a multiple of data blocks into the PICC card. The data block/trailer block must be authenticated first before executing the Update Binary Blocks command.

Update Binary APDU Format (Multiples of 16 + 5 bytes)

Command	Class	INS	P1	P2	Le	Data In
Update Binary Blocks	FFh	D6h	00h	Block Number	Number of Bytes to Update	Block Data (Multiples of 16 bytes)

Where:

Block Number — 1 byte. The starting block to be updated.

Number of Bytes to Update — 1 byte. Multiples of16 bytes for MIFARE 1K/4K or Multiples of 4 bytes for MIFARE Ultralight.

Maximum of 48 bytes for MIFARE 1K (Multiple blocks mode; 3 consecutive blocks).

Maximum of 240 bytes for MIFARE 4K (Multiple blocks mode; 15 consecutive blocks).

Maximum of 16 bytes for MIFARE Ultralight.

Example 1: 10h (16 bytes). The starting block only. (Single Block Mode)

Example 2: 30h (48 bytes). From the starting block to starting block +2. (Multiple Blocks Mode)

Note: *For safety reasons, the Multiple Block Mode is used for accessing data blocks only. The Trailer Block is not supposed to be accessed in Multiple Blocks Mode. Please use Single Block Mode to access the Trailer Block.*

Block Data — Multiples of 16 + 2 bytes (or 6 bytes)

The data to be written into the binary block/s.

Update Binary Block Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully.
Error	63 00h	The operation failed.

Examples:

//Update the binary block 04h of MIFARE 1K or 4K with data {00 01 … 0F}

APDU = {FF D6 00 04 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F}

//Update the binary block 04h of MIFARE Ultralight with data {00 01 02 03}

APDU = {FF D6 00 04 04 00 01 02 03}

9.2.5. Value Block Operation (INC, DEC, STORE)

The Value Block Operation command is used in manipulating value-based transactions, such as incrementing a value of the value block.

Value Block Operation APDU Format (10 bytes)

Command	Class	INS	P1	P2	Le	Data In	
Value Block Operation	FFh	D7h	00h	Block Number	05h	VB_OP	VB_Value (4 bytes) {MSB .. LSB}

Where:

Block Number 1 byte. The value block to be manipulated.

VB_OP 1 byte.

00h = Store the VB_Value into the block. The block will then be converted to a value block.

01h = Increment the value of the value block by the VB_Value. This command is only valid for value block.

02h = Decrement the value of the value block by the VB_Value. This command is only valid for value block.

VB_Value 4 bytes**.** The value used for value manipulation. The value is a signed long integer.

Example 1: Decimal –4 = {FFh, FFh, FFh, FCh}

VB_Value			
MSB			LSB
FFh	FFh	FFh	FCh

Example 2: Decimal 1 = {00h, 00h, 00h, 01h}

VB_Value			
MSB			LSB
00h	00h	00h	01h

Value Block Operation Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Value Block Operation Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully.
Error	63 00h	The operation failed.

9.2.6. Read Value Block

The Read Value Block command is used in retrieving the value from the value block. This command is only valid for value block.

Read Value Block APDU Format (5 bytes)

Command	Class	INS	P1	P2	Le
Read Value Block	FFh	B1h	00h	Block Number	04h

Where:

Block Number 1 byte. The value block to be accessed.

Read Value Block Response Format (4 + 2 bytes)

Response	Data Out		
Result	Value {MSB … LSB}	SW1	SW2

Where:

Value 4 bytes. The value returned by the card.

The value is a signed long integer.

Example 1: Decimal –4 = {FFh, FFh, FFh, FCh}

Value			
MSB			LSB
FFh	FFh	FFh	FCh

Example 2: Decimal 1 = {00h, 00h, 00h, 01h}

Value			
MSB			LSB
00h	00h	00h	01h

Read Value Block Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully.
Error	63 00h	The operation failed.

9.2.7. Copy Value Block

The Copy Value Block command is used to copy a value from a value block to another value block.

Copy Value Block APDU Format (7 bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Value Block Operation	FFh	D7h	00h	Source Block Number	02h	03h	Target Block Number

Where:

Source Block Number 1 byte.

The value of the source value block will be copied to the target value block.

Target Block Number 1 byte.

The value block to be restored.

The source and target value blocks must be in the same sector.

Copy Value Block Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Copy Value Block Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully.
Error	63 00h	The operation failed.

Examples:

//Store a value 1 into block 05h

APDU = {FF D7 00 05 05 00 00 00 00 01}

//Read a value into block 05h

APDU = {FF B1 00 05 04}

//Copy the value from value block 05h to value block 06h

APDU = {FF D7 00 05 04 02 03 06}

//Increment the value block 05h by 5

APDU = {FF D7 00 05 05 01 00 00 00 05}

9.2.8. Access PC/SC Compliant Tags (ISO14443-4)

Basically, all ISO14443-4 compliant cards (PICC) would understand the ISO7816-4 APDUs. The ACR35 will communicate with the ISO14443-4 compliant cards by exchanging ISO7816-4 APDUs and responses. ACR35 will handle the ISO14443 Parts 1-4 protocols internally.

MIFARE 1K, 4K, Mini, and Ultralight tags are supported through the T=CL emulation.

ISO7816-4 APDU Format

Command	Class	INS	P1	P2	Lc	Data In	Le
ISO7816 Part 4 Command					Length of the Data In		Expected length of the Response Data

ISO7816-4 Response Format (Data + 2 bytes)

Response	Data Out		
Result	Response Data	SW1	SW2

ISO7816-4 Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation was completed successfully.
Error	63 00h	The operation failed.

Typical sequence may be:

1. Present the tag and connect the PICC Interface.
2. Read/Update the memory of the tag.

To do this:

1. Connect the tag.

   The ATR of the tag is 3B 88 80 01 00 00 00 00 33 81 81 00 3Ah.

   In which,

   The Application Data of ATQB = 00 00 00 00, protocol information of ATQB = 33 81 81. It is an ISO 14443-4 Type B tag.

2. Send an APDU, Get Challenge.

   << 00 84 00 00 08h

   >> 1A F7 F3 1B CD 2B A9 58h [90 00h]

   Note: *For ISO 14443-4 Type A tags, the ATS can be obtained by using the APDU "FF CA 01 00 00h."*

Example:

//To read 8 bytes from an ISO14443-4 Type B PICC

APDU = {80 B2 80 00 08}

Class	INS	P1	P2	Lc	Data In	Le
80h	B2h	80h	00h	None	None	08h

Answer: 00 01 02 03 04 05 06 07 [$9000]

9.2.9. Access FeliCa Tags

For FeliCa Access, the command is different with PC/SC-compliant tags and MIFARE. The command follows FeliCa specification with header added.

FeliCa Command Format

Command	Class	INS	P1	P2	Lc	Data In
Felica Command	FFh	00h	00h	00h	Length of the Data In	Felica Command (start with Length byte)

FeliCa Response Format (Data + 2 bytes)

Response	Data Out
Result	Response Data

Read Memory Block Example:

1. Connect the FeliCa.

   The ATR = 3B 8F 80 01 80 4F 0C A0 00 00 03 06 11 00 3B 00 00 00 00 42h

   In which, 11 00 3Bh = FeliCa

2. Read FeliCa IDM.

   CMD = FF CA 00 00 00h

   RES = [IDM (8bytes)] 90 00h

   e.g., FeliCa IDM = 01 01 06 01 CB 09 57 03h

3. FeliCa command access.

   Example: “Read” Memory Block.

   CMD = FF 00 00 00 10 10 06 01 01 06 01 CB 09 57 03 01 09 01 01 80 00h

   where:

   Felica Command = 10 06 01 01 06 01 CB 09 57 03 01 09 01 01 80 00h

   IDM = 01 01 06 01 CB 09 57 03h

   RES = Memory Block Data

10.0.Sensitive Data Injection Method

In this section, a method will be discussed on injecting sensitive data, such as the Customer Master Key, DUKPT Initial PIN Encryption Key, AES Encryption Key, and Custom ID into the ACR3x in a more secured scenario.

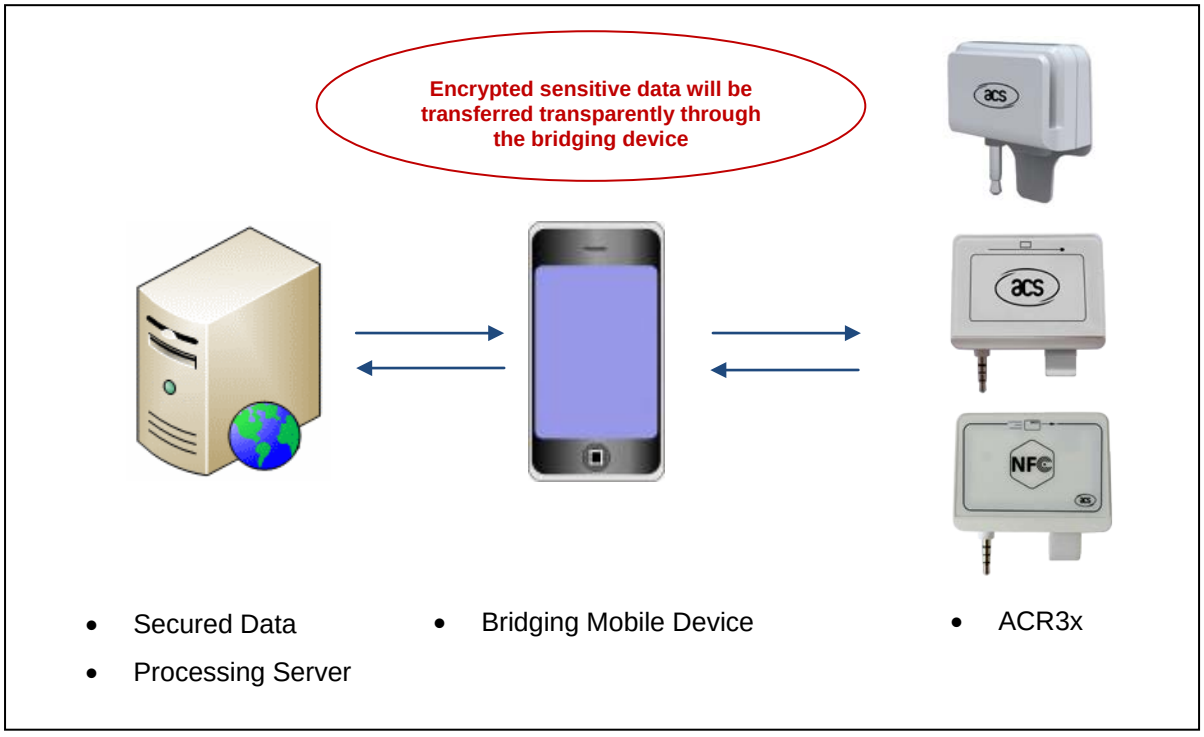

Figure 4: Sensitive Data Injection Model

In the figure shown above, there are three entities involved, namely the Secured Data Processing Server, the Bridging Mobile Device and ACR3x. The Secured Data Processing Server is responsible for receiving and generating ciphered sensitive data targeted at ACR3x, while the mobile device only act as a message bridging channel between the data processing server and ACR3x. No data between the server and ACR3x will undergo processing (except the need to repack the data into the frames suitable to be sent through the audio channel to ACR3x) in the mobile device.

10.1. Authentication

Before any sensitive data can be loaded into ACR3x, the data processing server (at the same time the mobile device is connected to the server) must be authenticated by ACR3x for the privilege to modify the secured data inside ACR3x. In ACR3x, a mutual authentication method is being used.

An authentication request is always initiated by either the data processing server or the bridging device, which will then trigger ACR3x to return a sequence of 16 bytes of random numbers (RND_A[0:15]). The random numbers are encrypted with the Customer Master Key currently stored in ACR3x using the AES-128 CBC ciphering mode before being sent out from ACR3x. The bridging device must pass this sequence of encrypted random numbers to the data processing server, which will then undergo AES-128 CBC cipher mode encryption using the Customer Master Key that is being used in the data processing server (which should be the same as the one that is being used in ACR3x and should be kept securely by the customer). The 16 bytes of decrypted random numbers from ACR3x is then padded to the end of another 16 bytes of random numbers generated by the data processing server (RND_B[0:15]). The final sequence of 32 bytes of random numbers (RND_C[0:31]), that is:

RND_C[0:31] = RND_B[0:15] + RND_A[0:15],

will undergo decryption operation with the Customer Master Key being used in the server and the final

output data is sent to ACR3x through the bridging device using an authentication response message.

When ACR31 receives the authentication response message, the message data will undergo an encryption operation using its own Customer Master Key and will be converted back to the normal 32 bytes of random numbers. In theory, the first 16 bytes of random numbers should be equal to RND_B[0:15] and are generated by the data processing server while the other 16 bytes should be equal to RND_A[0:15] and are originally generated by ACR3x.

ACR3x will first compare if RND_A[0:15] is the same as the original version. If it is the same, then the data processing server is authenticated by ACR3x. ACR3x will then encrypt RND_B[0:15] obtained using the Customer Master Key and the feedback to the data processing server through the bridging device using the answer to the authentication response message.

Upon receiving the answer to the authentication response message, the data processing server will decrypt the data contained in the message and check if the 16 bytes of random numbers are all equal to those originally generated RND_B[0:15]. If they are the same, then ACR3x is authenticated by the server. At this moment, the whole authentication process is completed and sensitive data can be injected into ACR3x.

After successful authentication, a 16-byte Session Key is generated in both ACR3x and the data processing server. The Session Key (SK[0:15]) is obtained by padding the first 8 bytes of RND_A at the end of the first 8 bytes of RND_B, that is:

SK[0:15] = RND_B[0:7] + RND_A[0:7]

All sensitive data leaving out of the Secured Data Processing Server must be encrypted with this Session Key using the AES-128 CBC ciphering mode. Thus, even if the encrypted data may be captured in the bridging mobile device, it is still very difficult to retrieve the original sensitive data without any prior knowledge of the Customer Master Key.

For better pictorial illustration, please refer to figure below (The picture below has omitted the bridging device for simplicity and better illustration):

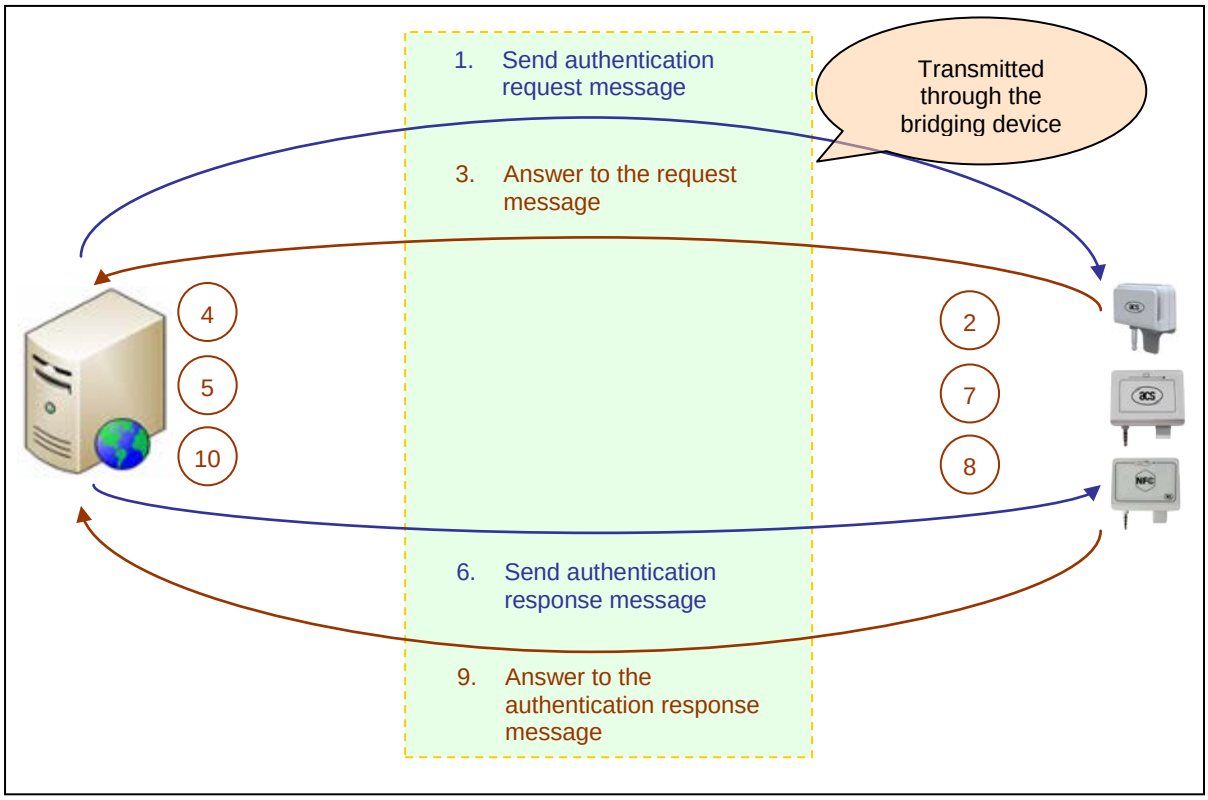

Figure 5: Authentication Procedure

Below is a summary of the above mentioned steps:

1. The data processing server/bridging device initiates an authentication request from ACR3x by issuing an authentication request message.
2. Upon receiving the authentication request message, ACR3x will generate 16 bytes of random numbers (RND_A[0:15]). The whole 16 bytes of data is encrypted with the Customer Master Key currently being used by ACR3x.
3. The encrypted version of RND_A[0:15] is then transferred to the data processing server through the answer to the authentication response message.
4. The data processing server will decrypt the data received to recover RND_A[0:15].
5. The data processing server will generate another 16 bytes of random numbers (RND_B[0:15]). RND_A[0:15] will be padded to the end of RND_B[0:15] to form a sequence of 32-byte random numbers (RND_C[0:31] = RND_B[0:15] + RND_A[0:15]). All the 32 bytes of random numbers will undergo a decryption process with the Customer Master Key currently being used in the server.
6. The final output data from the decryption process will be transferred to ACR3x through the authentication response message.
7. In ACR3x, an encryption process will be performed on the received data to recover the 32 bytes of random number. ACR3x will check the result RND_A[0:15] to see if they are the same as the original ones. If not, the authentication process will be terminated.
8. ACR3x will encrypt the resultant RND_B[0:15] with the Customer Master Key. At the same time, a 16-byte Session Key is created by padding the first 8 bytes of RND_A to the end of the first 8 bytes of RND_B.
9. The encrypted RND_B[0:15] will be transferred to the data processing server through the authentication response message.
10. The data processing server will decrypt the message data and compare if the content is equal to the original RND_B[0:15]. If not, the authentication process will be terminated. Otherwise, the authentication process is completed and a 16-byte Session Key is created by padding the first 8 bytes of RND_A to the end of the first 8 bytes of RND_B.

10.2. Customer Master Key Injection

At the time ACR3x is manufactured in the factory, its flash memory should have been reset to some default values. For the Customer Master Key, it should have been reset to all 0s.

In order to change the Customer Master Key, the data processing server and ACR3x must be authenticated with the old one first. After successful authentication, the data processing server can send the set master key command message with the new Customer Master Key to ACR3x. The new Customer Master Key must be encrypted with the current Session Key. After the new Customer Master Key has been successfully loaded to ACR3x, the already established authentication by ACR3x will be dropped. The server should perform a new authentication request with the new Customer Master Key before any further injection of sensitive data can proceed.

10.3. AES Key Injection

At the time ACR3x is manufactured in the factory, there is a default ACS AES Key being loaded into the flash memory. Customer can change this key to any values after authentication.

The AES Key is used to encrypt the magnetic stripe track data if the DUKPT is disabled. The new AES Key is immediately effective and it does not affect the current authenticated session.

10.4. DUKPT Initialization

Before the DUKPT Key Management algorithm can work properly, some initialization processes have to be performed.

First, the data processing server must provide the 10-byte Initial Key Serial Number (IKSN) and the 16-byte Initial PIN Encryption Key (IPEK) to ACR31. These two sequences of numbers will be used by the DUKPT Key Management Engine to initialize its future key tables and other settings. The encryption counter of the DUKPT Engine will automatically be reset.

After the DUKPT is initialized, the DUKPT Option should be enabled so that the key used for encrypting the magnetic stripe track data will be generated by the DUKPT algorithm. After every successful card swipe, a unique encryption key is requested from the DUKPT instead of using fixed AES Key for every transaction.

It should be noted that if there is an error in the swiped card data, no key will be requested from the DUKPT Engine. Instead, the track data will all be set to zero in the response message only the error code will be setup to indicate the type(s) of errors in the card data detected. The approach of not requesting key from the DUKPT for unsuccessful card swipe is to allow the mobile device application to prompt user to swipe again without pushing the useless data to the backend server, while maintaining a more synchronized encryption counter with the server.

11.0. Card Data Encryption

Every time a card is swiped, a response message will be automatically sent to the mobile device. The track data encapsulated in the message will be encrypted using AES-128 CBC cipher mode (The initialization vector will be equal to 16 bytes of zeros).

If DUKPT is enabled, the key used for the track data encryption will be generated by the DUKPT Key Management algorithm for every successful swipe. As a result, different key will be used for the track data encryption for every successful transaction.

In case the DUKPT is disabled, the AES Key will be used for the track data encryption. You can modify the AES Key by using the Sensitive Data Injection Method in **Section 10.0**. When ACR3x is shipped out from factory, a default AES key is pre-loaded inside ACR3x. The default AES Key is equal to:

4E 61 74 68 61 6E 2E 4C 69 20 54 65 64 79 20h

It should be noted that when there is data error during card swipe, the track data field will be filled with zeros and only the error will be reported in the message.

12.0. AES-128 CBC Encryption Test Vectors

The table below illustrates several test vectors for the AES-128 CBC cipher mode encryption being used in ACR3x.

Original Data:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
Initial Vector:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
Key:	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
Encryption Output:	6B 1E 2F FF E8 A1 14 00 9D 8F E2 2F 6D B5 F8 76h

Original Data:	69 88 44 21 13 84 0A 10 00 0C 02 22 11 88 00 0E 12 84 00 B1 40 80 80 11 31 02 45 20 20 28 E4 00h
Initial Vector:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
Key:	10 29 02 14 03 90 53 09 12 08 20 20 02 C0 9A 80h
Encryption Output:	EF 14 C9 C9 F3 48 96 5B 18 36 0A 2F 81 1A 93 C7 E2 FF F3 61 04 B8 D4 5E 13 F7 26 FE 2A 94 2B 69h

Original Data:	80 83 11 13 09 D1 11 30 0E 00 0A 49 04 00 26 99 C0 58 D1 7A 45 CD 17 10 30 00 22 08 10 4C 41 51h
Initial Vector:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
Key:	9A 04 2A 10 21 00 06 20 10 84 20 01 00 00 22 1Ch
Encryption Output:	56 85 B9 6B A1 B2 09 AB 58 71 58 B5 E0 30 42 71 64 62 51 FA 55 94 52 BC 78 33 24 FB 15 F5 33 62h

Original Data:	41 01 06 02 21 A8 C4 40 08 00 44 11 11 88 0D 09 10 81 92 10 01 20 20 2E 20 C4 05 81 58 08 18 86h
Initial Vector:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
Key:	30 13 30 C8 91 53 49 44 0E 29 98 42 84 17 00 D0h
Encryption Output:	ED 0F 2E BC 7D EA 58 C4 AB E8 72 91 87 74 2F C3 B1 8B 66 4F F5 E5 3F 8B BD A9 63 40 F8 0D 11 97h

13.0. TDES ECB Encryption Test Vectors

The table below illustrates several test vectors for the triple DES ECB cipher mode encryption being used in ACR3x.

Original Data:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
Key:	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
Encryption Output:	49 F9 E7 A6 0C 40 6D BF 49 F9 E7 A6 0C 40 6D BFh
Original Data:	02 50 88 82 22 21 13 C4 42 00 08 44 60 24 8A 04h
Key:	00 C0 08 28 8E 28 16 10 01 80 50 4D 72 00 28 88h
Encryption Output:	8E BF 16 AA B4 59 AA C0 13 DB 32 E5 1D 04 BD 66h
Original Data:	61 10 88 19 42 31 01 26 42 02 74 24 00 07 0C 82h
Key:	00 10 80 42 09 20 13 24 82 22 24 89 62 08 09 90h
Encryption Output:	74 57 DF 51 3B 04 7A F2 2B 26 C4 BF 81 6B 4D 58h

Appendix A. Track Data Error Code

Bit 7 MSB	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 LSB
MSB	0	0	0	0	LRC error	End Sentinel error	Start Sentinel error

Notes:

1. *Bits 7 to 1 are error codes.*
2. *Error-free = 0*

Appendix B. System Error Codes

The following table lists all the system error codes and their corresponding description for ACR3x.

Error Code	Status
00h	ERROR_SUCCESS
FFh	ERROR_INVALID_CMD
FEh	ERROR_INVALID_PARAM
FDh	ERROR_INVALID_CHECKSUM
FCh	ERROR_INVALID_STARTBYTE
FBh	ERROR_UNKNOWN
FAh	ECODE_DUKPT_CEASE_OPERATION
F9h	ECODE_DUKPT_DATA_CORRUPTED
F8h	ECODE_FLASH_DATA_CORRTPTED
F7h	ECODE_VERIFICATION_FAILED

Table 6: System Error Codes