Advanced Card Systems Ltd.
Card & Reader Technologies

# ACR1281U-C1

## USB Dual Interface Reader

Application Programming Interface V1.07

*Subject to change without prior notice*

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

# Table of Contents

## List of Figures

## List of Tables

# 1.0. Introduction

ACR1281U-C1 DualBoost II is the second generation of ACS's ACR128 DualBoost Reader. ACR1281U-C1 is a powerful and efficient dual interface smart card reader, which can be used to access ISO 7816 MCU cards, MIFARE® cards and ISO 14443 Type A and B contactless cards. It makes use of the USB CCID class driver and USB interface to connect to a PC and accept card commands from the computer application.

ACR1281U-C1 acts as the intermediary device between the computer and the card. The reader, which communicates with a contactless tag, MCU card, SAM card, or the device peripherals (LED or buzzer), will carry out a command issued from the computer. It has three interfaces namely the PICC, ICC and SAM interfaces, which all follow the PC/SC specifications. The contact interface makes use of the APDU commands as defined in ISO 7816 specifications. For contact MCU card operations, please refer to the related card documentation and the PC/SC specifications.

This API document details how the PC/SC APDU commands are implemented for the contactless interface, contact memory card support and device peripherals of ACR1281U-C1.

## 2.0. Features

The ACR1281U-C1 USB Dual Interface Reader has the following features:

- USB 2.0 Full-speed Interface
- CCID-compliant
- Smart Card Reader:
    - Contactless Interface:
        - Read/Write speed of up to 848 Kbps
        - Built-in antenna for contactless tag access, with card reading distance of up to 50 mm (depending on tag type)
        - Supports ISO 14443 Part 4 Type A and B cards and MIFARE series
        - Built-in anti-collision feature (only one tag is accessed at any time)
        - Supports extended APDU (max. 64 KB)
    - Contact Interface:
        - Supports ISO 7816 Class A, B and C (5 V, 3 V and 1.8 V)
        - Supports microprocessor cards with T=0 or T=1 protocol
        - Supports memory cards
    - SAM Interface:
        - One SAM Slot
        - Supports ISO 7816 Class A SAM cards
- Application Programming Interface:
    - Supports PC/SC
    - Supports CT-API (through wrapper on top of PC/SC)
- Built-in Peripherals:
    - Two user-controllable LEDs
    - User-controllable buzzer
- USB Firmware Upgradability
- Supports Android™ 3.1 and later[1]
- Compliant with the following standards:
    - ISO 14443
    - ISO 7816
    - PC/SC
    - CCID
    - CE
    - FCC
    - RoHS 2
    - Microsoft® WHQL

[1] *Uses an ACS-defined Android Library*

# 3.0. ACR1281U-C1 Architecture

## 3.1. Reader Block Diagram

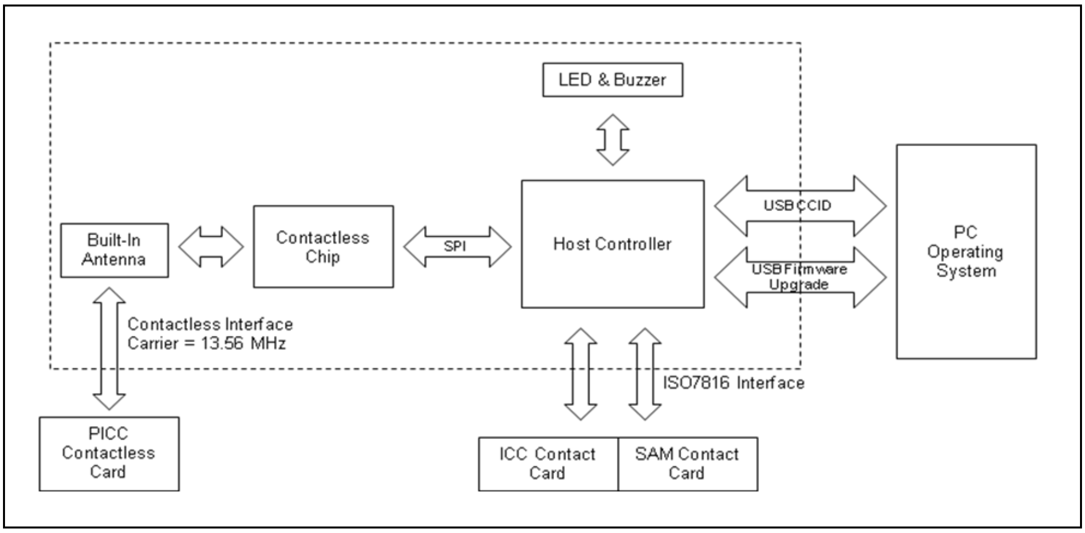

**Figure 1**: ACR1281U-C1 Reader Block Diagram

## 3.2. Communication between PC/SC driver and ICC, PICC and SAM

The protocol being used between ACR1281U-C1 and the PC is CCID. All communications between ICC, PICC and SAM are PC/SC-compliant.

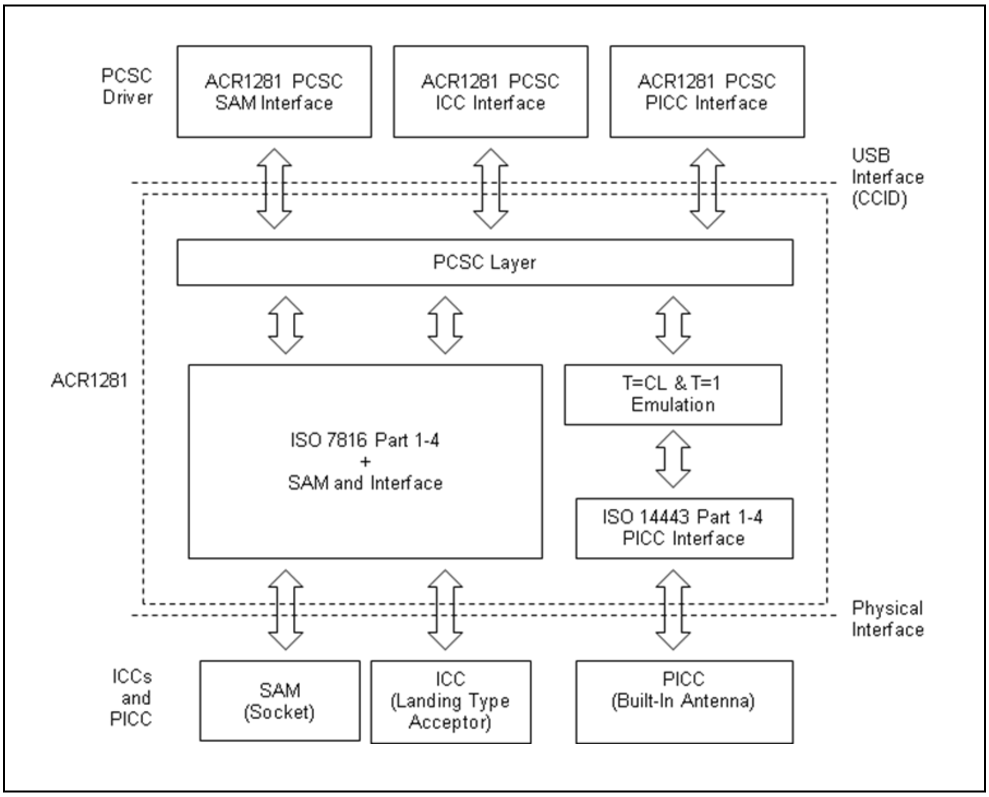
PCSC Driver
ACR1281 PCSC SAM Interface
ACR1281 PCSC ICC Interface
ACR1281 PCSC PICC Interface
USB Interface (CCID)
PCSC Layer
ACR1281
ISO 7816 Part 1-4 + SAM and Interface
T=CL & T=1 Emulation
ISO 14443 Part 1-4 PICC Interface
Physical Interface
ICCs and PICC
SAM (Socket)
ICC (Landing Type Acceptor)
PICC (Built-In Antenna)

**Figure 2**: ACR1281U-C1 Architecture

# 4.0. Hardware Design

## 4.1. USB

The ACR1281U-C1 connects to a computer through USB following the USB standard.

### 4.1.1. Communication Parameters

The ACR1281U-C1 connects to a computer through USB as specified in the USB Specification 2.0. The ACR1281U-C1 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 ACR1281U-C1 and PC |
| 3 | D+ | Differential signal transmits data between ACR1281U-C1 and PC |
| 4 | GND | Reference voltage level for power supply |

**Table 1**: USB Interface Wiring

***Note:*** *For ACR1281U-C1 to function properly through USB interface, the device driver should be installed.*

### 4.1.2. Endpoints

The ACR1281U-C1 uses the following endpoints to communicate with the host computer:

**Control Endpoint** – For setup and control purposes.

**Bulk-OUT** – For commands to be sent from host to ACR1281U-C1 (data packet size is 64 bytes).

**Bulk-IN** – For response to be sent from ACR1281U-C1 to host (data packet size is 64 bytes).

**Interrupt-IN** – For card status message to be sent from ACR1281U-C1 to host (data packet size is 8 bytes).

## 4.2. Contact Smart Card Interface

The interface between the ACR1281U-C1 and the inserted smart card follows the specifications of ISO 7816-3 with certain restrictions or enhancements to increase the practical functionality of the ACR1281U-C1.

### 4.2.1. Smart Card Power Supply VCC (C1)

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

### 4.2.2. Card Type Selection

Before activating the inserted card, the controlling PC always needs to select the card type through the proper command sent to the ACR1281U-C1. This includes both memory card 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 a 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.

### 4.2.3. 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).

## 4.3. Contactless Smart Card Interface

The interface between the ACR1281U-C1 and the contactless card follows the specifications of ISO 14443 with certain restrictions or enhancements to increase the practical functionality of the ACR1281U-C1.

### 4.3.1. Carrier Frequency

The carrier frequency for ACR1281U-C1 is 13.56 MHz.

### 4.3.2. Card Polling

The ACR1281U-C1 automatically polls the contactless cards that are within the field. ISO 14443-4 Type A, ISO 14443-4 Type B and MIFARE cards are supported.

## 4.4. User Interface

### 4.4.1. Buzzer

A monotone buzzer is used to show the “Card Insertion” and “Card Removal” events.

| Events | Buzzer |
|---|---|
| 1. The reader is powered up and successfully initialized. | Beep |
| 2. Card Insertion Event (ICC or PICC) | Beep |
| 3. Card Removal Event (ICC or PICC) | Beep |

**Table 2**: Buzzer Event

### 4.4.2. LED

The LEDs are used for showing the state of the contact and contactless interfaces. The Red LED is used for showing PICC status and Green LED for ICC.

| Reader States | Red LED<br>PICC Indicator | Green LED<br>ICC Indicator |
|---|---|---|
| 1. No PICC found or PICC is available but not activated. | A single pulse per ~ 5 seconds | |
| 2. PICC is available and activated. | ON | |
| 3. PICC is operating. | Blinking | |
| 4. ICC is available and activated. | | ON |
| 5. ICC is unavailable or inactive. | | OFF |
| 6. ICC is operating. | | Blinking |

**Table 3**: LED Indicator

# 5.0. Software Design

## 5.1. Contact Smart Card Protocol

### 5.1.1. Memory Card – 1/2/4/8/16 kilobits I2C Card

#### 5.1.1.1. Select card type

This command powers down/up the selected card in the reader, and then performs a card reset after.

Command

| Command | Class | INS | P1 | P2 | Lc | Card Type |
|---|---|---|---|---|---|---|
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 01h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.1.2. Select page size

This command chooses the page size to read in the card. The default value is an 8-byte page write. It resets to the default value whenever the card is removed or the reader is turned off.

Command

| Command | Class | INS | P1 | P2 | Lc | Page Size |
|---|---|---|---|---|---|---|
| Select Page Size | FFh | 01h | 00h | 00h | 01h | |

Where:

**Page Size** (1 byte)

03h = 8-byte page write

04h = 16-byte page write

05h = 32-byte page write

06h = 64-byte page write

07h = 128-byte page write

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.1.3. Read memory card

This command reads the memory card's content from a specified address.

Command

| Command | Class | INS | Byte Address | | MEM_L |
|---|---|---|---|---|---|
| | | | MSB | LSB | |
| Read Memory Card | FFh | B0h | | | |

Where:

**Byte Address** (2 bytes)
Memory address location of the memory card

**MEM_L** (1 byte)
Length of data to be read from the memory card

Response

| Response | Byte 1 | … | … | Byte N | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.1.4. Write memory card

This command writes the memory card's content to a specified address.

Command

| Command | Class | INS | Byte Address | | MEM_L | Byte 1 | … | … | Byte N |
|---|---|---|---|---|---|---|---|---|---|
| | | | MSB | LSB | | | | | |
| Write Memory Card | FFh | D0h | | | | | | | |

Where:

**Byte Address** (2 bytes)
Memory address location of the memory card

**MEM_L** (1 byte)
Length of data to be read from the memory card

**Byte (1…N)** Data to be written to the memory card.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.2. Memory Card – 32/64/128/256/512/1024 kbits I2C Card

#### 5.1.2.1. Select card type

This command powers down/up the selected card in the reader, and then performs a card reset after.

Command

| Command | Class | INS | P1 | P2 | Lc | Card Type |
|---|---|---|---|---|---|---|
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 02h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.2.2. Select page size

This command chooses the page size to read in the card. The default value is an 8-byte page write. It resets to the default value whenever the card is removed or the reader is turned off.

Command

| Command | Class | INS | P1 | P2 | Lc | Page Size |
|---|---|---|---|---|---|---|
| Select Page Size | FFh | 01h | 00h | 00h | 01h | |

Where:

**Page Size** (1 byte)

03h = 8-byte page write

04h = 16-byte page write

05h = 32-byte page write

06h = 64-byte page write

07h = 128-byte page write

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.2.3. Read memory card

This command reads the memory card's content from a specified address.

Command

| Command | Class | INS | Byte Address | | MEM_L |
|---|---|---|---|---|---|
| | | | MSB | LSB | |
| Read Memory Card | FFh | | | | |

Where:

**INS** (1 byte)

B0h = For 32, 64, 128, 256, 512 kbit I2C card

1011 000*b; where * is the MSB of the 17 bit addressing = For 1024 kbit I2C card

**Byte Address** (2 bytes)

Memory address location of the memory card

**MEM_L** (1 byte)

Length of data to be read from the memory card

Response

| Response | Byte 1 | ... | ... | Byte N | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.2.4. Write memory card

This command writes the memory card's content to a specified address.

Command

| Command | Class | INS | Byte Address | | MEM_L | Byte 1 | ... | ... | Byte N |
|---|---|---|---|---|---|---|---|---|---|
| | | | MSB | LSB | | | | | |
| Write Memory Card | FFh | | | | | | | | |

Where:

**INS** (1 byte)

D0h = For 32, 64, 128, 256, 512 kbit I2C card

1101 000*b; where * is the MSB of the 17 bit addressing = For 1024 kilobit I2C card

**Byte Address** (2 bytes)
Memory address location of the memory card

**MEM_L** (1 Byte)
Length of data to be read from the memory card

**Byte (1…N)** Data to be written to the memory card.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.3. Memory Card – ATMEL AT88SC153

#### 5.1.3.1. Select card type

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

Command

| Pseudo-APDU | | | | | | |
|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **Card Type** |
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 03h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.3.2. Read memory card

This command will read the Memory Card's Content from specified address.

Command

| Pseudo-APDU | | | | | |
|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **Byte Address** | **MEM_L** |
| Read Memory Card | FFh | | 00h | | |

Where:

**INS** (1 byte)

For reading zone 00b, INS = B0h

For reading zone 01b, INS = B1h

For reading zone 10b, INS = B2h

For reading zone 11b, INS = B3h

For reading fuse, INS = B4h

**Byte Address** (1 byte)

Memory address location of the memory card.

**MEM_L** (1 byte)

Length of data to be read from the memory card.

Response

| Response | Byte 1 | … | … | Byte N | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.3.3. Write memory card

This command writes the memory card's content from a specified address.

Command

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **Byte Address** | **MEM_L** | **Byte 1** | **…** | **…** | **Byte N** |
| Write Memory Card | FFh | | 00h | | | | | | |

Where:

**INS** (1 byte)

For reading zone 00b, INS = D0h

For reading zone 01b, INS = D1h

For reading zone 10b, INS = D2h

For reading zone 11b, INS = D3h

For reading fuse, INS = D4h

**Byte Address** (1 byte)

Memory address location of the memory card.

**MEM_L** (1 byte)

Length of data to be written to the memory card

**Byte (1…N)** Data to be written to the memory card.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.3.4. Verify password

This command verifies whether the memory card's password matches the user's entered PIN.

Command

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **RP** | **PW (0)** | **PW (1)** | **PW (2)** |
| Verify Password | FFh | 20h | 00h | | 03h | | | | |

Where:

**PW (0), PW (1), PW (2)** = Password to be sent to memory card.

**P2** (1 Byte)

**=** 0000 00r pb

Where the two bits "r p" indicates the password to compare

r = 0: Write password,

r = 1: Read password,

p = Password set number

r p = 01b for the secure code.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | ErrorCnt |

Where:

**SW1** = 90h

**ErrorCnt** (1 byte)

= Error Counter

FFh indicates the verification is correct. 00h indicates the password is locked (exceed maximum number of retries). Other values indicate the current verification is failed.

#### 5.1.3.5. Initialize authentication

This command initializes the memory card's authentication.

Command

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **Q (0)** | **Q (1)** | **…** | **Q (7)** |
| Initialize Authentication | FFh | 84h | 00h | 00h | 08h | | | | |

Where:

**Q (0…7)** (8 bytes)

**=** Host random number

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.3.6. Verify authentication

This command verifies the memory card's authentication.

Command

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **Ch (0)** | **Ch (1)** | **…** | **Ch (7)** |
| Verify Authentication | FFh | 82h | 00h | 00h | 08h | | | | |

Where:

**Ch (0…7)** (8 bytes)

**=** Host challenge

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.4. Memory Card – ATMEL AT88SC1608

#### 5.1.4.1. Select card type

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

Command

| Pseudo-APDU | | | | | | |
|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **Card Type** |
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 04h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.4.2. Read memory card

This command reads the memory card's content from a specified address.

Command

| Pseudo-APDU | | | | | |
|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **Zone Address** | **Byte Address** | **MEM_L** |
| Read Memory Card | FFh | | | | |

Where:

**INS** (1 byte)

For reading user zone, INS = B0h

For reading configuration zone or reading fuse, INS = B1h

**Zone Address** (1 byte)

= 00000 A10 A9 A8b, where A10 is the MSB of zone address

** don't care for reading fuse

**Byte Address** (1 byte)

= A7 A6 A5 A4 A3 A2 A1 A0b is the memory address location of the memory card

For reading fuse, Byte Address = 1000 0000b

**MEM_L** (1 byte)

Length of data to be read from the memory card.

Response

| Response | Byte 1 | … | … | Byte N | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.4.3. Write to memory card

This command writes the memory card's content from a specified address.

Command

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Command | Class | INS | Zone Address | Byte Address | MEM_L | Byte 1 | … | … | Byte N |
| Write Memory Card | FFh | | | | | | | | |

Where:

**INS** (1 byte)

For reading user zone, **INS = D0h**

For reading configuration zone or reading fuse, **INS = D1h**

**Zone Address** (1 byte)

= 00000 A10 A9 A8b, where A10 is the MSB of zone address

** don't care for reading fuse

**Byte Address** (1 byte)

= A7 A6 A5 A4 A3 A2 A1 A0b is the memory address location of the memory card

For reading fuse, Byte Address = 1000 0000b

**MEM_L** (1 byte)

Length of data to be written to the memory card

**Byte (1…N)** Data to be written to the memory card.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.4.4. Verify password

This command verifies if the memory card's password matches the user's entered PIN.

Command

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **RP** | **PW (0)** | **PW (1)** | **PW (2)** |
| Verify Password | FFh | 20h | 00h | 00h | 04h | | | | |

Where:

**PW (0), PW (1), PW (2)** = Password to be sent to memory card.

**RP** (1 byte)

**=** 0000 r p2 p1 p0b

Where the two bits "r p2 p1 p0" indicate the password to compare

r = 0 : Write password,

r = 1: Read password,

p2 p1 p0 = Password set number

r p2 p1 p0 = 0111b for the secure code.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | ErrorCnt |

Where:

**SW1** = 90h

**ErrorCnt** (1 byte)

= Error Counter

FFh indicates the verification is correct. 00h indicates the password is locked (exceed maximum number of retries). Other values indicate the current verification is failed.

### 5.1.4.5. Initialize authentication

This command initializes the memory card's authentication.

Command

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **Q (0)** | **Q (1)** | **…** | **Q (7)** |
| Initialize Authentication | FFh | 84h | 00h | 00h | 08h | | | | |

Where:

**Q (0…7)** (8 bytes)

**=** Host random number

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.4.6. Verify authentication

This command verifies the memory card's authentication.

Command

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **Ch (0)** | **Ch (1)** | **…** | **Ch (7)** |
| Verify Authentication | FFh | 82h | 00h | 00h | 08h | | | | |

Where:

**Ch (0…7)** (8 bytes)

**=** Host challenge

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.5. Memory Card – SLE4418/SLE4428/SLE5518/SLE5528

#### 5.1.5.1. Select card type

This command powers down/up the selected card in the reader, and then performs a card reset after.

Command

| Command | Class | INS | P1 | P2 | Lc | Card Type |
|---|---|---|---|---|---|---|
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 05h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.5.2. Read memory card

This command reads the memory card's content from a specified address.

Command

| Command | Class | INS | Byte Address | | MEM_L |
|---|---|---|---|---|---|
| | | | MSB | LSB | |
| Read Memory Card | FFh | B0h | | | |

Where:

**MSB Byte Address** (1 byte)

= 0000 00 A9 A8b is the memory address location of the memory card

**LSB Byte Address** (1 byte)

= A7 A6 A5 A4 A3 A2 A1 A0b is the memory address location of the memory card

**MEM_L** (1 byte)

Length of data to be read from the memory card

Response

| Response | Byte 1 | ... | ... | Byte N | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.5.3. Read presentation error counter memory card (for SLE4428 and SLE5528 only)

This command reads the presentation error counter for the secret code.

Command

| Command | Class | INS | P1 | P2 | MEM_L |
|---|---|---|---|---|---|
| Read Presentation Error Counter | FFh | B1h | 00h | 00h | 03h |

Response

| Response | ErrCnt | Dummy 1 | Dummy 2 | SW1 | SW2 |
|---|---|---|---|---|---|
| Result | | | | | |

Where:

**ErrCnt** (1 byte)

The value of the presentation error counter

FFh = indicates the verification is correct

00h = indicates the password is locked (exceeding the maximum number of retries)

Other values indicate the verification failed.

**Dummy 1, Dummy 2** (2 bytes)

Dummy data read from the card

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.5.4. Read protection bit

This command reads the protection bit.

Command

| Command | Class | INS | Byte Address | | MEM_L |
|---|---|---|---|---|---|
| | | | MSB | LSB | |
| Read Protection Bit | FFh | B2h | | | |

Where:

**MSB Byte Address** (1 byte)

The memory address location of the memory card

= 0000 00 A9 A8b

**LSB Byte Address** (1 byte)

The memory address location of the memory card

= A7 A6 A5 A4 A3 A2 A1 A0b

**MEM_L** (1 byte)

Length of protection bits read from the card, in multiples of 8 bits. The 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 B1 00 10 01h

Response

| Response | PROT 1 | … | … | PROT L | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**PROT (1..L)** Bytes containing the protection bits.

**SW1 SW2** = 90 00h if the operation is completed successfully.

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 response data:

0 = byte is write protected

1 = byte can be written

### 5.1.5.5. Write memory card

This command writes the memory card's content to a specified address.

Command

| Command | Class | INS | Byte Address | | MEM_L | Byte 1 | … | … | Byte N |
|---|---|---|---|---|---|---|---|---|---|
| | | | MSB | LSB | | | | | |
| Write Memory Card | FFh | D0h | | | | | | | |

Where:

**MSB Byte Address** (1 byte)

= 0000 00 A9 A8b is the memory address location of the memory card

**LSB Byte Address** (1 byte)

= A7 A6 A5 A4 A3 A2 A1 A0b is the memory address location of the memory card

**MEM_L** (1 byte)

Length of data to be written to the memory card

**Byte (1…N)** Data to be written to the memory card.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.5.6. Write protection memory card

Each byte specified in the command is compared with the bytes stored in the specific address, and if the data matches, the corresponding protection bit is irreversibly programmed to '0'.

Command

| Command | Class | INS | Byte Address | | MEM_L | Byte 1 | … | … | Byte N |
|---|---|---|---|---|---|---|---|---|---|
| | | | MSB | LSB | | | | | |
| Write Protection Memory Card | FFh | D1h | | | | | | | |

Where:

**MSB Byte Address** (1 byte)

= 0000 00 A9 A8b is the memory address location of the memory card

**LSB Byte Address** (1 byte)

= A7 A6 A5 A4 A3 A2 A1 A0b is the memory address location of the memory card

**MEM_L** (1 byte)

Length of data to be written to the memory card

**Byte (1…N)** Byte values compared with the data in the card starting at the 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

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.5.7. Present code memory card (for SLE44428 and SLE5528 only)

This command submits 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 ‘0’.
2. Present the specified code to the card.
3. Try to erase the presentation error counter.

Command

| Command | Class | INS | P1 | P2 | MEM_L | Code | |
|---|---|---|---|---|---|---|---|
| | | | | | | Byte 1 | Byte 2 |
| Present Code Memory Card | FFh | 20h | 00h | 00h | 02h | | |

Where:

**Code** (3 bytes)

Secret code (PIN)

Response

| Response | Data Out | |
|---|---|---|
| Result | 90h | ErrorCnt |

Where:

**ErrorCnt** (1 byte)

Error Counter

FFh = indicates the verification is correct.

00h = indicates the password is locked (exceeding maximum number of retries).

Other values indicate the verification failed.

### 5.1.6. Memory Card – SLE4432/SLE4442/SLE5532/SLE5542

#### 5.1.6.1. Select card type

This command powers down/up the selected card in the reader, and then performs a card reset after.

Command

| Command | Class | INS | P1 | P2 | Lc | Card Type |
|---|---|---|---|---|---|---|
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 06h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully

#### 5.1.6.2. Read memory card

This command reads the memory card's content from a specified address.

Command

| Command | Class | INS | P1 | Byte Address | MEM_L |
|---|---|---|---|---|---|
| Read Memory Card | FFh | B0h | 00h | | |

Where:

**Byte Address** (1 byte)

=A7 A6 A5 A4 A3 A2 A1 A0b is the memory address location of the memory card

**MEM_L** (1 byte)

Length of data to be read from the memory card

Response

| Response | Byte 1 | … | … | Byte N | PROT1 | PROT2 | PROT3 | PROT4 | SW1 | SW2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Result | | | | | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**PROT (1…4)** Bytes containing the protections bits from protection.

**SW1 SW2** = 90 00h if the operation is completed successfully.

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 response data:

0 = byte is write protected

1 = byte can be written

### 5.1.6.3. Read presentation error counter memory card (for SLE4442 and SLE5542 only)

This command reads the presentation error counter for the secret code.

Command

| Command | Class | INS | P1 | P2 | MEM_L |
|---|---|---|---|---|---|
| Read Presentation Error Counter | FFh | B1h | 00h | 00h | 04h |

Response

| Response | ErrCnt | Dummy 1 | Dummy 2 | Dummy 3 | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**ErrCnt** (1 byte)

The value of the presentation error counter

07h = indicates the verification is correct.

00h = indicates the password is locked (exceeding the maximum number of retries).

Other values indicate the verification failed.

**Dummy 1, Dummy 2, Dummy 3** (3 bytes)

Dummy data read from the card

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.6.4. Read Protection Bit

This command reads the protection bits for the first 32 bytes.

Command

| Command | Class | INS | P1 | P2 | MEM_L |
|---|---|---|---|---|---|
| Read Protection Bit | FFh | B2h | 00h | 00h | 04h |

Response

| Response | PROT 1 | PROT 2 | PROT 3 | PROT 4 | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**PROT (1..4)** Bytes containing the protection bits.

**SW1 SW2** = 90 00h if the operation is completed successfully.

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** protection bit of bytes in the response data:

0 = byte is write protected

1 = byte can be written

### 5.1.6.5. Write memory card

This command writes the memory card's content to a specified address.

Command

| Command | Class | INS | P1 | Byte Address | MEM_L | Byte 1 | … | … | Byte N |
|---|---|---|---|---|---|---|---|---|---|
| Write Memory Card | FFh | D0h | 00h | | | | | | |

Where:

**Byte Address** (1 byte)

= A7 A6 A5 A4 A3 A2 A1 A0b is the memory address location of the memory card

**MEM_L** (1 byte)

Length of data to be written to the memory card

**Byte (1…N)** Data to be written to the memory card.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.6.6. Write protection memory card

Each of the byte specified in the command is compared with the bytes stored in the specific address and if the data matches, the corresponding protection bit is irreversibly programmed to '0'.

Command

| Command | Class | INS | P1 | Byte Address | MEM_L | Byte 1 | … | … | Byte N |
|---|---|---|---|---|---|---|---|---|---|
| Write Protection Memory Card | FFh | D1h | 00h | | | | | | |

Where:

**Byte Address** (1 byte)

= 000A4 A3 A2 A1b (00h – 1Fh) is the protection memory address location of the memory card

**MEM_L** (1 byte)

Length of data to be written to the memory card

**Byte (1…N)** Byte values compared with the data in the card starting at the 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

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.6.7. Present code memory card (for SLE4442 and SLE5542 only)

This command submits 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 bit '0'.
2. Present the specified code to the card.
3. Try to erase the presentation error counter.

Command

| Command | Class | INS | P1 | P2 | MEM_L | Code | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Byte 1 | Byte 2 | Byte 3 |
| Present Code Memory Card | FFh | 20h | 00h | 00h | 03h | | | |

Where:

**Code** (3 bytes)

Secret code (PIN)

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | ErrorCnt |

Where:

**ErrorCnt** (1 byte)

Error Counter

07h = indicates the verification is correct.

00h = indicates the password is locked (exceeding the maximum number of retries).

Other values indicate the verification failed.

### 5.1.6.8. Change code memory card (for SLE4442 and SLE5542 only)

This command writes the specified data as the new secret code in the card. The existing secret code must be presented to the card using the "Present Code" command prior to the execution of this command.

Command

| Command | Class | INS | P1 | P2 | MEM_L | Code | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Byte 1 | Byte 2 | Byte 3 |
| Change Code Memory Card | FFh | D2h | 00h | 01h | 03h | | | |

Where:

**Code** (3 bytes)

Secret code (PIN)

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.7. Memory Card – SLE4406/SLE4436/SLE5536/SLE6636

#### 5.1.7.1. Select card type

This command powers down/up the selected card in the reader, and then performs a card reset after.

Command

| Command | Class | INS | P1 | P2 | Lc | Card Type |
|---|---|---|---|---|---|---|
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 07h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.7.2. Read Memory Card

This command reads the memory card's content from a specified address.

Command

| Command | Class | INS | P1 | Byte Address | MEM_L |
|---|---|---|---|---|---|
| Read Memory Card | FFh | B0h | 00h | | |

Where:

**Byte Address** (1 byte)
Memory address location of the memory card

**MEM_L** (1 byte)
Length of data to be read from the memory card

Response

| Response | Byte 1 | ... | ... | Byte N | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.7.3. 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 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. This mode can therefore only be used for updating the counter value in the card.

**c. Write with backup enabled (for 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. This mode can therefore 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

| Command | Class | INS | P1 | Byte Address | MEM_L | Mode | Byte |
|---|---|---|---|---|---|---|---|
| Read Memory Card | FFh | D0h | 00h | | 02h | | |

Where:

**Byte Address** (1 byte)

Memory address location of the memory card

**Mode** (1 byte)

Specifies the write mode and backup option

00h = Write.

01h = Write with carry.

02h = Write with backup enabled (for SLE4436, SLE5536 and SLE6636 only).

03h = Write with carry and with backup enabled (for SLE4436, SLE5536 and SLE6636 only).

**Byte** (1 byte)

Byte value to be written to the card

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.7.4. Present code memory card

This command submits the secret code to the memory card to enable card personalization mode. The following actions are executed:

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

Command

| Command | Class | INS | P1 | P2 | MEM_L | Code | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Addr | Byte 1 | Byte 2 | Byte 3 |
| Present Code Memory Card | FFh | 20h | 00h | 00h | 04h | 09h | | | |

Where:

**Addr** (1 byte)
Byte address of the presentation counter in the card

**Code** (3 bytes)
Secret code (PIN)

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.7.5. Authenticate memory card (for SLE4436, SLE5536 and SLE6636 only)

This command reads the authentication certificate from the card. The following actions are executed:

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 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 is 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

| Command | Class | INS | P1 | P2 | MEM_L | Code | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | **Key** | **CLK_CNT** | **Byte 1** | **…** | **Byte 6** |
| Send Authentication Certificate | FFh | 84h | 00h | 00h | 08h | | | | | |

Where:

**Key** (1 byte)

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 (for SLL5536 and SLE6636 only).

81h = Key 2 with cipher block chaining (for SLL5536 and SLE6636 only).

**CLK_CNT** (1 byte)

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 (A0h).

**Byte (1...6)** Card challenge data.

Response

| Response | SW1 | SW2 |
|---|---|---|
| Result | 61h | 02h |

**Step 2:** Get the authentication data (Get Response).

Command

| Command | Class | INS | P1 | P2 | MEM_L |
|---|---|---|---|---|---|
| Get Authentication Data | FFh | C0h | 00h | 00h | 02h |

Response

| Response | Cert | | SW1 | SW2 |
|---|---|---|---|---|
| Result | | | | |

Where:

**Cert** (2 bytes)

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 the operation is completed successfully.

### 5.1.8. Memory Card – SLE4404

### 5.1.8.1. Select card type

This command powers down/up the selected card in the reader, and then performs a card reset after.

Command

| Command | Class | INS | P1 | P2 | Lc | Card Type |
|---|---|---|---|---|---|---|
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 08h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.8.2. Read memory card

This command reads the memory card's content from a specified address.

Command

| Command | Class | INS | P1 | Byte Address | MEM_L |
|---|---|---|---|---|---|
| Read Memory Card | FFh | B0h | 00h | | |

Where:

**Byte Address** (1 byte)
Memory address location of the memory card

**MEM_L** (1 byte)
Length of data to be read from the memory card

Response

| Response | Byte 1 | ... | ... | Byte N | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.8.3. Write memory card

This command writes the memory card's content to a specified address. 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

| Command | Class | INS | P1 | Byte Address | MEM_L | Byte 1 | … | … | Byte N |
|---|---|---|---|---|---|---|---|---|---|
| Write Memory Card | FFh | D0h | 00h | | | | | | |

Where:

**Byte Address** (1 byte)

Memory address location of the memory card

**MEM_L** (1 byte)

Length of data to be written to the memory card

**Byte (1…N)** Data to be written to the memory card.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.8.4. Erase scratch pad memory card

This command erases 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'.

Command

| Command | Class | INS | P1 | Byte Address | MEM_L |
|---|---|---|---|---|---|
| Erase Scratch Pad | FFh | D2h | 00h | | 00h |

Where:

**Byte Address** (1 byte)

Memory byte address location of the scratch pad. Typical value is 02h.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.8.5. Verify user code

This command submits the User Code (2 bytes) to the inserted card. The User Code enables access to the memory 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 '0'.
3. Erase the presentation error counter. The Error User Counter can be erased when the submitted code is correct.

Command

| Command | Class | INS | Error Counter LEN | Byte Address | MEM_L | Code | |
|---|---|---|---|---|---|---|---|
| | | | | | | Byte 1 | Byte 2 |
| Verify User Code | FFh | 20h | 04h | 08h | 02h | | |

Where:

**Error Counter LEN** (1 byte)
Length of presentation error counter in bits

**Byte Address** (1 byte)
Byte address of the key in the card

**Code** (1 byte)
User Code

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.
= 63 00h if there are no more retries left.

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

### 5.1.8.6. Verify memory code

This command submits memory code (4 bytes) to the inserted card. The 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.

   ***Note:*** *The Memory Error Counter cannot be erased.*

Command

| Command | Class | INS | Error Counter LEN | Byte Address | MEM_L | Code | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
| Verify Memory Code | FFh | 20h | 40h | 28h | 04h | | | | |

Where:

**Error Counter LEN** (1 byte)
Length of presentation error counter in bits

**Byte Address** (1 byte)
Byte address of the key in the card

**Code** (4 bytes)
Memory Code

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.
= 63 00h if there are no more retries left.

***Note:*** *After SW1 SW2 = 90 00h has been received, read back the User Error Counter to check whether the Verify Memory Code is correct. If all data in Application Area is erased and is equal to ‘FFh’, the previous verification is successful.*

### 5.1.9. Memory Card – AT88SC101/AT88SC102/AT88SC1003

#### 5.1.9.1. Select card type

This command powers down and up the selected card inserted in the card reader and performs a card reset.

Command

| Pseudo-APDU | | | | | | |
|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **P2** | **Lc** | **Card Type** |
| Select Card Type | FFh | A4h | 00h | 00h | 01h | 09h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

#### 5.1.9.2. Read Memory Card

This command reads the memory card's content from specified address.

Command

| Pseudo-APDU | | | | | |
|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **Byte Address** | **MEM_L** |
| Read Memory Card | FFh | B0h | 00h | | |

Where:

**Byte Address** (1 byte)
Memory address location of the memory card.

**MEM_L** (1 byte)
Length of data to be read from the memory card.

Response

| Response | Byte 1 | ... | ... | Byte N | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | | | | | | |

Where:

**Byte (1…N)** Data read from memory card.

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.9.3. Write Memory Card

This command writes 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

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **Byte Address** | **MEM_L** | **Byte 1** | **…** | **…** | **Byte N** |
| Write Memory Card | FFh | D0h | 00h | | | | | | |

Where:

**Byte Address** (1 byte)
Memory address location of the memory card.

**MEM_L** (1 byte)
Length of data to be written to the memory card

**Byte (1…N)** Byte value to be written to the card.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.9.4. Erase non-application zone

This command erases the data in non-application zones. The EEPROM memory is organized into 16 bit words. Although erases are performed on single bits 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:

- Erase Application Zone With Erase command as specified
- Erase Application Zone With Write and Erase command as specified
- Verify Security Code commands as specified

Command

| Pseudo-APDU | | | | | |
|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **P1** | **Byte Address** | **MEM_L** |
| Erase Non-Application Zone | FFh | D2h | 00h | | 00h |

Where:

**Byte Address** (1 byte)

Memory byte address location of the word to be erased.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

### 5.1.9.5. Erase Application Zone with Erase

This command can be used in the following cases:

- AT88SC101: To erase the data in Application Zone with EC Function Disabled
- AT88SC102: To erase the data in Application Zone 1
- AT88SC102: To erase the data in Application Zone 2 with EC2 Function Disabled
- AT88SC1003: To erase the data in Application Zone 1
- AT88SC1003: To erase the data in Application Zone 2 with EC2 Function Disabled
- 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.
2. Erase the presentation error counter. The data in corresponding Application Zone can be erased when the submitted code is correct.

Command

| Pseudo-APDU | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Class** | **INS** | **Error Counter LEN** | **Byte Address** | **MEM_L** | **CODE** | | | | |
| | | | | | | **Byte 1** | **Byte 2** | **…** | **…** | **Byte N** |
| Erase Application Zone with Erase | FFh | 20h | 00h | | | | | | | |

Where:

**Error Counter LEN** (1 byte)

**=** Length of presentation error counter in bits. The value should be 00h always.

**Byte Address** (1 byte)

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

**MEM_L** (1 byte)

= Length of the Erase Key. Please refer to the table below for the correct value.

**CODE (1…N)** = Erase Key

| Cases | 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 |
| AT88SC1003: Erase Application Zone 2 with EC2 function disabled | 5Ch | 04h |
| AT88SC1003: Erase Application Zone 3 | C0h | 06h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

***Note:*** *After SW1SW2 = 90 00h been received, read back the data in Application Zone can check whether the Erase Application Zone with Erase is correct. If all data in Application Zone is erased and equals to “FFh”, the previous verification is success.*

### 5.1.9.6. Erase Application Zone with Write and Erase

This command can be used in the following cases:

- AT88SC101: To erase the data in Application Zone with EC Function Enabled
- AT88SC102: To erase the data in Application Zone 2 with EC2 Function Enabled
- 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 unblown 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

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Command | Class | INS | Error Counter LEN | Byte Address | MEM_L | CODE | | | |
| | | | | | | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
| Erase Application Zone with Write and Erase | FFh | 20h | 80h | | 04h | | | | |

Where:

**Error Counter LEN** (1 byte)

**=** Length of presentation error counter in bits. The value should be 80h always.

**Byte Address** (1 byte)

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

**CODE** (4 bytes)

= Erase Key

| Cases | Byte Address |
|---|---|
| AT88SC101 | 96h |
| AT88SC102 | 9Ch |
| AT88SC1003 | 5Ch |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

**=** 63 00 if there is no more retry chance.

***Note:*** *After SW1SW2 = 90 00 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 equals to "FFh", the previous verification is success.*

### 5.1.9.7. Verify Security Code

This command submits 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

| Pseudo-APDU | | | | | | | |
|---|---|---|---|---|---|---|---|
| Command | Class | INS | Error Counter LEN | Byte Address | MEM_L | CODE | |
| | | | | | | Byte 1 | Byte 2 |
| Verify Security Code | FFh | 20h | 08h | 0Ah | 02h | | |

Where:

**Error Counter LEN** (1 byte)
**=** Length of presentation error counter in bits.

**Byte Address** (1 byte)
= Byte address of the key in the card**.**

**CODE** (2 bytes)
= Security Code

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.
**=** 63 00 if there is no more retry chance.

***Note:*** *After SW1SW2 = 90 00h been received, read back the Security Code Attempts Counter (SCAC) can check whether the Verify User Code is correct. If SCAC is erased and equals to "FFh", the previous verification is success.*

### 5.1.9.8. Blow Fuse

This command blows 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

| Pseudo-APDU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Command | Class | 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 |
| Blown Fuse | FFh | 05h | 00h | 00h | 04h | | | 01h | 00h<br>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** (1 byte)

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

**State of RST Pin** (1 byte)

= 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 |

**Table 4**: Blown Fuse Code Values

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h if the operation is completed successfully.

**=** 63 00 if there is no more retry chance.

## 5.2. Contactless Smart Card Protocol

### 5.2.1. ATR Generation

If the reader detects a PICC, an ATR will be sent to the PC/SC driver for identifying the PICC.

### 5.2.2. ATR Format for ISO 14443 Part 3 PICCs

| Byte | Value (Hex) | Designation | Description |
|---|---|---|---|
| 0 | 3B | Initial Header | - |
| 1 | 8N | T0 | Higher nibble 8 means: no TA1, TB1, TC1 only TD1 is following.<br>Lower nibble N is the number of historical bytes (HistByte 0 to HistByte N-1) |
| 2 | 80 | TD1 | Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following.<br>Lower nibble 0 means T = 0 |
| 3 | 01 | TD2 | Higher nibble 0 means no TA3, TB3, TC3, TD3 following.<br>Lower nibble 1 means T = 1 |
| 4 to 3+N | 80 | T1 | Category indicator byte, 80 means A status indicator may be present in an optional COMPACT-TLV data object |
| | 4F | Tk | Application identifier Presence Indicator |
| | 0C | | Length |
| | RID | | Registered Application Provider Identifier (RID) # A0 00 00 03 06h |
| | SS | | Byte for standard |
| | C0.. C1 | | Bytes for card name |
| | 00 00 00 00 | RFU | RFU # 00 00 00 00h |
| 4+N | UU | TCK | Exclusive-oring of all the bytes T0 to Tk |

**Table 5**: ISO 14443 Part 3 ATR Format

**Example:**

ATR for MIFARE 1K = {3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6Ah}

| ATR | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Initial Header** | **T0** | **TD1** | **TD2** | **T1** | **Tk** | **Length** | **RID** | **Standard** | **Card Name** | **RFU** | **TCK** |
| 3Bh | 8Fh | 80h | 01h | 80h | 4Fh | 0Ch | A0<br>00<br>00<br>03<br>06h | 03h | 00h<br>01h | 00<br>00<br>00<br>00h | 6Ah |

Where:

| | |
|---|---|
| **Length (YY)** | = 0Ch |
| **RID** | = A0 00 00 03 06h (PC/SC Workgroup) |
| **Standard (SS)** | = 03h (ISO 14443A, Part 3) |
| **Card Name (C0 ... C1)** | = [00 01h] (MIFARE 1K) |
| | [00 02h] (MIFARE 4K) |
| | [00 03h] (MIFARE Ultralight) |
| | [00 26h] (MIFARE Mini) |
| | [FF 28h] JCOP 30 |
| | FF SAK undefined tags |

### 5.2.3. ATR Format for ISO 14443 Part 4 PICCs

<table>
<tr><th>Byte</th><th>Value (Hex)</th><th>Designation</th><th>Description</th></tr>
<tr><td>0</td><td>3B</td><td>Initial Header</td><td>-</td></tr>
<tr><td>1</td><td>8N</td><td>T0</td><td>Higher nibble 8 means: no TA1, TB1, TC1 only TD1 is following.<br>Lower nibble N is the number of historical bytes (HistByte 0 to HistByte N-1)</td></tr>
<tr><td>2</td><td>80</td><td>TD1</td><td>Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following.<br>Lower nibble 0 means T = 0</td></tr>
<tr><td>3</td><td>01</td><td>TD2</td><td>Higher nibble 0 means no TA3, TB3, TC3, TD3 following.<br>Lower nibble 1 means T = 1</td></tr>
<tr><td rowspan="2">4 to<br>3 + N</td><td>XX</td><td>T1</td><td rowspan="2">Historical Bytes:<br>ISO 14443A:<br>The historical bytes from ATS response. Refer to the ISO 14443-4 specification.<br><br>ISO 14443B:
<table>
<tr><th>Byte1-4</th><th>Byte5-7</th><th>Byte8</th></tr>
<tr><td>Application Data from ATQB</td><td>Protocol Info Byte from ATQB</td><td>Higher nibble=MBLI from ATTRIB command<br>Lower nibble (RFU)=0</td></tr>
</table></td></tr>
<tr><td>XX<br>XX<br>XX</td><td>Tk</td></tr>
<tr><td>4+N</td><td>UU</td><td>TCK</td><td>Exclusive-oring of all the bytes T0 to Tk</td></tr>
</table>

**Table 6**: ISO 14443 Part 4 ATR Format

**Example 1**: Consider the ATR from MIFARE DESFire as follows:

DESFire (ATR) = 3B 81 80 01 80 80h (6 bytes of ATR)

***Note:*** *Use the APDU "FF CA 01 00 00h" to distinguish the ISO 14443A-4 and ISO 14443B-4 PICCs and retrieve the full ATS if available. The ATS is returned for ISO 14443A-3 or ISO 14443B-3/4 PICCs.*

APDU Command = FF CA 01 00 00h

APDU Response = 06 75 77 81 02 90 00h

ATS = {06 75 77 81 02 80h}

**Example 2**: Consider the ATR from EZ-Link as follows:

EZ-Link (ATR) = 3B 88 80 01 1C 2D 94 11 F7 71 85 00 BEh

Application Data of ATQB = 1C 2D 94 11h

Protocol Information of ATQB = F7 71 85h

MBLI of ATTRIB = 00h

### 5.2.4. Pseudo APDUs for Contactless Interface

#### 5.2.4.1. Get data

This command returns the serial number or ATS of the "connected PICC."

Command

| Command | Class | INS | P1 | P2 | Le |
|---|---|---|---|---|---|
| Get Data | FFh | CAh | 00h<br>01h | 00h | 00h<br>(Full Length) |

Get UID Response if P1 = 00h

| Response | UID | … | … | UID | SW1 | SW2 |
|---|---|---|---|---|---|---|
| Result | LSB | | | MSB | | |

Get ATS Response if P1 = 01h (for ISO 14443-A cards only)

| Response | Data Out | | |
|---|---|---|---|
| Result | ATS | SW1 | SW2 |

Response Code

| Results | SW1 SW2 | Meaning |
|---|---|---|
| Success | 90 00h | The operation is completed successfully. |
| Warning | 62 82h | End of 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 |

**Example 1**: To get the serial number of the connected PICC:

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

**Example 2**: To get the ATS of the connected ISO 14443-A PICC:

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

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

### 5.2.4.3. Load authentication keys

This command loads the authentication keys into the reader. The authentication keys are used to authenticate the specified sector of the MIFARE 1K/4K Memory Card. Two kinds of authentication key locations are provided, volatile and non-volatile key locations.

Command

| Command | Class | INS | P1 | P2 | Le | Data In |
|---|---|---|---|---|---|---|
| Load Authentication Keys | FFh | 82h | Key Structure | Key Number | 06h | Key |

Where:

**Key Structure** (1 byte)

00h = Key is loaded into the reader's volatile memory

20h = Key is loaded into the reader's non-volatile memory

Other = Reserved.

**Key Number** (1 byte)

00h – 1Fh = Non-volatile memory for storing keys. The keys are permanently stored in the reader and will not be erased even if the reader is disconnected from the PC. It can store up to 32 keys inside the reader non-volatile memory.

20h (Session Key) = Volatile memory for temporarily storing keys. The keys will be erased when the reader is disconnected from the PC. Only one volatile memory is provided. The volatile key can be used as a session key for different sessions. Default value = FF FF FF FF FF FFh.

**Key** (6 bytes)

The key value loaded into the reader.

E.g. {FF FF FF FF FF FFh}

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h means the operation is completed successfully.

= 63 00h means the operation failed.

**Example1**:

Load a key { FF FF FF FF FF FFh } into the non-volatile memory location 05h.

APDU = {FF 82 20 05 06 FF FF FF FF FF FFh}

Load a key { FF FF FF FF FF FFh } into the volatile memory location 20h.

APDU = {FF 82 00 20 06 FF FF FF FF FF FFh}

***Notes***:

1. *The application should know all the keys being used. It is recommended to store all the required keys to the non-volatile memory for security reasons. The contents of both volatile and non-volatile memories are not readable by any application.*
2. *The content of the volatile memory "Session Key 20h" will remain valid until the reader is reset or powered-off. The session key is useful for storing any key value that is changing from time to time. The session key is stored in the "Internal RAM", while the non-volatile keys are stored in "EEPROM" that is relatively slower than the "Internal RAM".*
3. *It is not recommended to use the "non-volatile key locations 00-1Fh" to store any "temporary key" that will be changed frequently. The "non-volatile keys" are supposed to be used for storing any "key value" that will not change frequently. If the "key value" is supposed to be changed from time to time, store the "key value" to the "volatile key location 20h" instead.*

### 5.2.4.3.1. Authentication for MIFARE 1K/4K

This command is used to authenticate the MIFARE 1K/4K card (PICC) using the keys stored in the reader. Two types of authentication keys are used: Type_A and Type_B.

Command

| Command | Class | INS | P1 | P2 | P3 | Data In |
|---|---|---|---|---|---|---|
| Authentication<br>6 bytes (Obsolete) | FFh | 88h | 00h | Block Number | Key Type | Key Number |

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| Authentication<br>10 bytes | FFh | 86h | 00h | 00h | 05h | Authenticate Data Bytes |

Where:

**Authenticate Data Bytes** (5 bytes)

| Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 |
|---|---|---|---|---|
| Version<br>01h | 00h | Block Number | Key Type | Key Number |

Where:

**Block Number** (1 byte)

The memory block to be authenticated.

***Note:** For MIFARE 1K card, it has a total of 16 sectors and each sector consists of 4 consecutive blocks. For example, 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.*

**Key Type** (1 byte)

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

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

**Key Number** (1 byte)

00h – 1Fh = Non-volatile memory for storing keys. The keys are permanently stored in the reader and will not be erased even if the reader is disconnected from the PC. It can store up to 32 keys inside the reader non-volatile memory.

20h (Session Key) = Volatile memory for temporarily storing keys. The keys will be erased when the reader is disconnected from the PC. Only 1 volatile memory is provided. The volatile key can be used as a session key for different sessions. Default value = FF FF FF FF FF FFh.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h means the operation is completed successfully.

= 63 00h means the operation failed.

| Sectors<br>(Total of 16 sectors. Each sector consists of 4 consecutive blocks) | Data Blocks<br>(3 blocks, 16 bytes per block) | Trailer Block<br>(1 block, 16 bytes) |
|---|---|---|
| Sector 0 | 00h ~ 02h | 03h |
| Sector 1 | 04h ~ 06h | 07h |
| .. | | |
| .. | | |
| Sector 14 | 38h ~ 0Ah | 3Bh |
| Sector 15 | 3Ch ~ 3Eh | 3Fh |

1 KB

**Table 7**: MIFARE 1K Memory Map

| Sectors<br>(Total of 32 sectors. Each sector consists of 4 consecutive blocks) | Data Blocks<br>(3 blocks, 16 bytes per block) | Trailer Block<br>(1 block, 16 bytes |
|---|---|---|
| Sector 0 | 00h ~ 02h | 03h |
| Sector 1 | 04h ~ 06h | 07h |
| ... | | |
| ... | | |
| Sector 30 | 78h ~ 7Ah | 7Bh |
| Sector 31 | 7Ch ~ 7Eh | 7Fh |

2 KB

| Sectors<br>(Total of 32 sectors. Each sector consists of 4 consecutive blocks) | Data Blocks<br>(3 blocks, 16 bytes per block) | Trailer Block<br>(1 block, 16 bytes |
|---|---|---|
| Sector 32 | 80h ~ 8Eh | 8Fh |
| Sector 33 | 90h ~ 9Eh | 9Fh |
| ... | | |
| ... | | |
| Sector 38 | E0h ~ EEh | EFh |
| Sector 39 | F0h ~ FEh | FFh |

2 KB

**Table 8**: MIFARE 4K Memory Map

**Example 1:**

To authenticate Block 04h with the following characteristics: Key A, key number 00h, from PC/SC V2.01 (Obsolete).

APDU = { FF 88 00 04 60 00h }

**Example 2:**

Similar to the previous example, to authenticate Block 04h with the following characteristics: Key A, key number 00h, from PC/SC V2.07.

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

***Note:*** *MIFARE® Ultralight does not need authentication since it provides free access to the user data area.*

| 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 9**: MIFARE Ultralight Memory Map

### 5.2.4.3.2. Read binary blocks

This command retrieves multiple "data blocks" from the PICC. The data block/trailer must be authenticated first before executing the "Read Binary Blocks" command.

Command

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

Where:

**Block Number** (1 byte)

Starting Block

**Number of Bytes to Read** The length of the bytes to be read can be a multiple of 16 bytes for MIFARE 1K/4K or a multiple of 4 bytes for MIFARE Ultralight (1 Byte).

Maximum of 16 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).

**Example 1**: 10h (16 bytes). Starting block only. (Single Block Mode)

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

***Note:*** *For security considerations, 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.*

Response

| Response | Data Out | | |
|---|---|---|---|
| Result | Data (Multiple of 4 or 16 bytes) | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h means the operation is completed successfully.

= 63 00h means the operation failed.

**Example 1**: Read 16 bytes from the binary block 04h (MIFARE 1K or 4K).

APDU = { FF B0 00 04 10h }

**Example 2**: Read 240 bytes starting from the binary block 80h (MIFARE 4K). Block 80h to Block 8Eh (15 blocks).

APDU = { FF B0 00 80 F0 }

### 5.2.4.3.3. Update binary blocks

This command writes multiple data blocks into the PICC. The data block/trailer block must be authenticated first before executing the “Update Binary Blocks” command.

Command

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

Where:

**Block Number** (1 byte)

Starting Block

**Block Data** Multiple of 16 + 2 Bytes, or 6 Bytes. Data to be written into the binary blocks.

**Number of Bytes to Read** The length of the bytes to be read can be a multiple of 16 bytes for MIFARE 1K/4K or a multiple of 4 bytes for MIFARE Ultralight (1 byte).

Maximum of 16 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).

**Example 1:** 10h (16 bytes). Starting block only. (Single Block Mode)

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

***Note:*** *For security considerations, 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.*

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h means the operation is completed successfully.

= 63 00h means the operation failed.

**Example 1**: Update the binary block 04h of MIFARE 1K/4K with Data {00 01 .. 0Fh}

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

**Example 2**: Update the binary block 04h of MIFARE Ultralight with Data { 00 01 02 03h }

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

#### 5.2.4.3.4. Value block operation (Increment, Decrement, Store)

This command manipulates value-based transactions (e.g., increment a value block, etc.).

Command

| Command | Class | INS | P1 | P2 | Lc | Data In | |
|---|---|---|---|---|---|---|---|
| Value Block Operation | FFh | D7h | 00h | Block Number | 05h | VB_OP | VB_Value<br>(4 Bytes)<br>{MSB…LSB} |

Where:

**Block Number** (1 byte)

Value Block to be manipulated

**VB_OP** (1 byte)

Value block operation

00h = Store *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 blocks.

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

**VB_Value** (4 bytes)

The value used for manipulation. The value is a signed long integer.

**Example 1:** Decimal - 4 = { FF FF FF FCh }

| VB_Value | | | |
|---|---|---|---|
| MSB | | | LSB |
| FFh | FFh | FFh | FCh |

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

| VB_Value | | | |
|---|---|---|---|
| MSB | | | LSB |
| 00h | 00h | 00h | 01h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h means the operation is completed successfully.

= 63 00h means the operation failed.

### 5.2.4.3.5. Read value block

This command retrieves the value from the value block. This command is only valid for value blocks.

Command

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

Where:

**Block Number** (1 byte)

The value block to be accessed.

Response

| Response | Data Out | | |
|---|---|---|---|
| Result | Value<br>{MSB … LSB} | SW1 | SW2 |

Where:

**Value** (4 bytes)

The value returned from the cards. The value is a signed long integer.

**Example 1:** Decimal - 4 = { FF FF FF FCh }

| VB_Value | | | |
|---|---|---|---|
| MSB | | | LSB |
| FFh | FFh | FFh | FCh |

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

| VB_Value | | | |
|---|---|---|---|
| MSB | | | LSB |
| 00h | 00h | 00h | 01h |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h means the operation is completed successfully.

= 63 00h means the operation failed.

### 5.2.4.3.6. Copy value block

This command copies a value from a value block to another value block.

Command

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

Where:

**Source Block Number** (1 byte)

Block number where the value will come from and copied to the target value block.

**Target Block Number** (1 byte)

Block number where the value from the source block will be copied to. The source and target value blocks must be in the same sector.

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h means the operation is completed successfully.

= 63 00h means the operation failed.

**Example 1**: Store a value "1" into block 05h

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

**Example 2**: Read the value block 05h

APDU = {FF B1 00 05 00h}

**Example 3**: Copy the value from value block 05h to value block 06h

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

**Example 4**: Increment the value block 05h by "5"

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

### 5.2.4.4. Access PC/SC-compliant tags (ISO 14443-4)

Basically, all ISO 14443-4–compliant cards (PICCs) can understand the ISO 7816-4 APDUs. The ACR1281U-C1 reader only needs to communicate with the ISO 14443-4–compliant cards through exchanging ISO 7816-4 APDUs and responses. ACR1281U-C1 will handle the ISO 14443 Parts 1-4 Protocols internally.

The MIFARE Classic 1K/4K, MIFARE Mini and MIFARE Ultralight tags are supported through the T=CL emulation. Simply treat the MIFARE tags as standard ISO 14443-4 tags. For more information, see **PICC Commands (T=CL Emulation) for MIFARE 1K/4K Memory Cards**.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In | Le |
|---|---|---|---|---|---|---|---|
| ISO 7816 Part 4 Command | | | | | Length of the Data In | | Expected Length of the Response Data |

Response

| Response | Data Out | |
|---|---|---|
| Result | SW1 | SW2 |

Where:

**SW1 SW2** = 90 00h means the operation is completed successfully.

= 63 00h means the operation failed.

Typical sequence may be:

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

**Step 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 00h, protocol information of ATQB = 33 81 81h. It is an ISO 14443-4 Type B tag.

**Step 2:** Send an APDU, Get Challenge.

<< 00 84 00 00 08h

>> 1A F7 F3 1B CD 2B A9 58 [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:** ISO 7816-4 APDU

To read 8 bytes from an ISO 14443-4 Type B PICC (ST19XR08E)

APDU = { 80 B2 80 00 08h }

Class = 80h; INS = B2h; P1 = 80h; P2 = 00h;

Lc = None; Data In = None; Le = 08h

Answer: 00 01 02 03 04 05 06 07 [$90 00h]

## 5.3. Peripherals Control

The reader's peripherals control commands are implemented by using *PC_to_RDR_Escape*.

***Note**: The driver will add the Class, INS and P1 automatically.*

### 5.3.1. Get firmware version

This command gets the reader's firmware version.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| Get Firmware Version | E0h | 00h | 00h | 18h | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | Number of Bytes to be Received | Firmware Version |

**Example**:

Response = E1 00 00 00 0F 41 43 52 31 32 38 31 55 5F 56 35 30 33 2E 31

Firmware Version (HEX) = 41 43 52 31 32 38 31 55 5F 56 35 30 33 2E 31

Firmware Version (ASCII) = "ACR1281U_V503.1"

### 5.3.2. LED Control

This command controls the LED output.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| LED Control | E0h | 00h | 00h | 29h | 01h | LED Status |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | LED Status |

Where:

**LED Status** (1 byte)

| LED Status | Description | Description |
|---|---|---|
| Bit 0 | Red LED | 1 = ON<br>0 = OFF |
| Bit 1 | Green LED | 1 = ON<br>0 = OFF |
| Bit 2 – 7 | RFU | RFU |

### 5.3.3. LED Status

This command checks the existing LED status.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| LED Status | E0h | 00h | 00h | 29h | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | LED Status |

Where:

**LED Status** (1 byte)

| LED Status | Description | Description |
|---|---|---|
| Bit 0 | Red LED | 1 = ON<br>0 = OFF |
| Bit 1 | Green LED | 1 = ON<br>0 = OFF |
| Bit 2 – 7 | RFU | RFU |

### 5.3.4. Buzzer Control

This command controls the buzzer output.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| Buzzer Control | E0h | 00h | 00h | 28h | 01h | Buzzer on Duration |

Where:

**Buzzer on Duration** (1 byte)

01 – FFh = Duration (unit: 10 ms)

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | 00h |

### 5.3.5. Set default LED and buzzer behaviors

This command sets the default behavior of the LEDs and buzzer.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| Set Default LED and Buzzer Behaviors | E0h | 00h | 00h | 21h | 01h | Default Behaviors |

Where:

**Default Behaviors** (1 Byte)

Default value = FBh.

| LED Status | Description | Description |
|---|---|---|
| Bit 0 | ICC Activation Status LED | To show the activations status of the ICC interface.<br>1 = Enable<br>0 = Disable |
| Bit 1 | PICC Polling Status LED | To show the PICC polling status.<br>1 = Enable<br>0 = Disable |
| Bit 2 | RFU | RFU |
| Bit 3 | RFU | RFU |
| Bit 4 | Card Insertion and Removal Events Buzzer | To make a beep whenever a card insertion or removal event is detected (for both ICC and PICC).<br>1 = Enable<br>0 = Disable |
| Bit 5 | Contactless Chip Reset Indication Buzzer | To make a beep when the contactless chip is reset.<br>1 = Enable<br>0 = Disable |
| Bit 6 | Exclusive Mode Status Buzzer. Either ICC or PICC Interface can be activated | To make a beep when the exclusive mode is activated.<br>1 = Enable<br>0 = Disable |
| Bit 7 | Card Operation Blinking LED | To make the LED blink whenever the card (PICC or ICC) is being accessed. |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | Default Behaviors |

### 5.3.6. Read default LED and buzzer behaviors

This command reads the current default behaviors of LEDs and buzzer.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| Read Default LED and Buzzer Behaviors | E0h | 00h | 00h | 21h | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | Default Behaviors |

Where:

**Default Behaviors** (1 byte)

Default value = FBh.

| LED Status | Description | Description |
|---|---|---|
| Bit 0 | ICC Activation Status LED | To show the activations status of the ICC interface.<br>1 = Enable<br>0 = Disable |
| Bit 1 | PICC Polling Status LED | To show the PICC polling status.<br>1 = Enable<br>0 = Disable |
| Bit 2 | RFU | RFU |
| Bit 3 | RFU | RFU |
| Bit 4 | Card Insertion and Removal Events Buzzer | To make a beep whenever a card insertion or removal event is detected (for both ICC and PICC).<br>1 = Enable<br>0 = Disable |
| Bit 5 | Contactless Chip Reset Indication Buzzer | To make a beep when the contactless chip is reset.<br>1 = Enable<br>0 = Disable |
| Bit 6 | Exclusive Mode Status Buzzer. Either ICC or PICC Interface can be activated | To make a beep when the exclusive mode is activated.<br>1 = Enable<br>0 = Disable |
| Bit 7 | Card Operation Blinking LED | To make the LED blink whenever the card (PICC or ICC) is being accessed. |

### 5.3.7. Initialize cards insertion counter

This command initializes the cards insertion/detection counter.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In | | | |
|---|---|---|---|---|---|---|---|---|---|
| Initialize Cards Insertion Counter | E0h | 00h | 00h | 09h | 04h | ICC Cnt (LSB) | ICC Cnt (MSB) | PICC Cnt (LSB) | PICC Cnt (MSB) |

Where:

| | |
|---|---|
| **ICC Cnt (LSB)** | (1 byte)<br>ICC Insertion Counter (LSB) |
| **ICC Cnt (MSB)** | (1 byte)<br>ICC Insertion Counter (MSB) |
| **PICC Cnt (LSB)** | (1 byte)<br>PICC Insertion Counter (LSB) |
| **PICC Cnt (MSB)** | (1 byte)<br>PICC Insertion Counter (MSB) |

Response

| Response | Class | INS | P1 | P2 | Le |
|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 00h |

### 5.3.8. Read cards insertion counter

This command checks the cards insertion/detection counter value.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| Read Cards Insertion Counter | E0h | 00h | 00h | 09h | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out | | | |
|---|---|---|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 04h | ICC Cnt (LSB) | ICC Cnt (MSB) | PICC Cnt (LSB) | PICC Cnt (MSB) |

Where:

**ICC Cnt (LSB)** (1 byte)
ICC Insertion Counter (LSB)

**ICC Cnt (MSB)** (1 byte)
ICC Insertion Counter (MSB)

**PICC Cnt (LSB)** (1 byte)
PICC Insertion Counter (LSB)

**PICC Cnt (MSB)** (1 byte)
PICC Insertion Counter (MSB)

### 5.3.9. Update cards insertion counter

This command updates the cards insertion/detection counter value.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| Update Cards Insertion Counter | E0h | 00h | 00h | 0Ah | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out | | | |
|---|---|---|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 04h | ICC Cnt (LSB) | ICC Cnt (MSB) | PICC Cnt (LSB) | PICC Cnt (MSB) |

Where:

| | |
|---|---|
| **ICC Cnt (LSB)** | (1 byte)<br>ICC Insertion Counter (LSB) |
| **ICC Cnt (MSB)** | (1 byte)<br>ICC Insertion Counter (MSB) |
| **PICC Cnt (LSB)** | (1 byte)<br>PICC Insertion Counter (LSB) |
| **PICC Cnt (MSB)** | (1 byte)<br>PICC Insertion Counter (MSB) |

### 5.3.10. Set automatic PICC polling

This command sets the reader's polling mode.

Whenever the reader is connected to the computer, the PICC polling function will start the PICC scanning to determine if a PICC is placed on/removed from the built-in antenna.

You can send a command to disable the PICC polling function by sending a command through the PC/SC Escape Command interface. To meet the energy saving requirement, special modes are provided for turning off the antenna field whenever the PICC is inactive, or no PICC is found. The reader will consume less current in power saving mode.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| Set Automatic PICC Polling | E0h | 00h | 00h | 23h | 01h | Polling Setting |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | Polling Setting |

Where:

**Polling Setting** (1 byte)

Default value = 8Fh

| Polling Setting | Description | Description |
|---|---|---|
| Bit 0 | Auto PICC Polling | 1 = Enable<br>0 = Disable |
| Bit 1 | Turn off Antenna Field if no PICC found | 1 = Enable<br>0 = Disable |
| Bit 2 | Turn off Antenna Field if the PICC is inactive | 1 = Enable<br>0 = Disable |
| Bit 3 | RFU | RFU |
| Bit 5 – 4 | PICC Polling Interval for PICC | Bit 5 – Bit 4:<br>0 – 0 = 250 ms<br>0 – 1 = 500 ms<br>1 – 0 = 1000 ms<br>1 – 1 = 2500 ms |
| Bit 6 | RFU | RFU |
| Bit 7 | Enforce ISO 14443A Part 4 | 1 = Enable<br>0 = Disable |

***Notes**:*

1. *It is recommended to enable the option "Turn off Antenna Field is the PICC is inactive," so that the "Inactive PICC" will not be exposed to the field all the time to prevent the PICC from "warming up."*

2. *The longer the PICC Poll Interval, the more efficient it is for energy saving. However, the response time of PICC Polling will become longer. The Idle Current Consumption in Power Saving Mode is about 60 mA, while the Idle Current Consumption in Non-Power Saving mode is about 130 mA. Idle Current Consumption = PICC is not activated.*
3. *The reader will activate the ISO 14443A-4 mode of the "ISO 14443A-4 compliant PICC" automatically. Type B PICC will not be affected by this option.*
4. *The JCOP30 card comes with two modes: ISO 14443A-3 (MIFARE 1K) and ISO 14443A-4 modes. The application has to decide which mode should be selected once the PICC is activated.*

### 5.3.11. Read automatic PICC polling

This command checks the current automatic PICC polling.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| Read Automatic PICC Polling | E0h | 00h | 00h | 23h | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | Polling Setting |

Where:

**Polling Setting** (1 byte)

Default value = 8Fh

| Polling Setting | Description | Description |
|---|---|---|
| Bit 0 | Auto PICC Polling | 1 = Enable<br>0 = Disable |
| Bit 1 | Turn off Antenna Field if no PICC found | 1 = Enable<br>0 = Disable |
| Bit 2 | Turn off Antenna Field if the PICC is inactive | 1 = Enable<br>0 = Disable |
| Bit 3 | RFU | RFU |
| Bit 5 – 4 | PICC Polling Interval for PICC | Bit 5 – Bit 4:<br>0 – 0 = 250 ms<br>0 – 1 = 500 ms<br>1 – 0 = 1000 ms<br>1 – 1 = 2500 ms |
| Bit 6 | RFU | RFU |
| Bit 7 | Enforce ISO 14443A Part 4 | 1 = Enable<br>0 = Disable |

### 5.3.12. Manual PICC polling

This command determines if any PICC is within the detection range of the reader. This command can be used if the automatic PICC polling function is disabled.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| Manual PICC Polling | E0h | 00h | 00h | 22h | 01h | 0Ah |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | Status |

Where:

| | |
|---|---|
| **Status** | (1 byte) |
| | 00h = PICC is detected |
| | FFh = No PICC is detected |

### 5.3.13. Set PICC operating parameter

The command sets the PICC operating parameter.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| Set the PICC Operating Parameter | E0h | 00h | 00h | 20h | 01h | Operating Parameter |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | Operating Parameter |

Where:

**Operating Parameter** (1 byte)

Default value = 03h

| Operating Parameter | Parameter | Description | Option |
|---|---|---|---|
| Bit 0 | ISO 14443 Type A | The tag types to be detected during PICC Polling | 1 = Detect<br>0 = Skip |
| Bit 1 | ISO 14443 Type B | | 1 = Detect<br>0 = Skip |
| Bit 2 – 7 | RFU | RFU | RFU |

### 5.3.14. Read PICC operating parameter

This command checks current PICC operating parameter.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| Read the PICC Operating Parameter | E0h | 00h | 00h | 20h | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out |
|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 01h | Operating Parameter |

Where:

**Operating Parameter** (1 byte)

| Operating Parameter | Parameter | Description | Option |
|---|---|---|---|
| Bit 0 | ISO 14443 Type A | The tag types to be detected during PICC Polling | 1 = Detect<br>0 = Skip |
| Bit 1 | ISO 14443 Type B | | 1 = Detect<br>0 = Skip |
| Bit 2 – 7 | RFU | RFU | RFU |

### 5.3.15. Set exclusive mode

This command sets the reader in to/out from exclusive mode.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| Set Exclusive Mode | E0h | 00h | 00h | 2Bh | 01h | New Mode Configuration |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out | |
|---|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 02h | Mode Configuration | Current Mode Configuration |

Where:

**Exclusive Mode** (1 byte)

00h = Share Mode: ICC and PICC interfaces can work at the same time.

01h = Exclusive Mode: PICC is disabled when Auto Polling and Antenna Power Off when ICC is inserted (Default).

### 5.3.16. Read exclusive mode

This command checks the current exclusive mode setting.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| Read Exclusive Mode | E0h | 00h | 00h | 2Bh | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out | |
|---|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 02h | Mode Configuration | Current Mode Configuration |

Where:

**Exclusive Mode** (1 byte)

00h = Share Mode: ICC and PICC interfaces can work at the same time.

01h = Exclusive Mode: PICC is disabled when Auto Polling and Antenna Power Off when ICC is inserted (Default).

### 5.3.17. Set auto PPS

Whenever a PICC is recognized, the reader will try to change the communication speed between the PCD and PICC defined by the maximum connection speed. If the card does not support the proposed connection speed, the reader will try to connect the card with a slower speed setting.

Command

| Command | Class | INS | P1 | P2 | Lc | Data In |
|---|---|---|---|---|---|---|
| Set Auto PPS | E0h | 00h | 00h | 24h | 01h | Max Speed |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out | |
|---|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 02h | Max Speed | Current Speed |

Where:

**Max Speed** Maximum Speed (1 byte)

**Current Speed** Current Speed (1 byte)

00h = 106 Kbps; default setting, equal to No Auto PPS

01h = 212 Kbps

02h = 424 Kbps

03h = 848 Kbps

***Notes:***

1. *Normally, the application should know the maximum connection speed of the PICCs being used. The environment also affects the maximum achievable speed. The reader just uses the proposed communication speed to talk with the PICC. The PICC will become inaccessible if the PICC or environment does not meet the requirement of the proposed communication speed.*
2. *If the higher speed setting affects the performance of the reader, please switch back to a lower speed setting.*

### 5.3.18. Read auto PPS

This command checks the current auto PPS setting.

Command

| Command | Class | INS | P1 | P2 | Lc |
|---|---|---|---|---|---|
| Read Auto PPS | E0h | 00h | 00h | 24h | 00h |

Response

| Response | Class | INS | P1 | P2 | Le | Data Out | |
|---|---|---|---|---|---|---|---|
| Result | E1h | 00h | 00h | 00h | 02h | Max Speed | Current Speed |

Where:

**Max Speed** Maximum Speed (1 byte)

**Current Speed** Current Speed (1 byte)

00h = 106 Kbps; default setting, equal to No Auto PPS

01h = 212 Kbps

02h = 424 Kbps

03h = 848 Kbps

# Appendix A. Basic program flow for contactless applications

Step 0: Start the application. The reader will do the PICC polling and scan for tags continuously. Once the tag is found and detected, the corresponding ATR will be sent to the computer.

Step 1: Connect the “ACR1281U PICC Interface” with T=1 protocol.

Step 2: Access the PICC by exchanging APDUs.

..

Step N: Disconnect the “ACR1281U PICC Interface”. Shut down the application.

# Appendix B. Accessing MIFARE DESFire tags (ISO 14443-4)

MIFARE® DESFire supports ISO 7816-4 APDU Wrapping and Native modes. Once the DESFire tag is activated, the first APDU sent to the DESFire tag will determine the “Command Mode.” If the first APDU is “Native Mode,” the rest of the APDUs must be in “Native Mode” format. Similarly, if the first APDU is “ISO 7816-4 APDU Wrapping Mode,” the rest of the APDUs must be in “ISO 7816-4 APDU Wrapping Mode” format.

**Example 1**: MIFARE DESFire ISO 7816-4 APDU Wrapping.

To read 8 bytes random number from an ISO 14443-4 Type A PICC (DESFire):

APDU = {90 0A 00 00 01 00 00h}

Class = 90h; INS = 0Ah (DESFire Instruction); P1 = 00h; P2 = 00h

Lc = 01h; Data In = 00h; Le = 00h (Le = 00h for maximum length)

Answer: 7B 18 92 9D 9A 25 05 21h [$91AFh]

***Note:** Status Code {91 AFh} is defined in MIFARE DESFire specification. Please refer to MIFARE DESFire specification for more details.*

**Example 2**: MIFARE DESFire Frame Level Chaining (ISO 7816 wrapping mode)

In this example, the application has to do the “Frame Level Chaining”.

To get the version of the DESFire card:

Step 1: Send an APDU {90 60 00 00 00h} to get the first frame. INS=60h

Answer: 04 01 01 00 02 18 05 91 AFh [$91AFh]

Step 2: Send an APDU {90 AF 00 00 00h} to get the second frame. INS=AFh
Answer: 04 01 01 00 06 18 05 91 AFh [$91AFh]

Step 3: Send an APDU {90 AF 00 00 00h} to get the last frame. INS=AFh

Answer: 04 52 5A 19 B2 1B 80 8E 36 54 4D 40 26 04 91 00h [$9100h]

**Example 3**: MIFARE DESFire Native Command.

You can send Native DESFire Commands to the reader without ISO 7816 wrapping if we find that the Native DESFire Commands are easier to handle.

To read 8 bytes random number from an ISO 14443-4 Type A PICC (DESFire):

APDU = {0A 00h}

Answer: AF 25 9C 65 0C 87 65 1D D7h [$1DD7h]

In which, the first byte “AF” is the status code returned by the MIFARE DESFire card.

The Data inside the blanket [$1DD7h] can simply be ignored by the application.

**Example 4:** MIFARE DESFire Frame Level Chaining (Native Mode)

In this example, the application has to do the “Frame Level Chaining”.

To get the version of the DESFire card:

Step 1: Send an APDU {60h} to get the first frame. INS=60h

Answer: AF 04 01 01 00 02 18 05h [$1805h]

Step 2: Send an APDU {AFh} to get the second frame. INS=AFh
Answer: AF 04 01 01 00 06 18 05h [$1805h]

Step 3: Send an APDU {AFh} to get the last frame. INS=AFh

Answer: 00 04 52 5A 19 B2 1B 80 8E 36 54 4D 40 26 04h [$2604h]

***Note:*** *In MIFARE DESFire Native Mode, the status code [90 00h] will not be added to the response if the response length is greater than 1. If the response length is less than 2, the status code [90 00h] will be added in order to meet the requirement of PC/SC. The minimum response length is 2.*

# Appendix C. Extended APDU Example

Card: ACOS7 (supports Extended APDU, echo response)

Write CMD: 80 D2 00 00 XX XX XXh

CLA = 80h

INS = D2h

P1 = 00h

P2 = 00h

Data Len = XX XX XXh

**Example 1:** APDU length = 263 bytes

**APDU Command:**

```
80D2000000010000010203040506070809 0A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
```

**Response:**

```
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F202122232425262
2728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F404142434445464748494A4B4C4
D4E4F505152535455565758595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F70717273
7475767778797A7B7C7D7E7F808182838485868788898A8B8C8D8E8F909192939495969798999A
9B9C9D9E9FA0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B8B9BABBBCBDB
EBFC0C1C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDFE
0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF9000h
```

**Example 2:** APDU length = 775 bytes

**APDU Command:**

```
80D20000000300000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
202122232425262728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F40414243444546
4748494A4B4C4D4E4F505152535455565758595A5B5C5D5E5F606162636465666768696A6B6C6
D6E6F707172737475767778797A7B7C7D7E7F808182838485868788898A8B8C8D8E8F90919293
9495969798999A9B9C9D9E9FA0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B
8B9BABBBCBDBEBFC0C1C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9D
ADBDCDDDEDFE0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFD
FEFF000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F2021222324
25262728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F404142434445464748494A4B
4C4D4E4F505152535455565758595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F70717
2737475767778797A7B7C7D7E7F808182838485868788898A8B8C8D8E8F909192939495969798 9
99A9B9C9D9E9FA0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B8B9BABBBC
BDBEBFC0C1C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DADBDCDDDE
DFE0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF0001020
30405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F202122232425262728292
A2B2C2D2E2F303132333435363738393A3B3C3D3E3F404142434445464748494A4B4C4D4E4F50
5152535455565758595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F7071727374757677
```

```
78797A7B7C7D7E7F808182838485868788898A8B8C8D8E8F909192939495969798999A9B9C9D9
E9FA0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B8B9BABBBCBDBEBFC0C1
C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDFE0E1E2E3
E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFFh
```

**Response:**

```
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20212223242526
2728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F404142434445464748494A4B4C4
D4E4F505152535455565758595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F70717273
7475767778797A7B7C7D7E7F808182838485868788898A8B8C8D8E8F909192939495969798999A
9B9C9D9E9FA0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B8B9BABBBCBDB
EBFC0C1C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDFE
0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF00010203040
5060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F202122232425262728292A2B2
C2D2E2F303132333435363738393A3B3C3D3E3F404142434445464748494A4B4C4D4E4F505152
535455565758595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F707172737475767778 79
7A7B7C7D7E7F808182838485868788898A8B8C8D8E8F909192939495969798999A9B9C9D9E9FA
0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B8B9BABBBCBDBEBFC0C1C2C3
C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDFE0E1E2E3E4E5
E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF000102030405060708090A
0B0C0D0E0F101112131415161718191A1B1C1D1E1F202122232425262728292A2B2C2D2E2F303
132333435363738393A3B3C3D3E3F404142434445464748494A4B4C4D4E4F50515253545556575
8595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F707172737475767778797A7B7C7D7E
7F808182838485868788898A8B8C8D8E8F909192939495969798999A9B9C9D9E9FA0A1A2A3A4A
5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B8B9BABBBCBDBEBFC0C1C2C3C4C5C6C7C
8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDFE0E1E2E3E4E5E6E7E8E9EA
EBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF9000h
```

# Appendix D. Escape Command Example

**Example:** Get firmware version (using PCSCDirectCommand.exe).

Step 1: Plug in the ACR1281 Reader to the computer.

Step 2: Open the PCSCDirectCommand.exe.

Step 3: Connect the reader in Direct mode. The ATR will be displayed (if a card is present) or “No ATR retrieved (ATRLen = 0)” will be displayed (if no card).

Step 4: Enter Command: “2079”

Enter Data: “18 00” (APDU for Get Firmware Version)

Press Enter to send to reader, and then check the response.

***Note:** PCSCDirectCommand.exe is not available in the Software Development Kit (SDK). Please contact ACS for more information.*

# Appendix E. Supported Card Types

The following table summarizes the card type returned by GET_READER_INFORMATION correspond with the respective card type.

| Card Type Code | Card Type |
|---|---|
| 00h | Auto-select T=0 or T=1 communication protocol |
| 01h | I2C memory card (1k, 2k, 4k, 8k and 16k bits) |
| 02h | I2C memory card (32k, 64k, 128k, 256k, 512k and 1024k bits) |
| 03h | RFU |
| 04h | RFU |
| 05h | Infineon SLE4418 and SLE4428 |
| 06h | Infineon SLE4432 and SLE4442 |
| 07h | Infineon SLE4406, SLE4436 and SLE5536 |
| 08h | Infineon SLE4404 |
| 09h | RFU |

# Appendix F. ACR128 Compatibility

Below is the list of ACR128 functions that are implemented differently or not supported by ACR1281U-C1.

| Functions | ACR128 | ACR1281U-C1 |
|---|---|---|
| 1. Change the default FWI and Transmit Frame Size of the activated PICC. | 1F 03 [Data: 3 bytes] | Not supported. |
| 2. Transceiver Setting | 20 04 06 [Data: 3 bytes] | Not supported. |
| 3. PICC Setting | 2A 0C [Data: 12 bytes] | Not supported. |
| 4. PICC T=CL Data Exchange Error Handling | 2C 02 [Data:1 byte] | Not supported. |
| 5. Read Register | 19 01 [Reg. No.] | Not supported. |
| 6. Update Register | 1A 02 [Reg. No.] [Value] | Not supported. |
| 7. PICC Polling for Specific Types | 20 02 [Data: 1 byte] FF | 20 01 [Data: 1 byte] |
| 8. Buzzer Control | 28 01 [Duration]<br>Duration:<br>00 = Turn Off<br>01 – FE = Duration x 10 ms<br>FF = Turn On | 28 01 [Duration]<br>Duration:<br>01 – FF = Duration × 10 ms |

| Functions | ACR128 | ACR1281U-C1 |
|---|---|---|
| 9. Set/Read Default LED and Buzzer Behaviors | Set: 21 01 [Data: 1 byte]<br>Read: 21 00<br>Data:<br>Bit 0 = ICC Activation Status<br>Bit 1 = PICC Polling Status LED<br>Bit 2 = PICC Activation Status Buzzer<br>Bit 3 = PICC PPS Status Buzzer<br>Bit 4 = Card Insertion and Removal Events Buzzer<br>Bit 5 = Contactless Chip Reset Indication Buzzer<br>Bit 6 = Exclusive Mode Status Buzzer<br>Bit 7 = Card Operation Blinking LED | Set: 21 01 [Data: 1 byte]<br>Read: 21 00<br>Data:<br>Bit 0 = ICC Activation Status<br>Bit 1 = PICC Polling Status LED<br>Bit 2 = RFU<br>Bit 3 = RFU<br>Bit 4 = Card Insertion and Removal Events Buzzer<br>Bit 5 = Contactless Chip Reset Indication Buzzer<br>Bit 6 = Exclusive Mode Status Buzzer<br>Bit 7 = Card Operation Blinking LED |
| 10. Set/Read Automatic PICC Polling | Set: 23 01 [Data: 1 byte]<br>Read: 23 00<br>Data:<br>Bit 0 = Auto PICC Polling<br>Bit 1 = Turn off Antenna Field if no PICC is found<br>Bit 2 = Turn off Antenna Field if the PICC is inactive<br>Bit 3 = Activate the PICC when detected<br>Bit 4..5 = PICC Poll Interval for PICC<br>Bit 6 = Test Mode<br>Bit 7 = Enforce ISO 14443A Part 4 | Set: 23 01 [Data: 1 byte]<br>Read: 23 00<br>Data:<br>Bit 0 = Auto PICC Polling<br>Bit 1 = Turn off Antenna Field if no PICC is found<br>Bit 2 = Turn off Antenna Field if the PICC is inactive<br>Bit 3 = RFU<br>Bit 4..5 = PICC Poll Interval for PICC<br>Bit 6 = RFU<br>Bit 7 = Enforce ISO 14443A Part 4 |