



**Advanced Card Systems Ltd.**  
Card & Reader Technologies

# ACR122U USB NFC Reader



Application Programming Interface V2.04



## Table of Contents

|                    |   |           |
|--------------------|---|-----------|
| <b>1.0.</b>        | <b>Introduction .....</b>   | <b>4</b>  |
| 1.1.               | Features .....  | 4         |
| 1.2.               | USB Interface .....   | 5         |
| <b>2.0.</b>        | <b>Implementation .....</b>   | <b>6</b>  |
| 2.1.               | Communication Flow Chart of ACR122U .....                                   | 6         |
| 2.2.               | Smart Card Reader Interface Overview .....                                  | 7         |
| <b>3.0.</b>        | <b>PICC Interface Description .....</b>                                     | <b>8</b>  |
| 3.1.               | ATR Generation .....  | 8         |
| 3.1.1.             | ATR format for ISO 14443 Part 3 PICCs .....                                 | 8         |
| 3.1.2.             | ATR format for ISO 14443 Part 4 PICCs .....                                 | 9         |
| <b>4.0.</b>        | <b>PICC Commands for General Purposes .....</b>                             | <b>11</b> |
| 4.1.               | Get Data .....  | 11        |
| <b>5.0.</b>        | <b>PICC Commands (T=CL Emulation) for MIFARE Classic Memory Cards .....</b> | <b>12</b> |
| 5.1.               | Load Authentication Keys .....  | 12        |
| 5.2.               | Authentication .....  | 13        |
| 5.3.               | Read Binary Blocks .....  | 16        |
| 5.4.               | Update Binary Blocks .....  | 17        |
| 5.5.               | Value Block Related Commands .....  | 18        |
| 5.5.1.             | Value Block Operation .....   | 18        |
| 5.5.2.             | Read Value Block .....  | 19        |
| 5.5.3.             | Restore Value Block .....   | 20        |
| <b>6.0.</b>        | <b>Pseudo-APDU Commands .....</b>   | <b>21</b> |
| 6.1.               | Direct Transmit .....   | 21        |
| 6.2.               | Bi-color LED and Buzzer Control .....                                       | 22        |
| 6.3.               | Get firmware version of the reader .....                                    | 24        |
| 6.4.               | Get the PICC operating parameter .....                                      | 25        |
| 6.5.               | Set the PICC operating parameter .....                                      | 26        |
| 6.6.               | Set Timeout Parameter .....   | 27        |
| 6.7.               | Set buzzer output during card detection .....                               | 28        |
| <b>7.0.</b>        | <b>Basic Program Flow for Contactless Applications .....</b>                | <b>29</b> |
| 7.1.               | How to access PC/SC-compliant tags (ISO 14443-4)? .....                     | 31        |
| 7.2.               | How to access MIFARE DESFire tags (ISO 14443-4)? .....                      | 32        |
| 7.3.               | How to access FeliCa tags (ISO 18092)? .....                                | 34        |
| 7.4.               | How to access NFC Forum Type 1 Tags (ISO 18092)? .....                      | 35        |
| 7.5.               | Get the current setting of the contactless interface .....                  | 37        |
| <b>Appendix A.</b> | <b>ACR122U PC/SC Escape Command .....</b>                                   | <b>38</b> |
| <b>Appendix B.</b> | <b>APDU Command and Response Flow for ISO 14443-Compliant Tags ..</b>       | <b>41</b> |
| <b>Appendix C.</b> | <b>APDU command and response flow for ISO 18092-compliant tags .....</b>    | <b>42</b> |
| <b>Appendix D.</b> | <b>Error Codes .....</b>  | <b>43</b> |
| <b>Appendix E.</b> | <b>Sample codes for setting the LED .....</b>                               | <b>45</b> |

## List of Figures

|                   |   |   |
|-------------------|---|---|
| <b>Figure 1 :</b> | Communication Flow Chart of ACR122U .....               | 6 |
| <b>Figure 2 :</b> | Smart Card Reader Interface on the Device Manager ..... | 7 |



**Figure 3** : Basic Program Flow for Contactless Applications ..... 29  
**Figure 4** : Topaz Memory Map ..... 36

## List of Tables

**Table 1** : USB Interface ..... 5  
**Table 2** : ATR format for ISO 14443 Part 3 PICCs ..... 8  
**Table 3** : ATR format for ISO 14443 Part 4 PICCs ..... 9  
**Table 4** : MIFARE 1K Memory Map ..... 14  
**Table 5** : MIFARE Classic 4K Memory Map ..... 14  
**Table 6** : MIFARE Ultralight Memory Map ..... 15  
**Table 7** : Error Codes ..... 44





## 1.2. USB Interface

The ACR122U is connected to a computer through USB as specified in the USB Specification 1.1. The ACR122U is working in full-speed mode, i.e. 12 Mbps.

| Pin | Signal           | Function  |
|-----|------------------|---|
| 1   | V <sub>BUS</sub> | +5 V power supply for the reader (Max. 200 mA, Normal 100 mA) |
| 2   | D-               | Differential signal transmits data between ACR122U and PC     |
| 3   | D+               | Differential signal transmits data between ACR122U and PC     |
| 4   | GND              | Reference voltage level for power supply                      |

**Table 1:** USB Interface



## 2.2. Smart Card Reader Interface Overview

Go to the **Device Manager** to see the “ACR122U PICC Interface.” The standard Microsoft USB CCID Driver should be used.

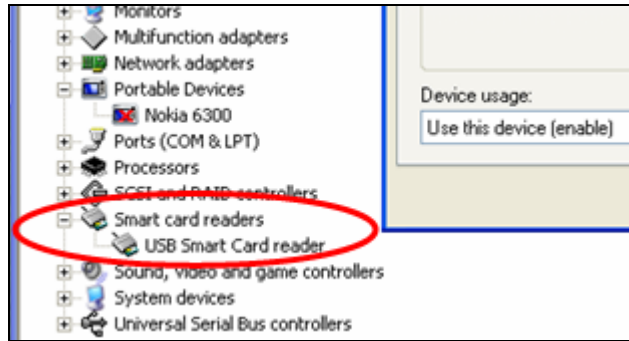


Figure 2: Smart Card Reader Interface on the Device Manager

### 3.0. PICC Interface Description

#### 3.1. ATR Generation

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

##### 3.1.1. ATR format for ISO 14443 Part 3 PICCs

| Byte           | Value (Hex)  | Designation    | Description   |
|----------------|--------------|----------------|---|
| 0              | 3Bh          | Initial Header | -   |
| 1              | 8Nh          | 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              | 80h          | TD1            | Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following.<br>Lower nibble 0 means T = 0  |
| 3              | 01h          | TD2            | Higher nibble 0 means no TA3, TB3, TC3, TD3 following.<br>Lower nibble 1 means T = 1  |
| 4<br>To<br>3+N | 80h          | T1             | Category indicator byte, 80 means A status indicator may be present in an optional COMPACT-TLV data object                                      |
|                | 4Fh          | Tk             | Application identifier Presence Indicator   |
|                | 0Ch          |                | Length  |
|                | RID          |                | Registered Application Provider Identifier (RID) # A0 00 00 03 06h  |
|                | SS           |                | Byte for standard   |
|                | C0 .. C1h    |                | Bytes for card name   |
|                | 00 00 00 00h |                | RFU   |
| 4+N            | UUh          | TCK            | Exclusive-oring of all the bytes T0 to Tk   |

**Table 2:** ATR format for ISO 14443 Part 3 PICCs

**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 00<br>00 03<br>06h | 03h      | 00 01h    | 00 00<br>00 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 Classic® 1K)

Where, Card Name (C0 .. C1)

- 00 01h: MIFARE Classic 1K
- 00 02h: MIFARE Classic 4K
- 00 03h: MIFARE® Ultralight®
- 00 26h: MIFARE Mini

- F0 04h: Topaz and Jewel
- F0 11h: FeliCa 212K
- F0 12h: FeliCa 424K
- FFh [SAK]: Undefined

### 3.1.2. ATR format for ISO 14443 Part 4 PICCs

| Byte             | Value (Hex)      | Designation    | Description  |
|------------------|------------------|----------------|--|
| 0                | 3Bh              | Initial Header | -  |
| 1                | 8Nh              | 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                | 80h              | TD1            | Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following.<br>Lower nibble 0 means T = 0   |
| 3                | 01h              | TD2            | Higher nibble 0 means no TA3, TB3, TC3, TD3 following.<br>Lower nibble 1 means T = 1   |
| 4<br>to<br>3 + N | XXh              | T1             | Historical Bytes:<br><br>ISO 14443A:<br>The historical bytes from ATS response. Refer to the ISO14443-4 specification.<br><br>ISO 14443B:<br>The higher layer response from the ATTRIB response (ATQB). Refer to the ISO14443-3 specification. |
|                  | XXh<br>XX<br>XXh | Tk             |  |
| 4+N              | UUh              | TCK            | Exclusive-oring of all the bytes T0 to Tk  |

**Table 3:** ATR format for ISO 14443 Part 4 PICCs



We take for example, an ATR for DESFire, which is:  
DESFire (ATR) = 3B 86 80 01 06 75 77 81 02 80 00h

| ATR            |     |     |     |     |                 |     |
|----------------|-----|-----|-----|-----|-----------------|-----|
| Initial Header | T0  | TD1 | TD2 | ATS |                 | TCK |
|                |     |     |     | T1  | Tk              |     |
| 3Bh            | 86h | 80h | 01h | 06h | 75 77 81 02 80h | 00h |

This ATR has 6 bytes of ATS, which is: [06 75 77 81 02 80h]

**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 ISO14443A-3 or ISO14443B-3/4 PICCs.

Another example would be the ATR for ST19XRC8E, which is:

ST19XRC8E (ATR) = 3B 8C 80 01 50 12 23 45 56 12 53 54 4E 33 81 C3 55h

| ATR            |     |     |     |      |                                   |     |
|----------------|-----|-----|-----|------|-----------------------------------|-----|
| Initial Header | T0  | TD1 | TD2 | ATQB |                                   | TCK |
|                |     |     |     | T1   | Tk                                |     |
| 3Bh            | 86h | 80h | 01h | 50h  | 12 23 45 56 12 53 54 4E 33 81 C3h | 55h |

Since this card follows ISO 14443 Type B, the response would be ATQB which is 50 12 23 45 56 12 53 54 4E 33 81 C3h is 12 bytes long with no CRC-B

**Note:** You can refer to the ISO7816, ISO14443 and PC/SC standards for more details.



## 4.0. PICC Commands for General Purposes

### 4.1. Get Data

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

Get UID APDU Format (5 bytes)

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

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

| Response | Data Out     |   |   |              |     |     |
|----------|--------------|---|---|--------------|-----|-----|
| Result   | UID<br>(LSB) | - | - | UID<br>(MSB) | SW1 | SW2 |

Get ATS of a ISO 14443 A card (ATS + 2 bytes) if P1 = 01h

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

Response Codes

| Results | SW1 SW2 | Meaning                               |
|---------|---------|---------------------------------------|
| Success | 90 00h  | The operation completed successfully. |
| Error   | 63 00h  | The operation failed.                 |
| Error   | 6A 81h  | Function not supported.               |

**Example:**

- To get the serial number of the connected PICC.  
`UINT8 GET_UID[5]={FFh, CAh, 00h, 00h, 04h};`
- To get the ATS of the connected ISO 14443 A PICC.  
`UINT8 GET_ATS[5]={FFh, CAh, 01h, 00h, 04h};`



## 5.0. PICC Commands (T=CL Emulation) for MIFARE Classic Memory Cards

### 5.1. Load Authentication Keys

This command loads the authentication keys into the reader. The authentication keys are used to authenticate the particular sector of the MIFARE Classic 1K/4K memory card. Volatile authentication key location is provided.

Load Authentication Keys APDU Format (11 bytes)

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

Where:

**Key Structure** 1 byte.

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

Other = Reserved.

**Key Number** 1 byte.

00h ~ 01h = Key Location. The keys will disappear once the reader is disconnected from the PC.

**Key** 6 bytes.

The key value loaded into the reader. e.g., {FF FF FF FF FF FFh}

Load Authentication Keys Response Format (2 Bytes)

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

Response Codes

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

**Example:**

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

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



## 5.2. Authentication

This command uses the keys stored in the reader to do authentication with the MIFARE 1K/4K card (PICC). Two types of authentication keys are used: TYPE\_A and TYPE\_B.

Load Authentication Keys APDU Format (6 bytes) [Obsolete]

| Command        | Class | INS | P1  | P2           | P3       | Data In    |
|----------------|-------|-----|-----|--------------|----------|------------|
| Authentication | FFh   | 88h | 00h | Block Number | Key Type | Key Number |

Load Authentication Keys APDU Format (10 bytes)

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

Authenticate Data Bytes (5 bytes)

| Byte1       | Byte 2 | Byte 3       | Byte 4   | Byte 5     |
|-------------|--------|--------------|----------|------------|
| Version 01h | 00h    | Block Number | Key Type | Key Number |

Where:

**Block Number** 1 byte. This is the memory block to be authenticated.

**Key Type** 1 byte

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

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

**Key Number** 1 byte

00h ~ 01h = Key Location.

**Note:** For MIFARE Classic 1K Card, it has totally 16 sectors and each sector consists of 4 consecutive blocks. E.g. Sector 00h consists of Blocks {00h, 01h, 02h and 03h}; Sector 01h consists of Blocks {04h, 05h, 06h and 07h}; the last sector 0F consists of Blocks {3Ch, 3Dh, 3Eh and 3Fh}.

Once the authentication is done successfully, there is no need to do the authentication again if the blocks to be accessed belong to the same sector. Please refer to the MIFARE Classic 1K/4K specification for more details.

Load Authentication Keys Response Format (2 bytes)

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

Response Codes

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



| Sectors<br>(Total 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 4:** MIFARE 1K Memory Map

| Sectors<br>(Total 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 8 sectors. Each sector consists of 16 consecutive blocks) | Data Blocks<br>(15 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 5:** MIFARE Classic 4K Memory Map



| 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 6:** MIFARE Ultralight Memory Map

Example:

- To authenticate the Block 04h with a {TYPE A, key number 00h}. For PC/SC V2.01, Obsolete.  
APDU = {FF 88 00 04 60 00h};
- To authenticate the Block 04h with a {TYPE A, key number 00h}. For PC/SC V2.07  
APDU = {FF 86 00 00 05 01 00 04 60 00h}

**Note:** MIFARE Ultralight does not need to do any authentication. The memory is free to access.



### 5.3. Read Binary Blocks

This command retrieves the data blocks from the PICC. The data block/trailer block must be authenticated first.

Read Binary APDU Format (5 bytes)

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

Where:

|                                |        |                           |
|--------------------------------|--------|---------------------------|
| <b>Block Number</b>            | 1 byte | The block to be accessed. |
| <b>Number of Bytes to Read</b> | 1 byte | Maximum 16 bytes.         |

Read Binary Block Response Format (N + 2 bytes)

| Response | Data Out     |     |     |
|----------|--------------|-----|-----|
| Result   | 0 <= N <= 16 | SW1 | SW2 |

Response Codes

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

**Example:**

- Read **16 bytes** from the binary block **04h** (MIFARE Classic 1K or 4K)  
APDU = {FF B0 00 **04** **10**}
- Read **4 bytes** from the binary Page **04h** (MIFARE Ultralight)  
APDU = {FF B0 00 **04** **04**}
- Read **16 bytes** starting from the binary Page **04h** (MIFARE Ultralight) (Pages 4, 5, 6 and 7 will be read)  
APDU = {FF B0 00 **04** **10**}

**Note:** Please add a 2-second delay when reading NDEF messages in MIFARE Classic 4K cards.





## 5.4. Update Binary Blocks

This command writes data blocks into the PICC. The data block/trailer block must be authenticated.

Update Binary APDU Format (4 or 16 + 5 bytes)

| Command              | Class | INS | P1  | P2           | Lc                        | Data In   |
|----------------------|-------|-----|-----|--------------|---------------------------|---|
| Update Binary Blocks | FFh   | D6h | 00h | Block Number | Number of Bytes to Update | Block Data<br>4 Bytes for MIFARE Ultralight or<br>16 Bytes for MIFARE 1K/4K |

Where:

|                                  |                      |  |
|----------------------------------|----------------------|--|
| <b>Block Number</b>              | 1 byte               | The starting block to be updated.                          |
| <b>Number of Bytes to Update</b> | 1 byte               | 16 bytes for MIFARE 1K/4K<br>4 bytes for MIFARE Ultralight |
| <b>Block Data</b>                | 4 bytes or 16 bytes. | The data to be written into the binary block/blocks.       |

Response Codes

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

**Example:**

- Update the binary block 04h of MIFARE Classic 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}
- Update the binary block 04h of MIFARE Ultralight with Data {00 01 02 03}  
APDU = {FF D6 00 04 04 00 01 02 03h}



## 5.5. Value Block Related Commands

The data block can be used as value block for implementing value-based applications.

### 5.5.1. Value Block Operation

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

Value Block Operation APDU Format (10 bytes)

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

Where:

**Block Number** 1 byte

The value block to be manipulated.

**VB\_OP** 1 byte

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

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

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

**VB\_Value** 4 bytes.

The value used for value manipulation. The value is a signed long integer (4 bytes).

**Example 1:** Decimal -4 = {FFh, FFh, FFh, FCh}

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

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

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

Value Block Operation Response Format (2 bytes)

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





### 5.5.3. Restore Value Block

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

Restore Value Block APDU Format (7 bytes)

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

Where:

- Source Block Number**      1 byte  
The value of the source value block will be copied to the target value block.
- Target Block Number**    1 byte  
The value block to be restored. The source and target value blocks must be in the same sector.

Restore Value Block Response Format (2 bytes)

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

Response Codes

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

#### Example:

- Store a value "1" into block 05h  
APDU = {FF D7 00 05 05 00 00 00 00 01h}  
Answer: 90 00h
- Read the value block 05h  
APDU = {FF B1 00 05 00h}  
Answer: 00 00 00 01 90 00h [9000h]
- Copy the value from value block 05h to value block 06h  
APDU = {FF D7 00 05 02 03 06h}  
Answer: 90 00h [9000h]
- Increment the value block 05h by "5"  
APDU = {FF D7 00 05 05 01 00 00 00 05h}  
Answer: 90 00h [9000h]



## 6.0. Pseudo-APDU Commands

The pseudo-APDU commands are used for the following:

- Exchanging data with non-PC/SC-compliant tags
- Retrieving and setting the reader parameters
- Pseudo-APDUs can be sent through the “ACR122U PICC Interface” if the tag is already connected
- Pseudo-APDUs can be sent using “Escape Command” if the tag is not yet presented

### 6.1. Direct Transmit

This is the payload to be sent to the tag or reader.

Direct Transmit Command Format (Length of the Payload + 5 bytes)

| Command         | Class | INS | P1  | P2  | Lc                      | Data In |
|-----------------|-------|-----|-----|-----|-------------------------|---------|
| Direct Transmit | FFh   | 00h | 00h | 00h | Number of Bytes to send | Payload |

Where:

- Lc**            1 byte.  
                  Number of bytes to send  
                  Maximum 255 bytes
- Data In**      Response

Direct Transmit Response Format

| Response        | Data Out      |
|-----------------|---------------|
| Direct Transmit | Response Data |



## 6.2. Bi-color LED and Buzzer Control

This command controls the states of the bi-color LED and Buzzer.

Bi-color LED and Buzzer Control Command Format (9 bytes)

| Command                         | Class | INS | P1  | P2                                  | Lc  | Data In (4 bytes)         |
|---------------------------------|-------|-----|-----|-------------------------------------|-----|---------------------------|
| Bi-Color and Buzzer LED Control | FFh   | 00h | 40h | LED State Control (Bit 7 --- Bit 0) | 04h | Blinking Duration Control |

Where:

**P2** LED State Control

Bi-Color LED and Buzzer Control Format (1 byte)

| CMD   | Item                              | Description                           |
|-------|-----------------------------------|---------------------------------------|
| Bit 0 | Final State: Red LED              | 1 = On; 0 = Off                       |
| Bit 1 | Final State: Green LED            | 1 = On; 0 = Off                       |
| Bit 2 | State Mask: Red LED               | 1 = Update the State<br>0 = No change |
| Bit 3 | State Mask: Green LED             | 1 = Update the State<br>0 = No change |
| Bit 4 | Initial Blinking State: Red LED   | 1 = On; 0 = Off                       |
| Bit 5 | Initial Blinking State: Green LED | 1 = On; 0 = Off                       |
| Bit 6 | Blinking Mask: Red LED            | 1 = Blink<br>0 = Not Blink            |
| Bit 7 | Blinking Mask: Green LED          | 1 = Blink<br>0 = Not Blink            |

**Data In** Blinking Duration Control

Bi-Color LED Blinking Duration Control Format (4 Bytes)

| Byte 0   | Byte 1  | Byte 2               | Byte 3         |
|--|---|----------------------|----------------|
| T1 Duration<br>Initial Blinking State<br>(Unit = 100 ms) | T2 Duration<br>Toggle Blinking State<br>(Unit = 100 ms) | Number of repetition | Link to Buzzer |

Where:

**Byte 3** Link to Buzzer. Control the buzzer state during the LED Blinking.

00h: The buzzer will not turn on

01h: The buzzer will turn on during the T1 Duration

02h: The buzzer will turn on during the T2 Duration

03h: The buzzer will turn on during the T1 and T2 Duration.

**Data Out** **SW1 SW2**. Status Code returned by the reader.



| Results | SW1 | SW2               | Meaning                               |
|---------|-----|-------------------|---------------------------------------|
| Success | 90h | Current LED State | The operation completed successfully. |
| Error   | 63h | 00h               | The operation failed.                 |

Current LED State (1 byte)

| Status     | Item              | Description     |
|------------|-------------------|-----------------|
| Bit 0      | Current Red LED   | 1 = On; 0 = Off |
| Bit 1      | Current Green LED | 1 = On; 0 = Off |
| Bits 2 – 7 | Reserved          |                 |

**Notes:**

1. LED Blinking will take effect only if the corresponding LED Blinking Mask is enabled and the number of repetition is greater than zero.
2. The term Initial Blinking State means that the LED of the chosen color will either be turned ON or OFF during the first blink in the duty cycle. For example, if the Initial Blinking State is turned ON for the Green LED and OFF for the Red LED, then the blinking will start with Green, followed by Red, and so on.
3. The change in LED State will take effect only if the corresponding LED State Mask is enabled.
4. If controlled at the same time, the LED State operation will be performed after the LED Blinking operation has been completed.
5. Under Blinking Duration Control, Both T1 and T2 duration parameters are used for controlling the duty cycle of LED blinking and Buzzer Turn-On duration. For example, if T1=1 and T2=1, the duty cycle = 50%. #Duty Cycle =  $T1 / (T1 + T2)$ .
6. To control the buzzer only, set the P2 “LED State Control” to zero.
7. To make the buzzer operating, the “number of repetition” must be greater than zero.
8. To control the LED only, set the parameter “Link to Buzzer” to zero.



### 6.3. Get firmware version of the reader

This command retrieves the firmware version of the reader.

Command Format (5 bytes)

| Command              | Class | INS | P1  | P2  | Le  |
|----------------------|-------|-----|-----|-----|-----|
| Get Firmware Version | FFh   | 00h | 48h | 00h | 00h |

Response Format (10 bytes)

| Response | Data Out         |
|----------|------------------|
| Result   | Firmware Version |

**Example:**

Response = 41 43 52 31 32 32 55 32 30 31h = ACR122U201 (ASCII)





## 6.4. Get the PICC operating parameter

This command retrieves the PICC operating parameter of the reader.

Command Format (5 bytes)

| Command                      | Class | INS | P1  | P2  | Le  |
|------------------------------|-------|-----|-----|-----|-----|
| Get PICC Operating Parameter | FFh   | 00h | 50h | 00h | 00h |

Response Format (2 bytes)

| Response | Data Out |                          |
|----------|----------|--------------------------|
| Result   | 90h      | PICC Operating Parameter |



## 6.5. Set the PICC operating parameter

This command sets the PICC operating parameter of the reader.

Command Format (5 bytes)

| Command                      | Class | INS | P1  | P2                           | Le  |
|------------------------------|-------|-----|-----|------------------------------|-----|
| Set PICC Operating Parameter | FFh   | 00h | 51h | New PICC Operating Parameter | 00h |

Response Format (2 bytes)

| Response | Data Out |                          |
|----------|----------|--------------------------|
| Result   | 90h      | PICC Operating Parameter |

PICC Operating Parameter

| Bit | Parameter   | Description   | Option                    |
|-----|---|---|---------------------------|
| 7   | Auto PICC Polling   | To enable the PICC Polling  | 1 = Enable<br>0 = Disable |
| 6   | Auto ATS Generation   | To issue ATS Request whenever an ISO14443-4 Type A tag is activated | 1 = Enable<br>0 = Disable |
| 5   | Polling Interval  | To set the time interval between successive PICC Polling.           | 1 = 250 ms<br>0 = 500 ms  |
| 4   | FeliCa 424K   | The Tag Types to be detected during PICC Polling.                   | 1 = Detect<br>0 = Skip    |
| 3   | FeliCa 212K   |   | 1 = Detect<br>0 = Skip    |
| 2   | Topaz   |   | 1 = Detect<br>0 = Skip    |
| 1   | ISO 14443 Type B  |   | 1 = Detect<br>0 = Skip    |
| 0   | ISO 14443 Type A<br>#To detect the MIFARE Tags, the Auto ATS Generation must be disabled first. |   | 1 = Detect<br>0 = Skip    |

**Note:** Default Value = FFh



## 6.6. Set Timeout Parameter

This command sets the timeout parameter of the contactless chip response time.

Command Format (5 bytes)

| Command               | Class | INS | P1  | P2                               | Le  |
|-----------------------|-------|-----|-----|----------------------------------|-----|
| Set Timeout Parameter | FFh   | 00h | 41h | Timeout Parameter (Unit: 5 sec.) | 00h |

Where:

|            |  |
|------------|--|
| <b>P2</b>  | Timeout Parameter                        |
| 00h:       | No Timeout check                         |
| 01h – FEh: | Timeout with 5 second unit               |
| FFh:       | Wait until the contactless chip responds |

Response Format (2 bytes)

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



## 6.7. Set buzzer output during card detection

This command sets the buzzer output during card detection. The default output is ON.

Command Format (5 bytes)

| Command                                 | Class | INS | P1  | P2             | Le  |
|---|-------|-----|-----|----------------|-----|
| Set Buzzer Output during Card Detection | FFh   | 00h | 52h | PollBuzzStatus | 00h |

Where:

**P2** PollBuzzStatus

00h: Buzzer will NOT turn on when a card is detected

FFh: Buzzer will turn on when a card is detected

Response Format (2 bytes)

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

## 7.0. 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 PC. You must make sure that the PC/SC Escape Command has been set. See **Appendix A** for more details.

Step 1. The first thing is to connect the “ACR122U PICC Interface”.

Step 2. Access the PICC by sending APDU commands.

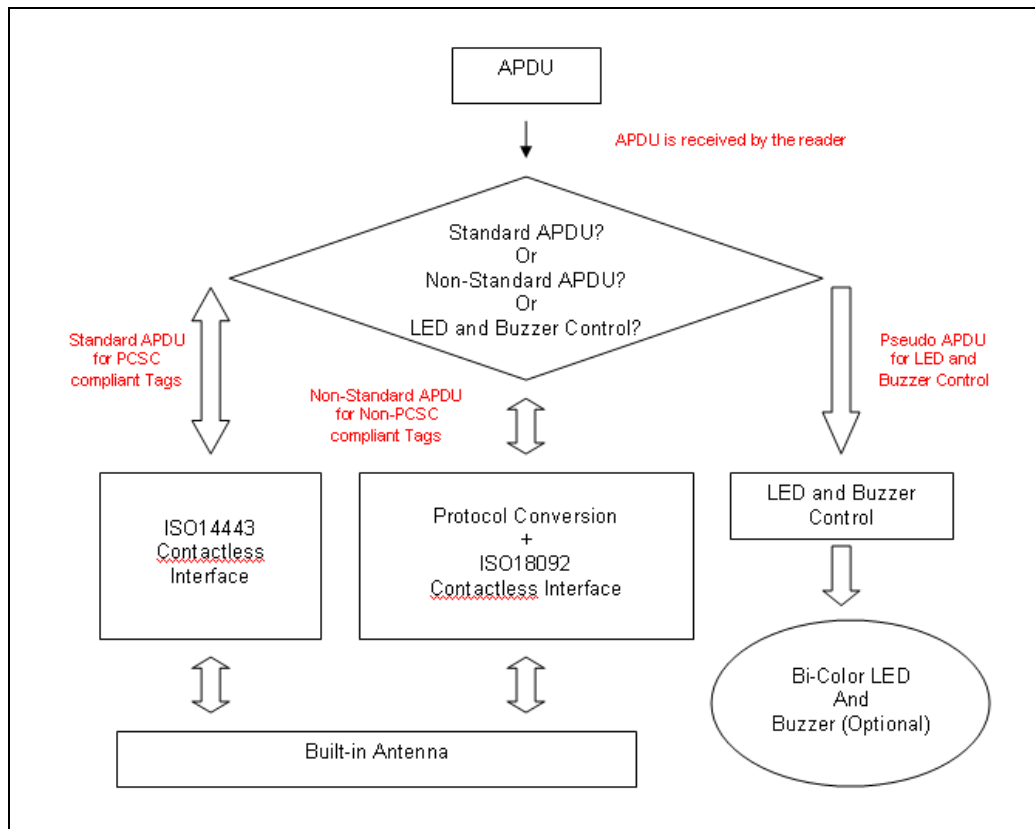
:

:

Step N. Disconnect the “ACR122U PICC Interface”. Shut down the application.

### Notes:

1. The antenna can be switched off in order to save the power.
  - Turn off the antenna power: FF 00 00 00 04 D4 32 01 00h
  - Turn on the antenna power: FF 00 00 00 04 D4 32 01 01h
2. Standard and Non-Standard APDUs Handling.
  - PICCs that use Standard APDUs: ISO14443-4 Type A and B, MIFARE .. etc
  - PICCs that use Non-Standard APDUs: FeliCa, Topaz .. etc.



**Figure 3:** Basic Program Flow for Contactless Applications



1. For the ACR122U PICC Interface, ISO 7816 T=1 protocol is used.
  - PC → Reader: Issue an APDU to the reader.
  - Reader → PC: The response data is returned.



## 7.1. How to access PC/SC-compliant tags (ISO 14443-4)?

Basically, all ISO 14443-4 compliant cards (PICCs) would understand the ISO 7816-4 APDUs. The ACR122U Reader just needs to communicate with the ISO 14443-4 compliant cards through exchanging ISO 7816-4 APDUs and Responses. ACR122U will handle the ISO 14443 Parts 1-4 Protocols internally.

MIFARE 1K, 4K, Mini and Ultralight tags are supported through the T=CL emulation. Simply treat the MIFARE tags as standard ISO 14443-4 tags. For more information, please refer to topic: **PICC Commands for MIFARE Classic Memory Tags.**

### ISO 7816-4 APDU Format

| 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 |

### ISO 7816-4 Response Format (Data + 2 bytes)

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

### Response Codes

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

Typical sequence may be:

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

1. Connect the Tag
2. Send an APDU, Get Challenge.  
<< 00 84 00 00 08h  
>> 1A F7 F3 1B CD 2B A9 58h [90 00h]

**Note:** For ISO14443-4 Type A tags, the ATS can be obtained by using the APDU "FF CA 00 00 01h"



## 7.2. How to access MIFARE DESFire tags (ISO 14443-4)?

MIFARE® DESFire® supports ISO 7816-4 APDU Wrapping and Native modes. Once the MIFARE 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 = 90; INS = 0A (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]

The Status Code [91 AFh] is defined in DESFire specification. Please refer to the 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 MIFARE DESFire card.

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

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=AF

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

We 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 [\$1DD7] 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 MIFARE 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 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.



### 7.3. How to access FeliCa tags (ISO 18092)?

Typical sequence may be:

1. Present the FeliCa Tag and Connect the PICC Interface.
2. Read/Update the memory of the tag.

Step 1) **Connect the tag.**

The ATR = 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 **F0 11** 00 00 00 00 8Ah

In which,

**F0 11 = FeliCa 212K**

Step 2) **Read the memory block without using Pseudo APDU.**

<< 10 06h [8-byte NFC ID] 01 09 01 01 80 00h

>> 1D 07h [8-byte NFC ID] 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AAh [90 00h]

or

Step 2) **Read the memory block using Pseudo APDU.**

<< **FF 00 00 00** [13] **D4 40 01** 10 06 [8-byte NFC ID] 01 09 01 01 80 00h

In which,

**[13]** is the length of the Pseudo Data “**D4 40 01**.. 80 00h”

**D4 40 01h** is the Data Exchange Command

>> **D5 41 00** 1D 07h [8-byte NFC ID] 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AAh [90 00h]

In which, **D5 41 00h** is the Data Exchange Response

**Note:** The **NFC ID** can be obtained by using the APDU “**FF CA 00 00 00h**”

Please refer to the FeliCa specification for more detailed information.



## 7.4. How to access NFC Forum Type 1 Tags (ISO 18092)?

Examples of these tags are Jewel and Topaz tags.

Typical sequence may be:

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

Step 1) **Connect the tag.**

The ATR = 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 **F0 04** 00 00 00 00 9Fh

In which, **F0 04** = Topaz

Step 2) **Read the memory address 08h (Block 1: Byte-0) without using Pseudo APDU**

<< **01 08h**

>> **18h** [90 00h]

In which, Response Data = **18h**

or

Step 2) **Read the memory address 08h (Block 1: Byte-0) using Pseudo APDU**

<< **FF 00 00 00 [05] D4 40 01 01 08h**

**In which,**

**[05h]** is the length of the Pseudo APDU Data "**D4 40 01 01 08h**"

**D4 40 01h** is the DataExchange Command.

**01 08h** is the data to be sent to the tag.

>> **D5 41 00 18h** [90 00h]

In which, Response Data = **18h**

*Tip: To **read all** the memory content of the tag*

<< **00h**

>> **11 48 18 26 .. 00h** [90 00h]

Step 3) **Update the memory address 08h (Block 1: Byte-0) with the data FFh**

<< **53 08 FFh**

>> **FFh** [90 00h]

In which, Response Data = **FFh**



Memory Address = Block No \* 8 + Byte No

E.g. Memory Address 08h (hex) = 1 x 8 + 0 = Block 1: Byte-0 = Data0

E.g. Memory Address 10h (hex) = 2 x 8 + 0 = Block 2: Byte-0 = Data8

| HR0             | HR1             |
|-----------------|-----------------|
| 11 <sub>h</sub> | XX <sub>h</sub> |

| EEPROM Memory Map |           |              |        |        |        |        |        |        |              |          |
|-------------------|-----------|--------------|--------|--------|--------|--------|--------|--------|--------------|----------|
| Type              | Block No. | Byte-0 (LSB) | Byte-1 | Byte-2 | Byte-3 | Byte-4 | Byte-5 | Byte-6 | Byte-7 (MSB) | Lockable |
| UID               | 0         | UID-0        | UID-1  | UID-2  | UID-3  | UID-4  | UID-5  | UID-6  |              | Locked   |
| Data              | 1         | Data0        | Data1  | Data2  | Data3  | Data4  | Data5  | Data6  | Data7        | Yes      |
| Data              | 2         | Data8        | Data9  | Data10 | Data11 | Data12 | Data13 | Data14 | Data15       | Yes      |
| Data              | 3         | Data16       | Data17 | Data18 | Data19 | Data20 | Data21 | Data22 | Data23       | Yes      |
| Data              | 4         | Data24       | Data25 | Data26 | Data27 | Data28 | Data29 | Data30 | Data31       | Yes      |
| Data              | 5         | Data32       | Data33 | Data34 | Data35 | Data36 | Data37 | Data38 | Data39       | Yes      |
| Data              | 6         | Data40       | Data41 | Data42 | Data43 | Data44 | Data45 | Data46 | Data47       | Yes      |
| Data              | 7         | Data48       | Data49 | Data50 | Data51 | Data52 | Data53 | Data54 | Data55       | Yes      |
| Data              | 8         | Data56       | Data57 | Data58 | Data59 | Data60 | Data61 | Data62 | Data63       | Yes      |
| Data              | 9         | Data64       | Data65 | Data66 | Data67 | Data68 | Data69 | Data70 | Data71       | Yes      |
| Data              | A         | Data72       | Data73 | Data74 | Data75 | Data76 | Data77 | Data78 | Data79       | Yes      |
| Data              | B         | Data80       | Data81 | Data82 | Data83 | Data84 | Data85 | Data86 | Data87       | Yes      |
| Data              | C         | Data88       | Data89 | Data90 | Data91 | Data92 | Data93 | Data94 | Data95       | Yes      |
| Reserved          | D         |              |        |        |        |        |        |        |              |          |
| Lock/Reserved     | E         | LOCK-0       | LOCK-1 | OTP-0  | OTP-1  | OTP-2  | OTP-3  | OTP-4  | OTP-5        |          |

|  |                           |
|--|---------------------------|
|  | Reserved for internal use |
|  | User Block Lock & Status  |
|  | OTP bits                  |

**Figure 4: Topaz Memory Map**

Please refer to the Jewel and Topaz specification for more detailed information.



## 7.5. Get the current setting of the contactless interface

Step 1. Get Status Command.

<< FF 00 00 00 02 D4 04h

>> D5 05h [Err] [Field] [NbTg] [Tg] [BrRx] [BrTx] [Type] 80 90 00h

Or if no tag is in the field

>> D5 05 00 00 00 80 90 00h

[Err] is an error code corresponding to the latest error detected.

Field indicates if an external RF field is present and detected (Field = 01h) or not (Field = 00h).

[NbTg] is the number of targets. The default value is 1.

[Tg]: logical number

[BrRx] : bit rate in reception

00h: 106 Kbps

01h: 212 Kbps

02h: 424 Kbps

[BrTx] : bit rate in transmission

00h: 106 Kbps

01h: 212 Kbps

02h: 424 Kbps

[Type ]: modulation type

00h: ISO 14443 or MIFARE

10h: FeliCa

01h: Active mode

02h: Innovision Jewel tag



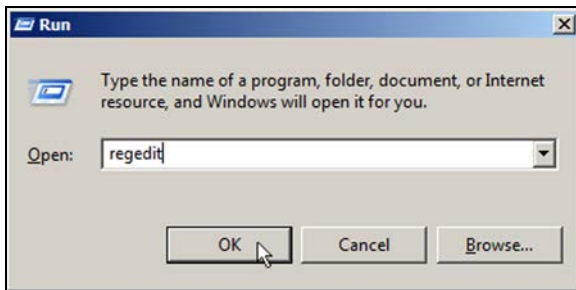
## Appendix A. ACR122U PC/SC Escape Command

1. Select the “ACS ACR122U PICC Interface 0”
2. Select the “Shared Mode” if the “ACR122U PICC Interface” is already connected, or “Direct Mode” if the “ACR122U PICC Interface” is not connected.
3. Press the **Connect** button to establish a connection between the PC and the ACR122U reader.
4. Enter “3500” in the Command Text Box
5. Enter the PC/SC Escape Command, e.g. “FF 00 48 00 00h” and press the button “Send” to send the command to the reader. **#Get the firmware version**
6. Press the **Disconnect** button to break the connection.
7. In order to send or receive **Escape commands** to a reader, follow the instructions below
8. The vendor IOCTL for the **Escape** command is defined as follows:

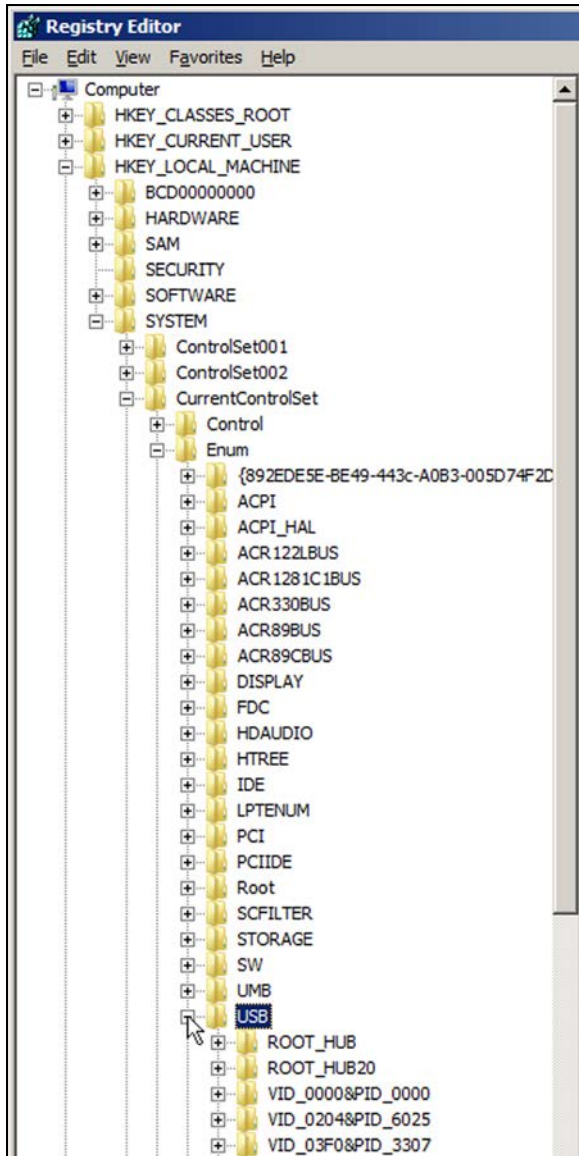
```
#define IOCTL_CCID_ESCAPE SCARD_CTL_CODE(3500)
```

The following instructions enumerate the steps to enable the PC/SC Escape command:

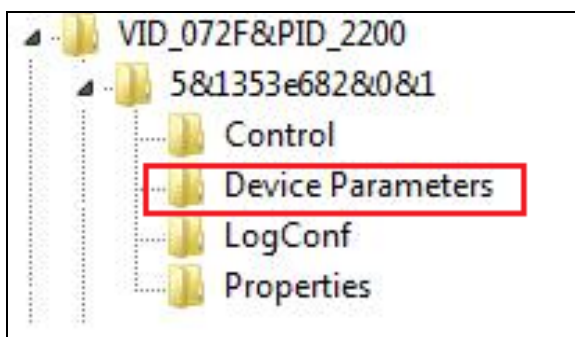
1. Execute the “regedit” in the “Run Command Menu” of Windows.



2. Add a DWORD “EscapeCommandEnable” under HKLM\SYSTEM\CCS\Enum\USB\Vid\_072F&Pid\_90CC\Device Parameters  
For Microsoft Vista, the path is:  
Computer\HKEY\_LOCAL\_MACHINE\SYSTEMS\CurrentControlSet\Enum\USB

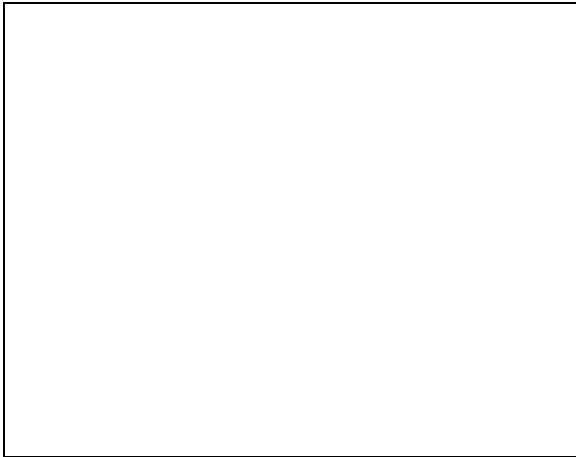


3. Look for: VID\_072F&PID\_2200, then expand the node. Look under Device parameters.

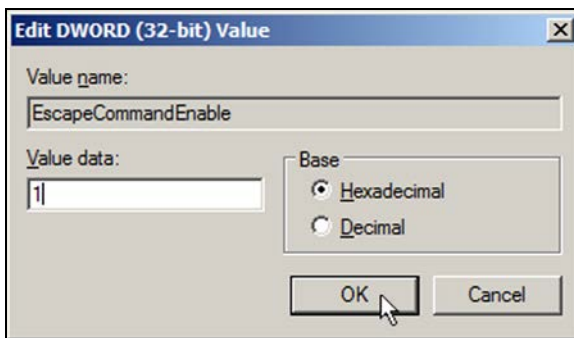




4. Create a DWORD entry (32-bit) with the name: `EscapeCommandEnable`



5. To Modify the value of the `EscapeCommandEnable` double click on the entry and input 1 in the Value data with the base set in Hexadecimal.

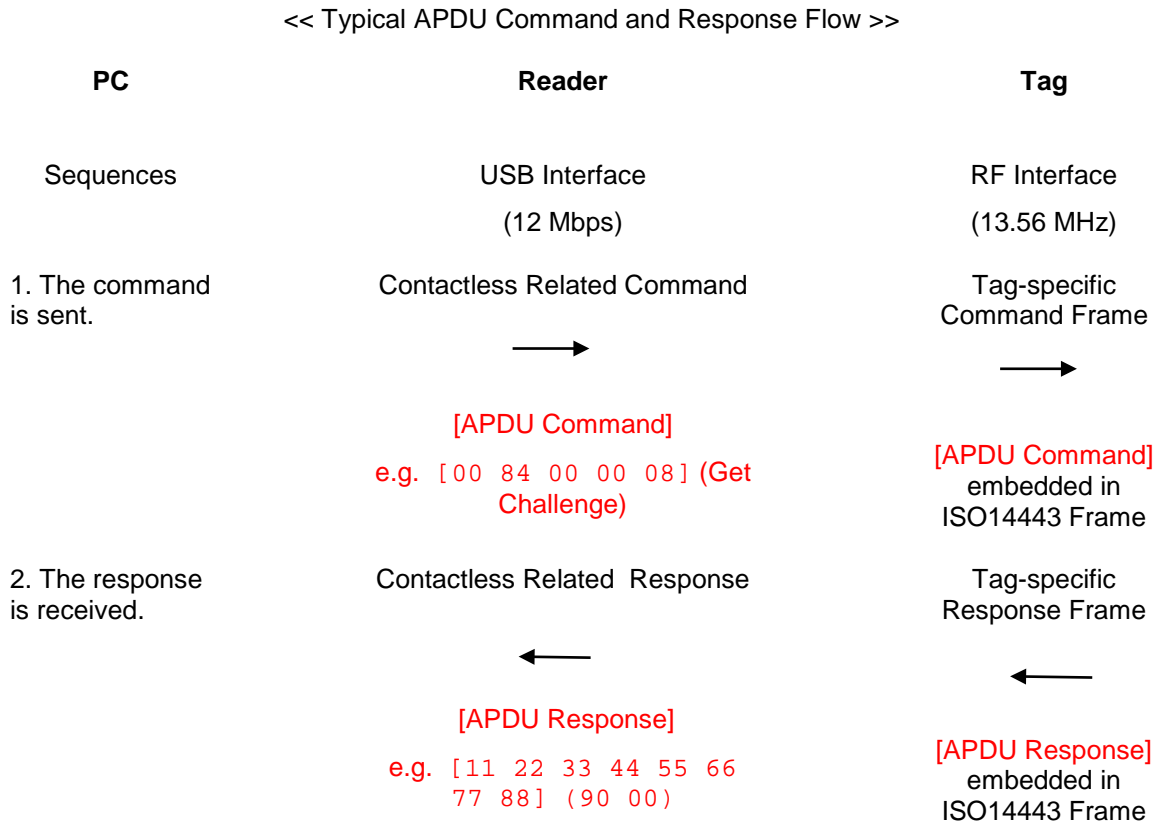






## Appendix B. APDU Command and Response Flow for ISO 14443-Compliant Tags

Assume an ISO 14443-4 Type B tag is used.

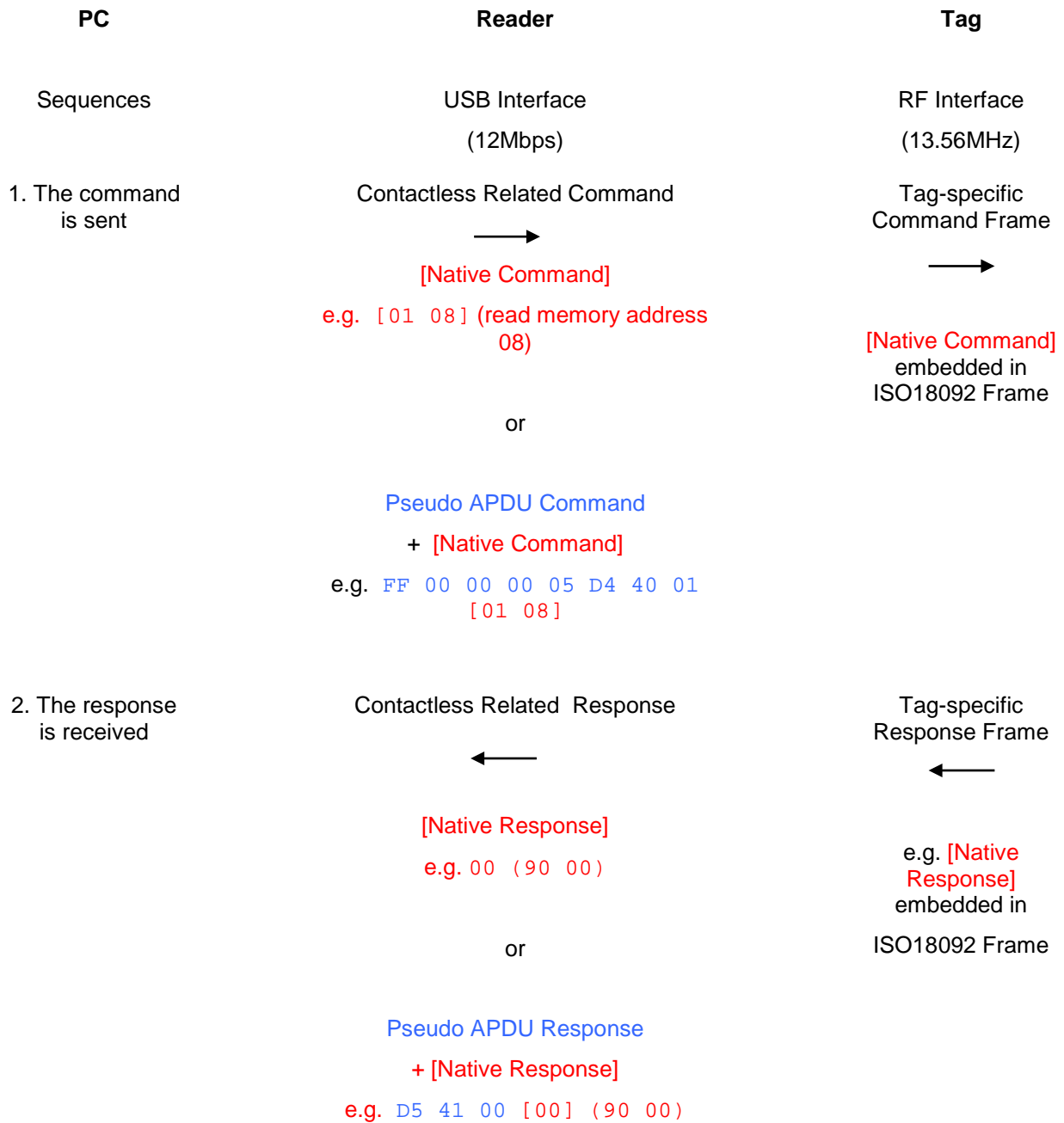




## Appendix C. APDU command and response flow for ISO 18092-compliant tags

*Note: Assume a TOPAZ tag is used.*

<< Typical APDU Command and Response Flow >>





## Appendix D. Error Codes

| Error Code | Error  |
|------------|--|
| 00h        | No Error   |
| 01h        | Time Out, the target has not answered  |
| 02h        | A CRC error has been detected by the contactless UART  |
| 03h        | A Parity error has been detected by the contactless UART   |
| 04h        | During a MIFARE anti-collision/select operation, an erroneous Bit Count has been detected  |
| 05h        | Framing error during MIFARE operation  |
| 06h        | An abnormal bit-collision has been detected during bit wise anti-collision at 106 Kbps   |
| 07h        | Communication buffer size insufficient   |
| 08h        | RF Buffer overflow has been detected by the contactless UART (bit BufferOvfl of the register <code>CL_ERROR</code> )   |
| 0Ah        | In active communication mode, the RF field has not been switched on in time by the counterpart (as defined in NFCIP-1 standard)  |
| 0Bh        | RF Protocol error (cf. reference [4], description of the <code>CL_ERROR</code> register)   |
| 0Dh        | Temperature error: the internal temperature sensor has detected overheating, and therefore has automatically switched off the antenna drivers  |
| 0Eh        | Internal buffer overflow   |
| 10h        | Invalid parameter (range, format, ...)   |
| 12h        | DEP Protocol: The chip configured in target mode does not support the command received from the initiator (the command received is not one of the following: <code>ATR_REQ</code> , <code>WUP_REQ</code> , <code>PSL_REQ</code> , <code>DEP_REQ</code> , <code>DSL_REQ</code> , <code>RLS_REQ</code> , ref. [1]).  |
| 13h        | DEP Protocol / MIFARE / ISO/IEC 14443-4: The data format does not match to the specification. Depending on the RF protocol used, it can be: <ul style="list-style-type: none"><li>• Bad length of RF received frame,</li><li>• Incorrect value of PCB or PFB,</li><li>• Invalid or unexpected RF received frame,</li><li>• NAD or DID incoherence.</li></ul> |
| 14h        | MIFARE: Authentication error   |
| 23h        | ISO/IEC 14443-3: UID Check byte is wrong   |
| 25h        | DEP Protocol: Invalid device state, the system is in a state which does not allow the operation  |
| 26h        | Operation not allowed in this configuration (host controller interface)  |
| 27h        | This command is not acceptable due to the current context of the chip (Initiator vs. Target, unknown target number, Target not in the good state, ...)   |
| 29h        | The chip configured as target has been released by its initiator   |
| 2Ah        | ISO/IEC 14443-3B only: the ID of the card does not match, meaning that the expected card has been exchanged with another one.  |



| Error Code | Error  |
|------------|--|
| 2Bh        | ISO/IEC 14443-3B only: the card previously activated has disappeared.                    |
| 2Ch        | Mismatch between the NFCID3 initiator and the NFCID3 target in DEP 212/424 kbps passive. |
| 2Dh        | An over-current event has been detected  |
| 2Eh        | NAD missing in DEP frame   |

**Table 7:** Error Codes



## Appendix E. Sample codes for setting the LED

**Example 1: To read the existing LED State.**

```
// Assume both Red and Green LEDs are OFF initially //
// Not link to the buzzer //
```

```
APDU = "FF 00 40 00 04 00 00 00 00h"
Response = "90 00h". RED and Green LEDs are OFF.
```

**Example 2: To turn on RED and Green Color LEDs.**

```
// Assume both Red and Green LEDs are OFF initially //
// Not link to the buzzer //
```

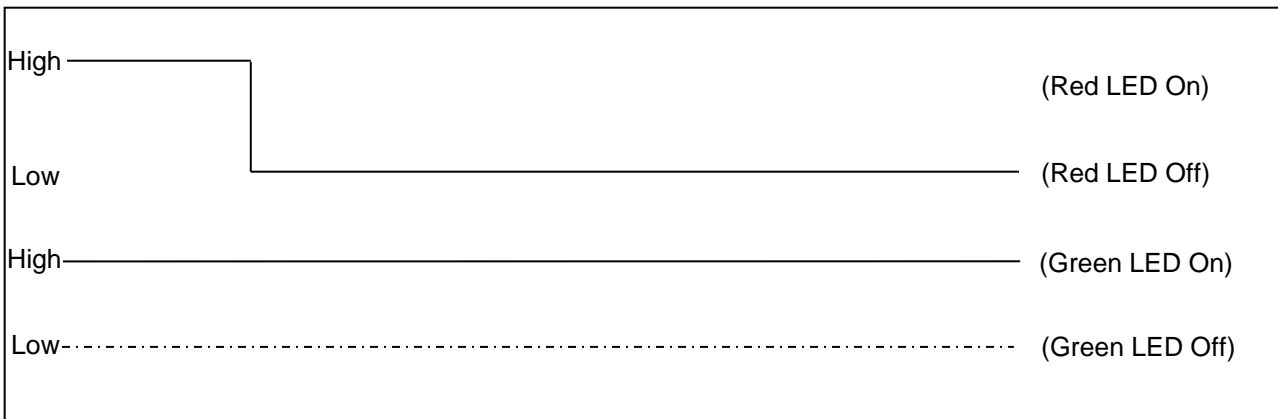
```
APDU = "FF 00 40 0F 04 00 00 00 00h"
Response = "90 03h". RED and Green LEDs are ON,
```

To turn off both RED and Green LEDs, APDU = "FF 00 40 0C 04 00 00 00 00h"

**Example 3: To turn off the RED Color LED only, and leave the Green Color LED unchanged.**

```
// Assume both Red and Green LEDs are ON initially //
// Not link to the buzzer //
```

```
APDU = "FF 00 40 04 04 00 00 00 00h"
Response = "90 02h". Green LED is not changed (ON); Red LED is OFF,
```

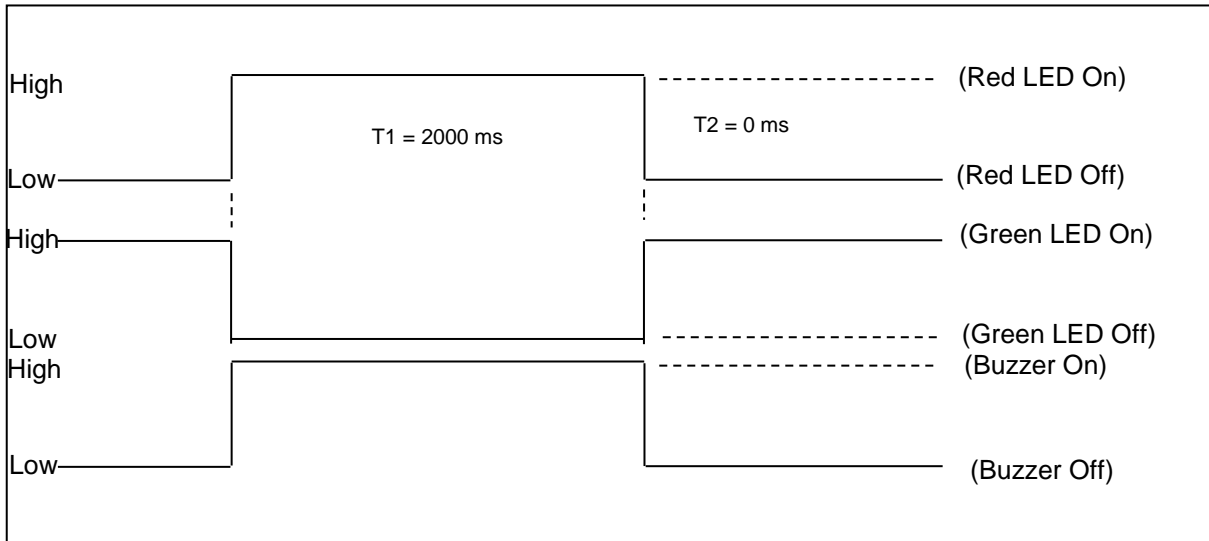




**Example 4: To turn on the Red LED for 2 seconds. After that, resume to the initial state.**

// Assume the Red LED is initially OFF, while the Green LED is initially ON. //

// The Red LED and buzzer will turn on during the T1 duration, while the Green LED will turn off during the T1 duration. //



1 Hz = 1000 ms Time Interval = 500 ms ON + 500 ms OFF

T1 Duration = 2000 ms = 14h

T2 Duration = 0 ms = 00h

Number of repetition = 01h

Link to Buzzer = 01h

APDU = "FF 00 40 50 04 14 00 01 01h"

Response = "90 02h"

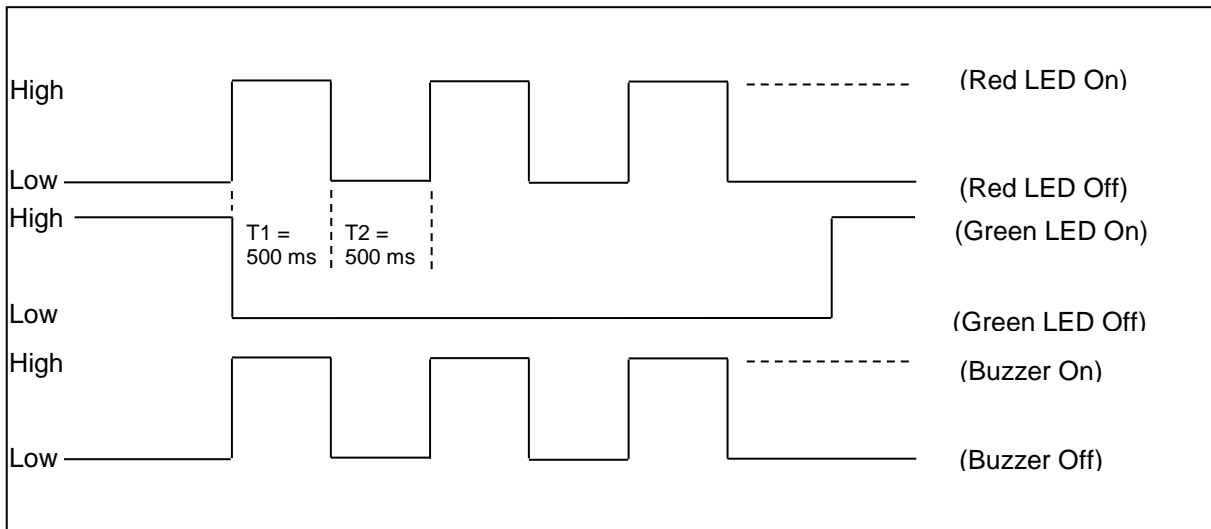
**Example 5: To make the Red LED blink at 1 Hz, three times. After which, it resumes to initial state.**

// Assume the Red LED is initially OFF, while the Green LED is initially ON. //

// The Initial Red LED Blinking State is ON. Only the Red LED will be blinking.

// The buzzer will turn on during the T1 duration, while the Green LED will turn off during both the T1 and T2 duration.

// After the blinking, the Green LED will turn ON. The Red LED will resume to the initial state after the blinking //



1 Hz = 1000 ms Time Interval = 500 ms ON + 500 ms OFF

T1 Duration = 500 ms = 05h

T2 Duration = 500 ms = 05h

Number of repetition = 03h

Link to Buzzer = 01h

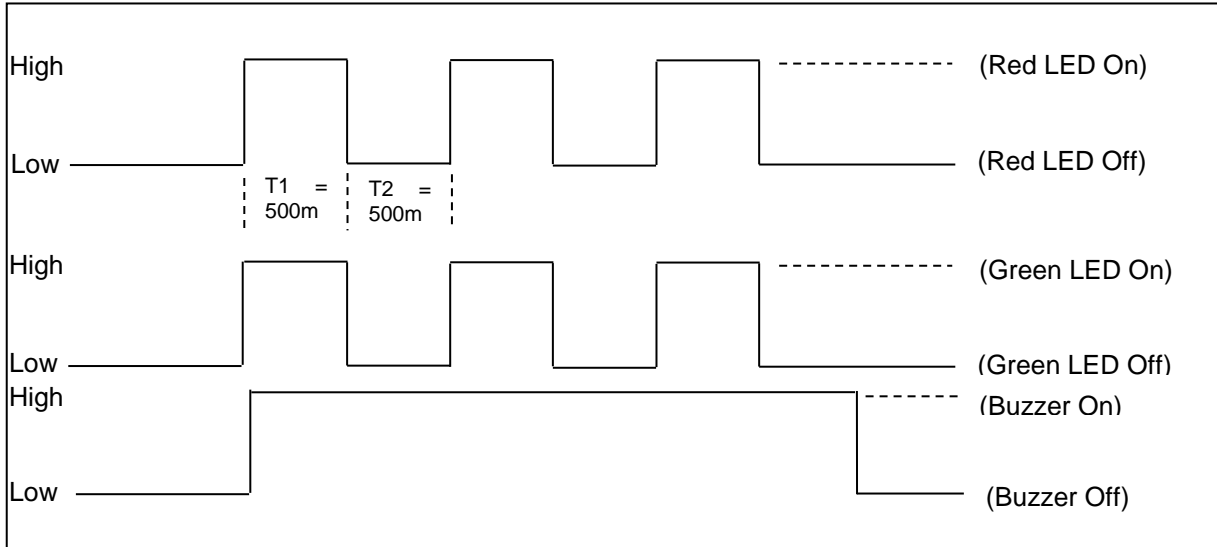
APDU = "FF 00 40 50 04 05 05 03 01h"

Response = "90 02h"



**Example 6: To make the Red and Green LEDs blink at 1 Hz three times.**

// Assume both the Red and Green LEDs are initially OFF. //  
 // Both Initial Red and Green Blinking States are ON //  
 // The buzzer will turn on during both the T1 and T2 duration//



1 Hz = 1000 ms Time Interval = 500 ms ON + 500 ms OFF

T1 Duration = 500 ms = 05h

T2 Duration = 500 ms = 05h

Number of repetition = 03h

Link to Buzzer = 03h

APDU = "FF 00 40 F0 04 05 05 03 03h"

Response = "90 00h"



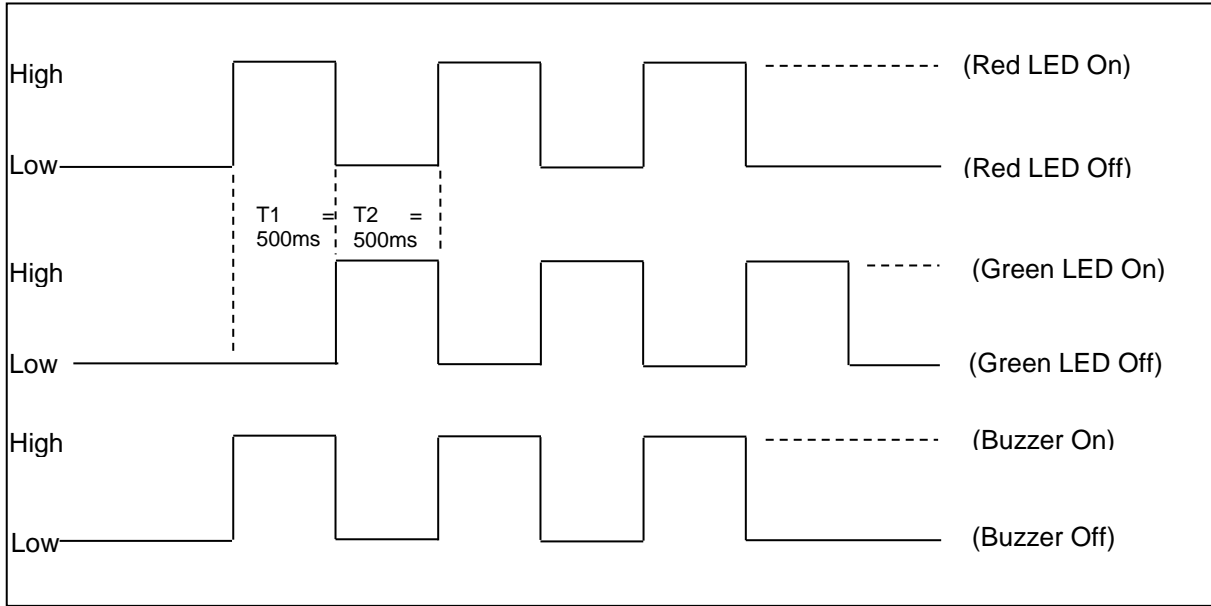


**Example 7: To make Red and Green LED blink in turns at 1 Hz three times.**

// Assume both Red and Green LEDs are initially OFF. //

// The Initial Red Blinking State is ON; The Initial Green Blinking States is OFF //

// The buzzer will turn on during the T1 duration//



1 Hz = 1000 ms Time Interval = 500 ms ON + 500 ms OFF

T1 Duration = 500 ms = 05h

T2 Duration = 500 ms = 05h

Number of repetition = 03h

Link to Buzzer = 01h

APDU = "FF 00 40 D0 04 05 05 03 01h"; Response = "90 00h"

MIFARE, MIFARE Classic, MIFARE DESFire and MIFARE Ultralight are registered trademarks of NXP B.V. and are used under license. Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries.