Advanced Card Systems Ltd.
Card & Reader Technologies

ACR122U USB NFC Reader

Application Programming Interface V2.02

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

The ACR122U is a PC-linked contactless smart card reader/writer used for accessing ISO 14443-4 Type A and Type B, Mifare, ISO 18092 or NFC, and FeliCa tags. The ACR122U is PC/SC compliant so it is compatible with existing PC/SC applications. Furthermore, the standard Microsoft CCID driver is used to simplify driver installation.

The ACR122U serves as the intermediary device between the personal computer and the contactless tag via the USB interface. The reader carries out the command from the PC whether the command is used in order to communicate with a contactless tag, or control the device peripherals (LED or buzzer).

The ACR122U uses the PC/SC APDUs for contactless tags following the PC/SC Specification and makes use of pseudo APDUs in sending commands for ISO 18092 tags and controlling the device peripherals. This document will discuss the ACR122U can be used in your smart card system.

1.1. Features

- USB 2.0 Full Speed Interface
- CCID Compliance
- Smart Card Reader:
    - Read/Write speed of up to 424 kbps
    - Built-in antenna for contactless tag access, with card reading distance of up to 50 mm (depending on tag type)
    - Support for ISO 14443 Part 4 Type A and B cards, Mifare, FeliCa, and all four types of NFC (ISO/IEC 18092 tags)
    - Built-in anti-collision feature (only one tag is accessed at any time)
- Application Programming Interface:
    - Supports PC/SC
    - Supports CT-API (through wrapper on top of PC/SC)
- Built-in Peripherals:
    - User-controllable bi-color LED
    - User-controllable buzzer
- Supports Android™ OS 3.1 and above
- Compliant with the following standards:
    - ISO 14443
    - CE
    - FCC
    - KC
    - VCCI
    - PC/SC
    - CCID
    - Microsoft WHQL
    - RoHS

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_{BUS}	+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

2.0. Implementation

2.1. Communication Flow Chart of ACR122U

The Standard Microsoft CCID and PC/SC drivers are used; thus, no ACS drivers are required because the drivers are already built inside the windows operating system. Your computer's registry settings can also be modified to be able to use the full capabilities of the ACR122U NFC Reader. See **Appendix A** for more details.

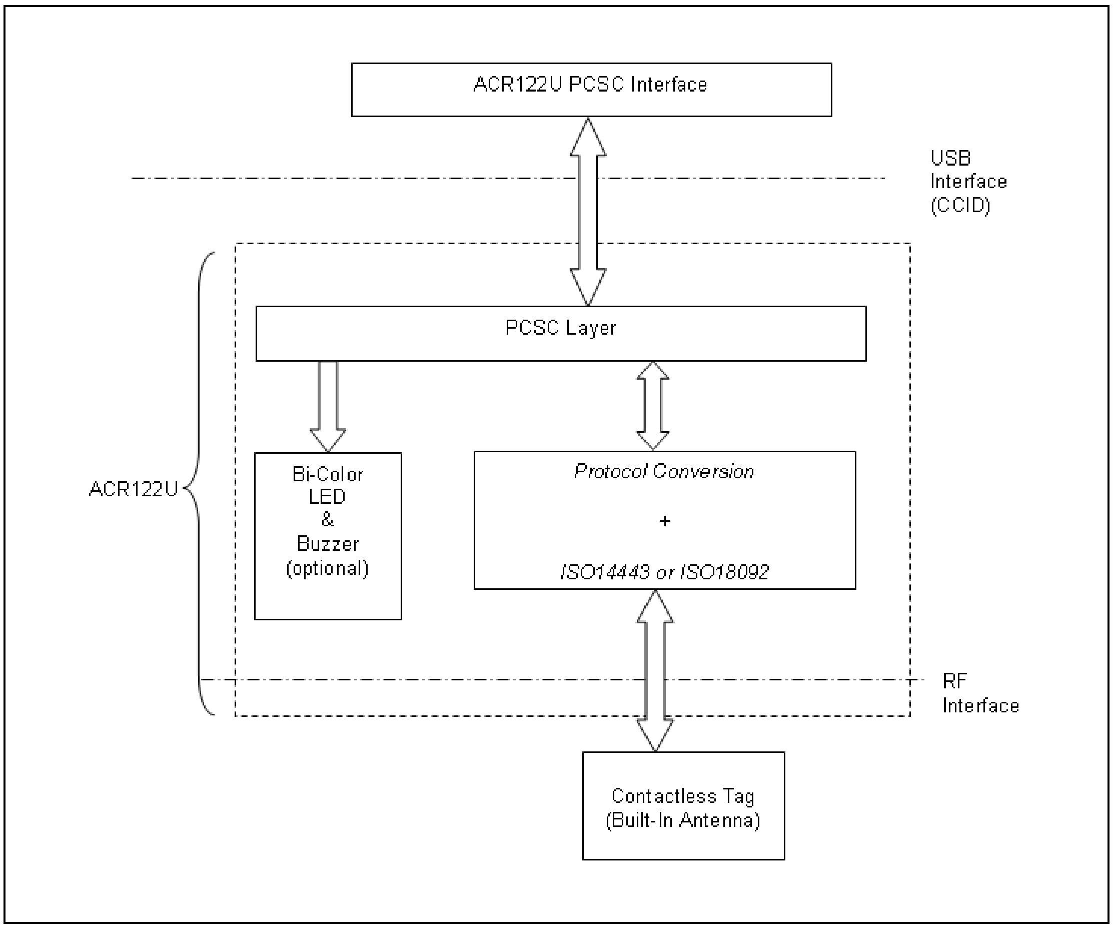

ACR122U PCSC Interface
USB Interface (CCID)
PCSC Layer
ACR122U
Bi-Color LED & Buzzer (optional)
Protocol Conversion + ISO14443 or ISO18092
RF Interface
Contactless Tag (Built-In Antenna)

Figure 1: Communication Flow Chart of ACR122U

2.2. Smart Card Reader Interface Overview

Click on the “Device Manager” to find out the “ACR122U PICC Interface.” The standard Microsoft USB CCID Driver is used.

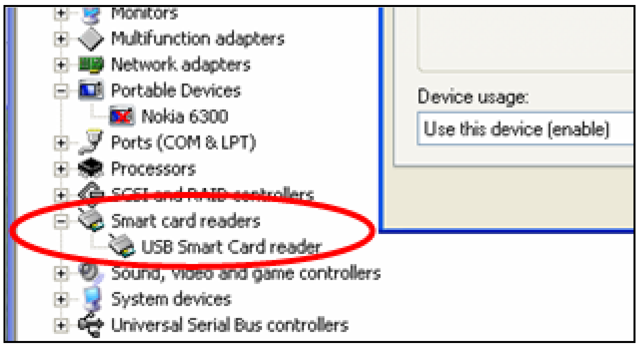


Figure 2: Smart Card Reader Interface on the Device Manager

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. 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. Lower nibble 0 means T = 0
3	01h	TD2	Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1
4 To 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	RFU # 00 00 00 00h
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 00 03 06h	03h	00 01h	00 00 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 1K)

Where, Card Name (C0 .. C1)

00 01h: Mifare 1K

00 02h: Mifare 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. 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. Lower nibble 0 means T = 0
3	01h	TD2	Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1
4 to 3 + N	XXh	T1	Historical Bytes: ISO 14443A: The historical bytes from ATS response. Refer to the ISO14443-4 specification. ISO 14443B: The higher layer response from the ATTRIB response (ATQB). Refer to the ISO14443-3 specification.
	XXh XX 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		
				T1	Tk	TCK
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		
				T1	Tk	TCK
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

The “Get Data command” will return 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 01h	00h	00h (Full Length)

Get UID Response Format (UID + 2 Bytes) if P1 = 0x00h

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

Get ATS of a ISO 14443 A card (ATS + 2 Bytes) if P1 = 0x01h

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:

1. To get the serial number of the “connected PICC”
   UINT8 GET_UID[5]={0xFFh, 0xCAh, 0x00h, 0x00h, 0x04h};
2. To get the ATS of the “connected ISO 14443 A PICC”
   UINT8 GET_ATS[5]={0xFFh, 0xCAh, 0x01h, 0x00h, 0x04h};

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

5.1. Load Authentication Keys

The "Load Authentication Keys command" will load the authentication keys into the reader. The authentication keys are used to authenticate the particular sector of the Mifare 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.
	0x00h = Key is loaded into the reader volatile memory.
	Other = Reserved.
Key Number:	1 Byte.
	0x00h ~ 0x01h = 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 0x00h.

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

5.2. Authentication

The "Authentication 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 0x01h	0x00h	Block Number	Key Type	Key Number

Where:

Block Number:	1 Byte. This is the memory block to be authenticated.
Key Type:	1 Byte
	0x60h = Key is used as a TYPE A key for authentication.
	0x61h = Key is used as a TYPE B key for authentication.
Key Number:	1 Byte
	0x00h ~ 0x01h = Key Location.

Note: *For Mifare 1K Card, it has totally 16 sectors and each sector consists of 4 consecutive blocks. E.g. Sector 0x00h consists of Blocks {0x00h, 0x01h, 0x02h and 0x03h}; Sector 0x01h consists of Blocks {0x04h, 0x05h, 0x06h and 0x07h}; the last sector 0x0F consists of Blocks {0x3Ch, 0x3Dh, 0x3Eh and 0x3Fh}.*

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 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 (Total 16 sectors. Each sector consists of 4 consecutive blocks)	Data Blocks (3 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)
Sector 0	0x00 ~ 0x02h	0x03h
Sector 1	0x04 ~ 0x06h	0x07h
..		
..		
Sector 14	0x38 ~ 0x0Ah	0x3Bh
Sector 15	0x3C ~ 0x3Eh	0x3Fh

1K Bytes

Table 4: Mifare 1k Memory Map

Sectors (Total 32 sectors. Each sector consists of 4 consecutive blocks)	Data Blocks (3 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)
Sector 0	0x00 ~ 0x02h	0x03h
Sector 1	0x04 ~ 0x06h	0x07h
..		
..		
Sector 30	0x78 ~ 0x7Ah	0x7Bh
Sector 31	0x7C ~ 0x7Eh	0x7Fh

2K Bytes

Sectors (Total 8 sectors. Each sector consists of 16 consecutive blocks)	Data Blocks (15 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)
Sector 32	0x80 ~ 0x8Eh	0x8Fh
Sector 33	0x90 ~ 0x9Eh	0x9Fh
..		
..		
Sector 38	0xE0 ~ 0xEEh	0xEFh
Sector 39	0xF0 ~ 0xFEh	0xFFh

2K Bytes

Table 5: Mifare 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:

1. To authenticate the Block 0x04h with a {TYPE A, key number 0x00h}. For PC/SC V2.01, Obsolete.

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

2. To authenticate the Block 0x04h with a {TYPE A, key number 0x00h}. For PC/SC V2.07

   alaAPDU = {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

The "Read Binary Blocks command" is used for retrieving "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:

Block Number:	1 Byte. The block to be accessed
Number of Bytes to Read:	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:

1. Read 16 bytes from the binary block 0x04h (Mifare 1K or 4K)

   APDU = {FF B0 00 04 10h}

2. Read 4 bytes from the binary Page 0x04h (Mifare Ultralight)

   APDU = {FF B0 00 04 04h}

3. Read 16 bytes starting from the binary Page 0x04h (Mifare Ultralight) (Pages 4, 5, 6 and 7 will be read)

   APDU = {FF B0 00 04 10h}

5.4. Update Binary Blocks

The “Update Binary Blocks command” is used for writing “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 4 Bytes for Mifare Ultralight or 16 Bytes for Mifare 1K/4K

Where:

Block Number:	1 Byte. The starting block to be updated.
Number of Bytes to Update:	1 Byte. 16 bytes for Mifare 1K/4K 4 bytes for Mifare Ultralight.
Block Data:	4 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:

1. Update the binary block 0x04h 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}

2. Update the binary block 0x04h 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

The "Value Block Operation command" is used for manipulating 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.

0x00h = Store the VB_Value into the block. The block will then be converted to a value block.

0x01h = Increment the value of the value block by the VB_Value. This command is only valid for value block.

0x02h = 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 = {0xFFh, 0xFFh, 0xFFh, 0xFCh}

VB_Value			
MSB			LSB
FFh	FFh	FFh	FCh

Example 2: Decimal 1 = {0x00h, 0x00h, 0x00h, 0x01h}

VB_Value			
MSB			LSB
00h	00h	00h	01h

Value Block Operation 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.

5.5.2. Read Value Block

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

Read Value Block APDU Format (5 Bytes)

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

Where:

Block Number: 1 Byte. The value block to be accessed.

Read Value Block Response Format (4 + 2 Bytes)

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

Where:

Value: 4 Bytes. The value returned from the card. The value is a signed long integer (4 bytes).

Example 1: Decimal –4 = {0xFFh, 0xFFh, 0xFFh, 0xFCh}

Value			
MSB			LSB
FFh	FFh	FFh	FCh

Example 2: Decimal 1 = {0x00h, 0x00h, 0x00h, 0x01h}

Value			
MSB			LSB
00h	00h	00h	01h

Response Codes

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

5.5.3. Restore Value Block

The "Restore Value Block command" is used to copy 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:

1. Store a value "1" into block 0x05h

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

   Answer: 90 00h

2. Read the value block 0x05h

   APDU = {FF B1 00 05 00h}

   Answer: 00 00 00 01 90 00h [9000h]

3. Copy the value from value block 0x05h to value block 0x06h

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

   Answer: 90 00h [9000h]

4. Increment the value block 0x05h by "5"

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

   Answer: 90 00h [9000h]

6.0. Pseudo-APDUs

Pseudo-APDUs are used for the following:

- Exchanging Data with Non-PC/SC Compliant Tags.
- Retrieving and setting the reader parameters.
- The Pseudo-APDUs can be sent through the "ACR122U PICC Interface" if the tag is already connected.
- Or the Pseudo-APDUs can be sent by using "Escape Command" if the tag is not presented yet.

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	0xFFh	0x00h	0x00h	0x00h	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 APDU is used to control 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	0xFFh	0x00h	0x40h	LED State Control	0x04h	Blinking Duration Control

P2: LED State Control

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

Table 7: Bi-Color LED and Buzzer Control Format (1 Byte)

Data In: Blinking Duration Control

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

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

Where:

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

0x00h: The buzzer will not turn on

0x01h: The buzzer will turn on during the T1 Duration

0x02h: The buzzer will turn on during the T2 Duration

0x03h: 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.

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	

Table 8: Current LED State (1 Byte)

Notes:

1. *The LED State operation will be performed after the LED Blinking operation is completed.*
2. *The LED will not be changed if the corresponding LED Mask is not enabled.*
3. *The LED will not be blinking if the corresponding LED Blinking Mask is not enabled. Also, the number of repetition must be greater than zero.*
4. *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).*
5. *To control the buzzer only, just set the P2 "LED State Control" to zero.*
6. *The make the buzzer operating, the "number of repetition" must greater than zero.*
7. *To control the LED only, just set the parameter "Link to Buzzer" to zero.*

6.3. Get the Firmware Version of the Reader

This is used to retrieve the firmware version of the reader.

Command Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Get Firmware Version	0xFFh	0x00h	0x48h	0x00h	0x00h

Response Format (10 Bytes)

Response	Data Out
Result	Firmware Version

E.g. Response = 41 43 52 31 32 32 55 32 30 31h (Hex) = ACR122U201 (ASCII)

6.4. Get the PICC Operating Parameter

This is used to retrieve the PICC Operating Parameter of the reader.

Command Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Get PICC Operating Parameter	0xFFh	0x00h	0x50h	0x00h	0x00h

Response Format (1Byte)

Response	Data Out
Result	PICC Operating Parameter

6.5. Set the PICC Operating Parameter

This is used to set the PICC Operating Parameter of the reader.

Command Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Set PICC Operating Parameter	0xFFh	0x00h	0x51h	New PICC Operating Parameter	0x00h

Response Format (1 Byte)

Response	Data Out
Result	PICC Operating Parameter

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

Table 9: PICC Operating Parameter. (Default Value = FFh)

6.6. Set Timeout Parameter

This is used to set the Time out Parameter of the contactless chip response time.

Command Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Set Timeout Parameter	0xFFh	0x00h	0x41h	Timeout Parameter (Unit: 5 sec.)	0x00h

Where:

P2: Timeout Parameter.

0x00h: No Timeout check

0x01h – 0xFEh: Timeout with 5 second unit

0xFFh: Wait until the contactless chip responds

Response Format (8 Bytes)

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

6.7. Set Buzzer Output Enable for Card Detection

This is used to set the buzzer output during card detection. The default output is ON.

Command Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Set Buzzer Output for Card Detection	0xFFh	0x00h	0x52h	PollBuzzStatus	0x00h

Where:

P2: PollBuzzStatus.

0x00h: Buzzer will NOT turn ON when a card is detected

0xFFh: Buzzer will turn ON when a card is detected

Response Format (8 Bytes)

Results	SW1 SW2	Meaning
Success	90 00	The operation completed successfully.
Error	63 00	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.*

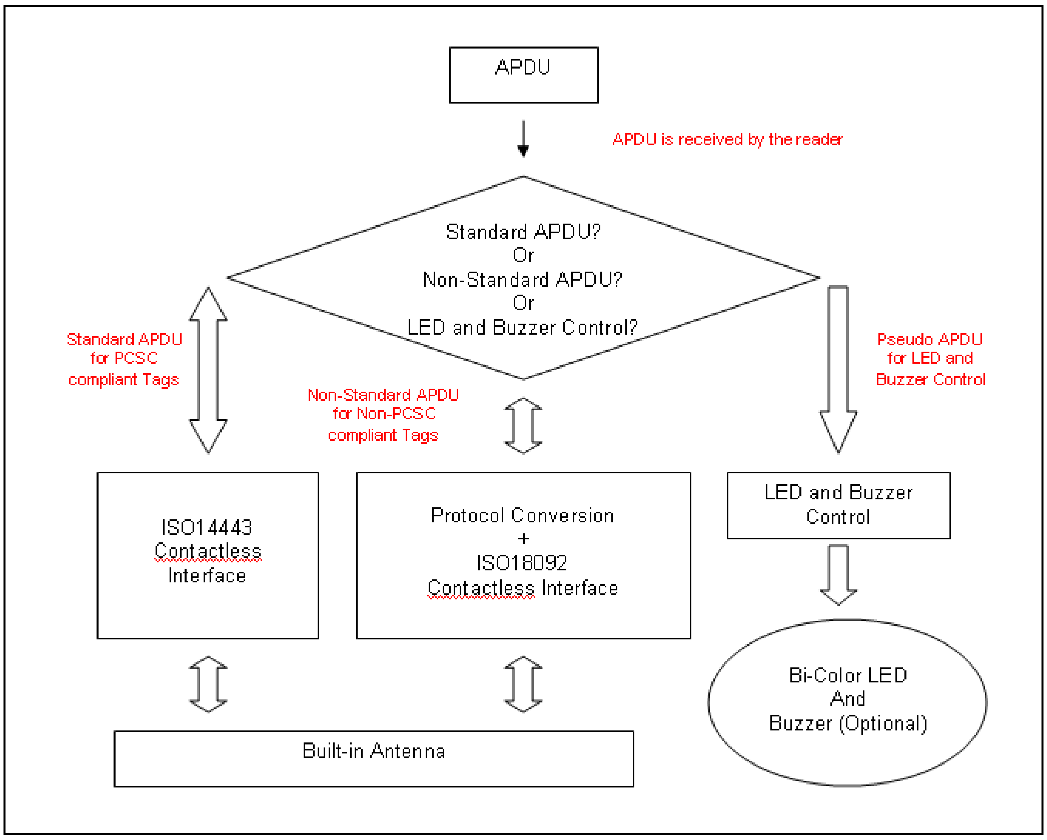

APDU
APDU is received by the reader
Standard APDU? Or Non-Standard APDU? Or LED and Buzzer Control?
Standard APDU for PCSC compliant Tags
Non-Standard APDU for Non-PCSC compliant Tags
Pseudo APDU for LED and Buzzer Control
ISO14443 Contactless Interface
Protocol Conversion + ISO18092 Contactless Interface
LED and Buzzer Control
Bi-Color LED And Buzzer (Optional)
Built-in Antenna

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 has 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. Just 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:

- Present the Tag and Connect the PICC Interface
- Read/Update the memory of the tag

Step 1) **Connect the Tag**

Step 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 DESFire Tags (ISO 14443-4)?

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: 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 = 0x90; INS = 0x0A (DESFire Instruction); P1 = 0x00h; P2 = 0x00h

Lc = 0x01h; Data In = 0x00h; Le = 0x00h (Le = 0x00h 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: 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=0x60

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=0xAF

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=0xAFh

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

Example 3: 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 DESFire Card.

The Data inside the blanket [$1DD7] can simply be ignored by the application.

Example 4: 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=0x60h

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

Step 2: Send an APDU {AFh} to get the second frame. INS=0xAFh

Answer: AF 04 01 01 00 06 18 05h[$1805h]

Step 3: Send an APDU {AFh} to get the last frame. INS=0xAFh

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), e.g. Jewel and Topaz Tags?

Typical sequence may be:

1. Present the Topaz 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 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**

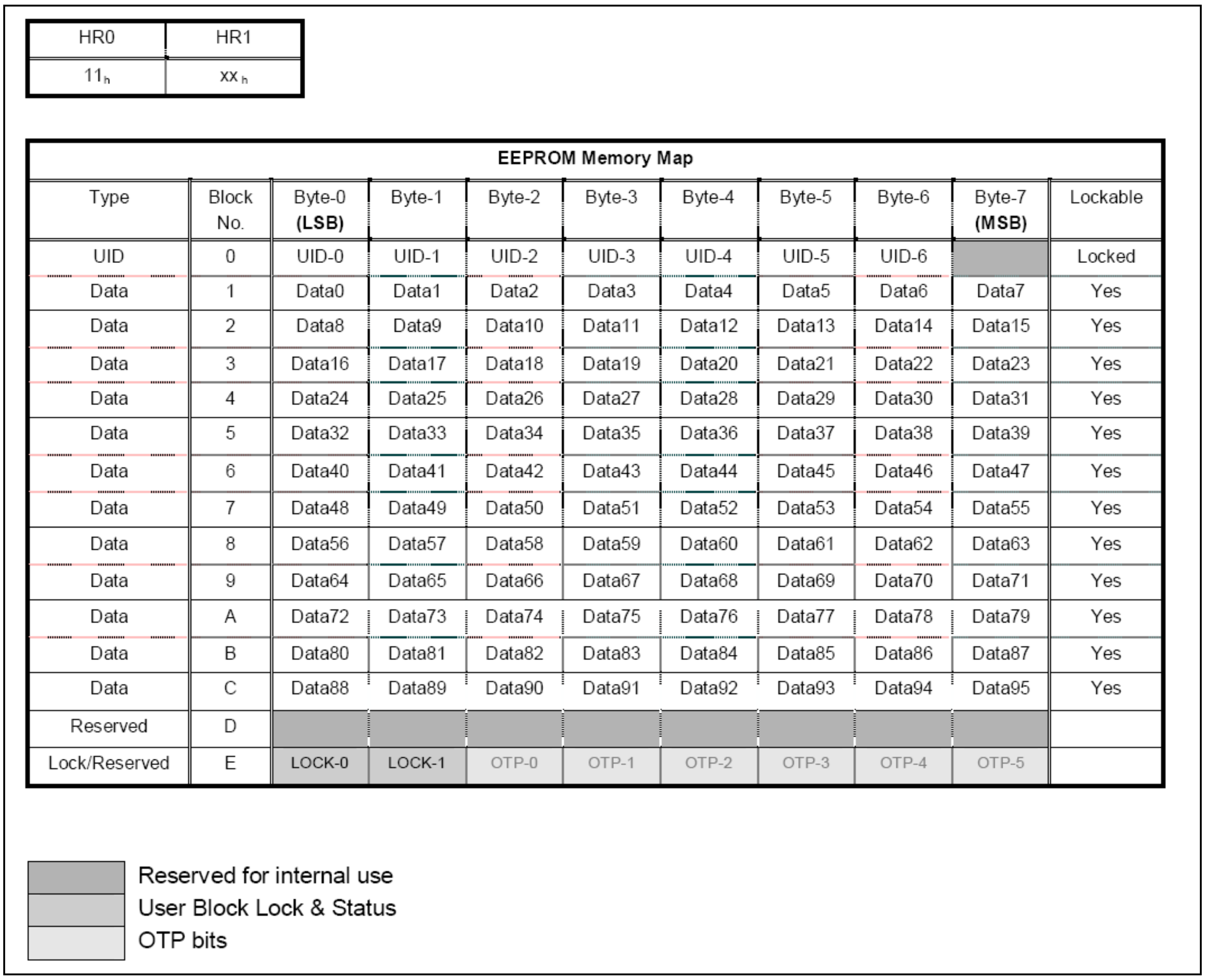
>> FFh [90 00h]

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_h	xx_h

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. Getting 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 = 0x01h) or not (Field = 0x00h).

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

[Tg]: logical number

[BrRx] : bit rate in reception

0x00h : 106 kbps

0x01h : 212 kbps

0x02h : 424 kbps

[BrTx] : bit rate in transmission

0x00h : 106 kbps

0x01h : 212 kbps

0x02h : 424 kbps

[Type]: modulation type

0x00h : ISO 14443 or Mifare

0x10h : FeliCa™

0x01h : Active mode

0x02h : Innovision Jewel tag

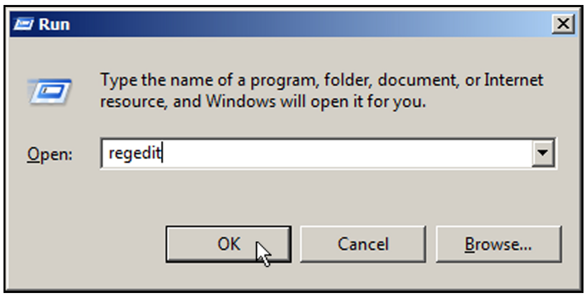
Appendix A. ACR122U PC/SC Escape Command

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.


Run

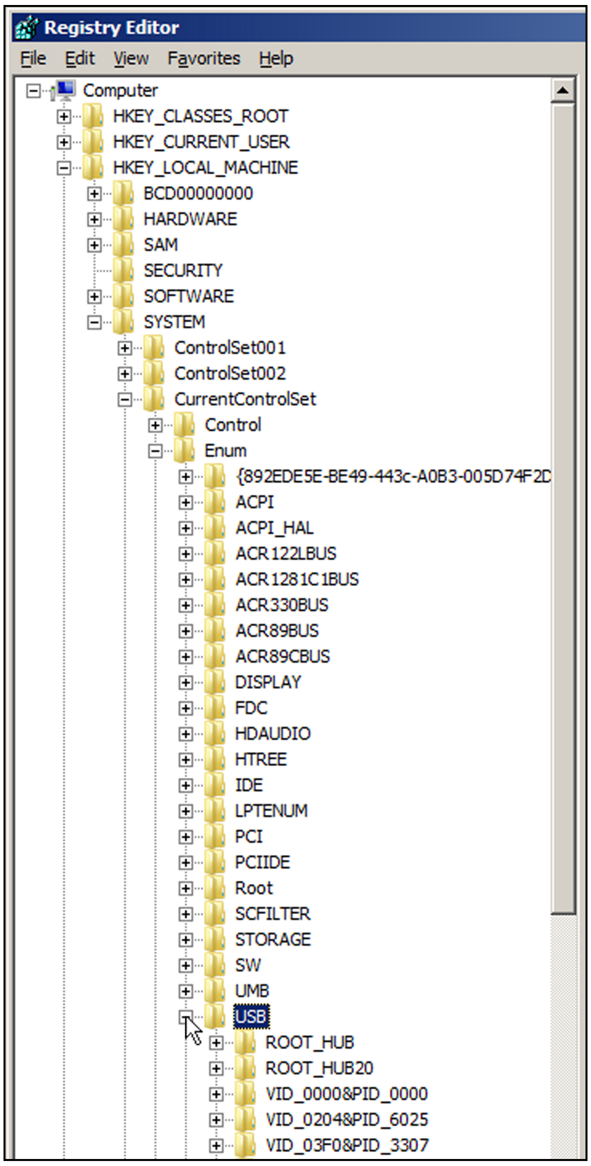
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: regedit

2. Add a DWORD “EscapeCommandEnable” under HKLM\SYSTEM\CCS\Enum\USB\Vid_072F&Pid_90CC\Device Parameters

   For Vista, the path is:

   Computer\HKEY_LOCAL_MACHINE\SYSTEMS\CurrentControlSet\Enum\USB

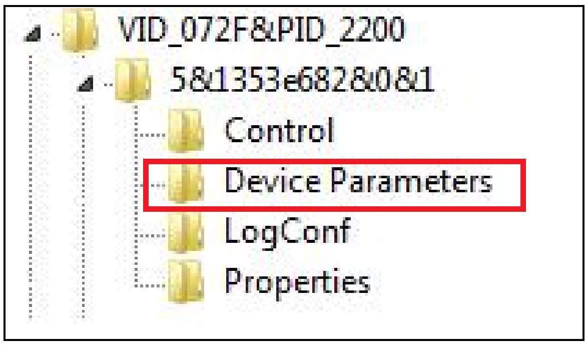


Registry Editor
File Edit View Favorites Help

```
Computer
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
    BCD00000000
    HARDWARE
    SAM
    SECURITY
    SOFTWARE
    SYSTEM
      ControlSet001
      ControlSet002
      CurrentControlSet
        Control
        Enum
          {892EDE5E-BE49-443c-A0B3-005D74F2D
          ACPI
          ACPI_HAL
          ACR122LBUS
          ACR1281C1BUS
          ACR330BUS
          ACR89BUS
          ACR89CBUS
          DISPLAY
          FDC
          HDAUDIO
          HTREE
          IDE
          LPTENUM
          PCI
          PCIIDE
          Root
          SCFILTER
          STORAGE
          SW
          UMB
          USB
            ROOT_HUB
            ROOT_HUB20
            VID_0000&PID_0000
            VID_0204&PID_6025
            VID_03F0&PID_3307
```

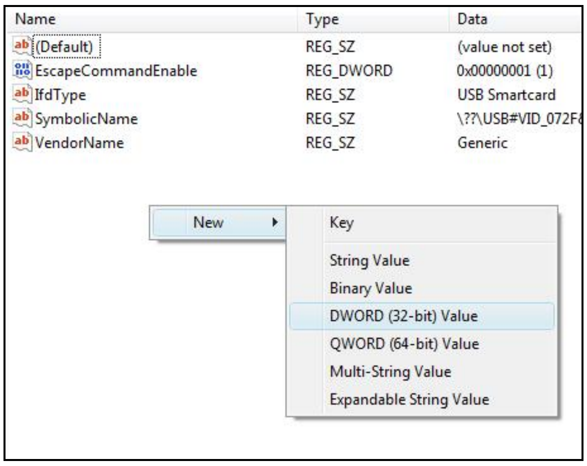
3. Look for: VID_072F&PID_2200

   Then expand the node. Look under Device parameters



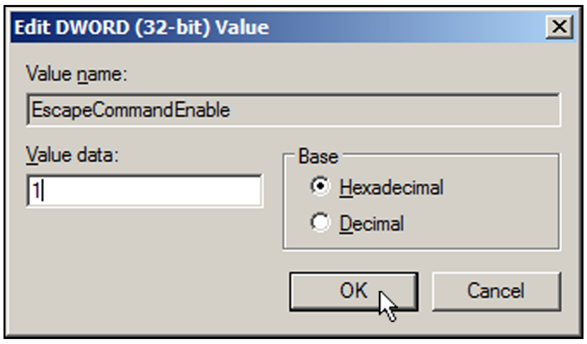
```
VID_072F&PID_2200
  5&1353e682&0&1
    Control
    Device Parameters
    LogConf
    Properties
```

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



Name	Type	Data
(Default)	REG_SZ	(value not set)
EscapeCommandEnable	REG_DWORD	0x00000001 (1)
IfdType	REG_SZ	USB Smartcard
SymbolicName	REG_SZ	\??\USB#VID_072F
VendorName	REG_SZ	Generic

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.



Edit DWORD (32-bit) Value

Value name: EscapeCommandEnable

Value data: 1

Base: (● Hexadecimal) ○ Decimal

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

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

<< Typical APDU Command and Response Flow >>

PC	Reader	Tag
Sequences	USB Interface (12 Mbps)	RF Interface (13.56 MHz)
1. The command is sent.	Contactless Related Command → [APDU Command] e.g. `[00 84 00 00 08]` (Get Challenge)	Tag-specific Command Frame → [APDU Command] embedded in ISO14443 Frame
2. The response is received.	Contactless Related Response ← [APDU Response] e.g. `[11 22 33 44 55 66 77 88] (90 00)`	Tag-specific Response Frame ← [APDU Response] embedded in ISO14443 Frame

Appendix C. APDU Command and Response Flow for ISO 18092-Compliant Tags

Assume a TOPAZ tag is used.

<< Typical APDU Command and Response Flow >>

PC	**Reader**	**Tag**
Sequences	USB Interface (12Mbps)	RF Interface (13.56MHz)
1. The command is sent	Contactless Related Command → [Native Command] e.g. `[01 08]` (read memory address 08) or Pseudo APDU Command + [Native Command] e.g. `FF 00 00 05 D4 40 01 [01 08]`	Tag-specific Command Frame → [Native Command] embedded in ISO18092 Frame
2. The response is received	Contactless Related Response ← [Native Response] e.g. `00 (90 00)` or Pseudo APDU Response + [Native Response] e.g. `D5 41 00 [00] (90 00)`	Tag-specific Response Frame ← e.g. [Native Response] embedded in ISO18092 Frame

Appendix D. Error Codes

Error Code	Error
0x00h	**No Error**
0x01h	Time Out, the target has not answered
0x02h	A CRC error has been detected by the contactless UART
0x03h	A Parity error has been detected by the contactless UART
0x04h	During a Mifare anti-collision/select operation, an erroneous Bit Count has been detected
0x05h	Framing error during Mifare operation
0x06h	An abnormal bit-collision has been detected during bit wise anti-collision at 106 kbps
0x07h	Communication buffer size insufficient
0x08h	RF Buffer overflow has been detected by the contactless UART (bit BufferOvfl of the register `CL_ERROR`)
0x0Ah	In active communication mode, the RF field has not been switched on in time by the counterpart (as defined in NFCIP-1 standard)
0x0Bh	RF Protocol error (cf. reference [4], description of the `CL_ERROR` register)
0x0Dh	Temperature error: the internal temperature sensor has detected overheating, and therefore has automatically switched off the antenna drivers
0x0Eh	Internal buffer overflow
0x10h	Invalid parameter (range, format, …)
0x12h	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: `ATR_REQ`, `WUP_REQ`, `PSL_REQ`, `DEP_REQ`, `DSL_REQ`, `RLS_REQ`, ref. [1]).
0x13h	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: • Bad length of RF received frame, • Incorrect value of PCB or PFB, • Invalid or unexpected RF received frame, • NAD or DID incoherence.
0x14h	Mifare: Authentication error
0x23h	ISO/IEC 14443-3: UID Check byte is wrong
0x25h	DEP Protocol: Invalid device state, the system is in a state which does not allow the operation
0x26h	Operation not allowed in this configuration (host controller interface)
0x27h	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, …)
0x29h	The chip configured as target has been released by its initiator
0x2Ah	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
0x2Bh	ISO/IEC 14443-3B only: the card previously activated has disappeared.
0x2Ch	Mismatch between the NFCID3 initiator and the NFCID3 target in DEP 212/424 kbps passive.
0x2Dh	An over-current event has been detected
0x2Eh	NAD missing in DEP frame

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"

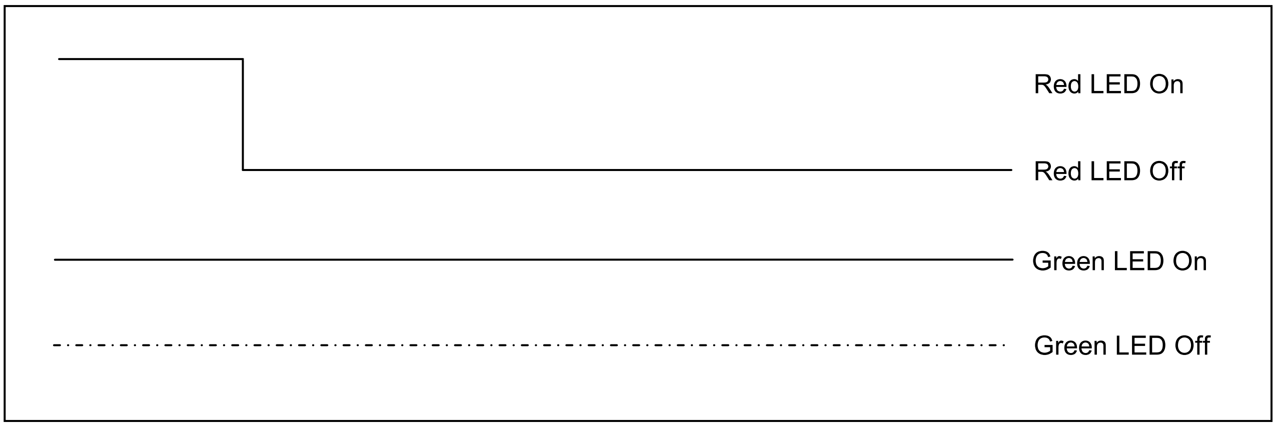
Example 3: To turn off the RED Color LED only, and leave the Green Color LED unchanged.

// Assume both Red and Green LEDs are ON initially //

// Not link to the buzzer //

APDU = "FF 00 40 04 04 00 00 00 00h"

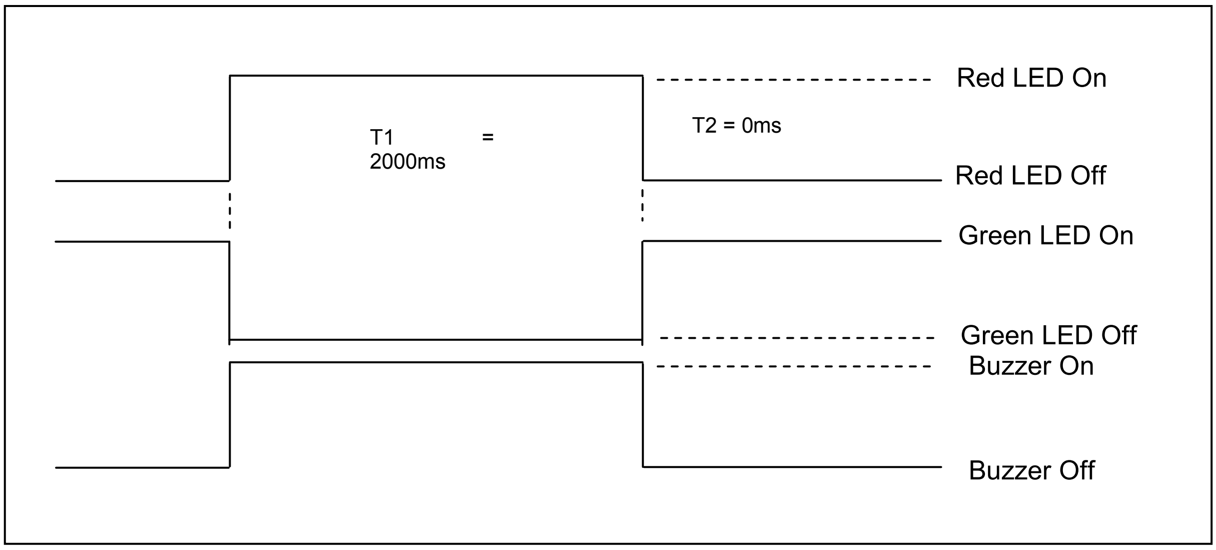
Response = "90 02h". Green LED is not changed (ON); Red LED is OFF,



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. //



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

T1 Duration = 2000ms = 0x14h

T2 Duration = 0ms = 0x00h

Number of repetition = 0x01h

Link to Buzzer = 0x01h

APDU = “FF 00 40 50 04 14 00 01 01h”

Response = “90 02h”

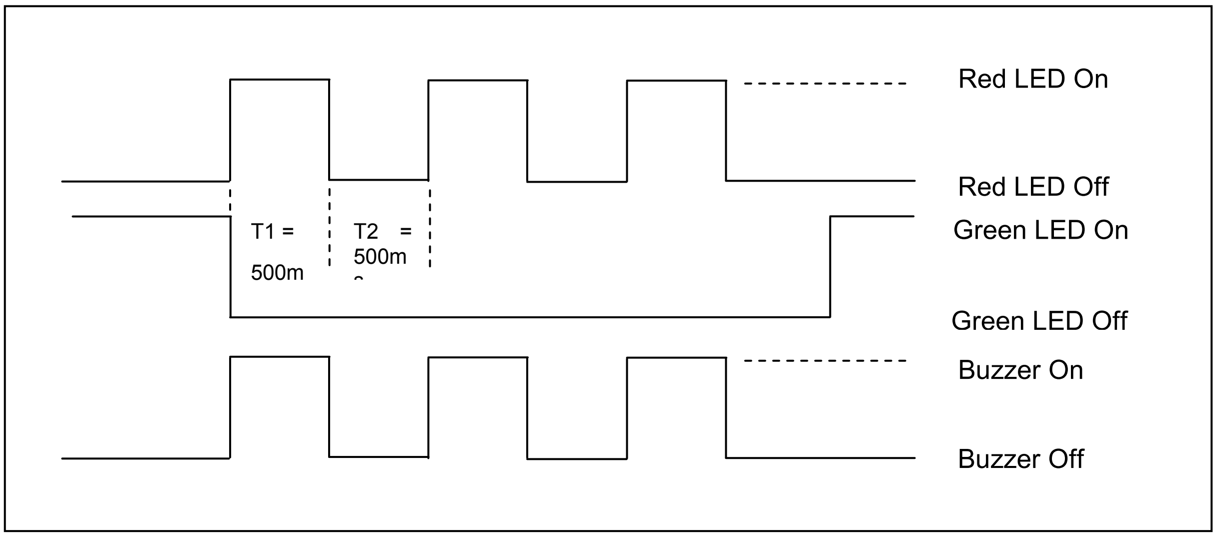
Example 5: To make the Red LED blink at 1 Hz, three times. After which, it resumes to initial state.

// 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 //



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

T1 Duration = 500ms = 0x05h

T2 Duration = 500ms = 0x05h

Number of repetition = 0x03h

Link to Buzzer = 0x01h

APDU = “FF 00 40 50 04 05 05 03 01h”

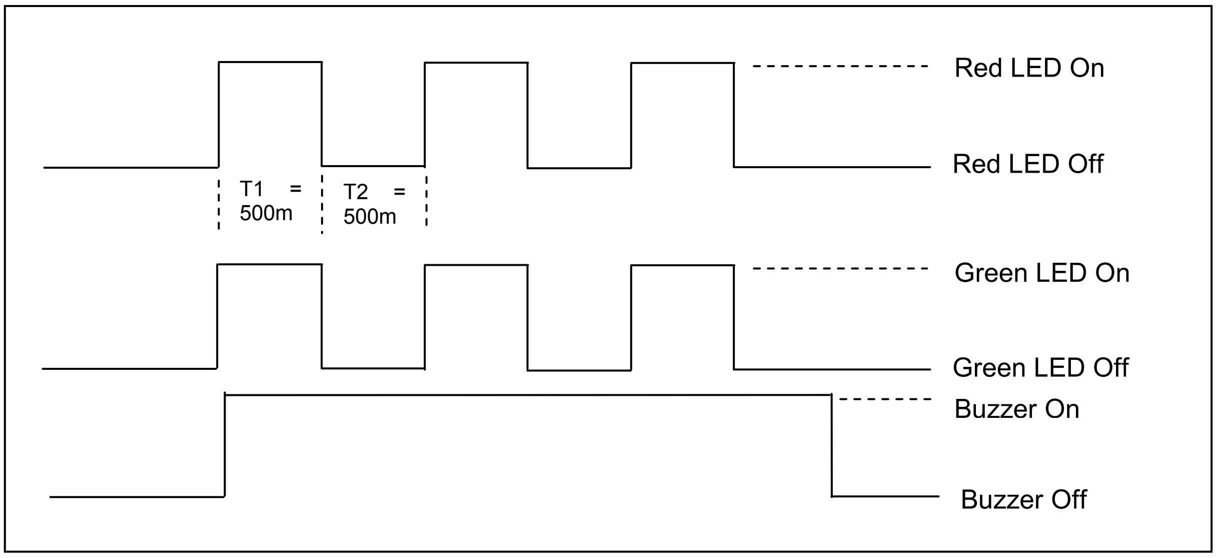
Response = “90 02h”

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//



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

T1 Duration = 500ms = 0x05h

T2 Duration = 500ms = 0x05h

Number of repetition = 0x03h

Link to Buzzer = 0x03h

APDU = “FF 00 40 F0 04 05 05 03 03h”

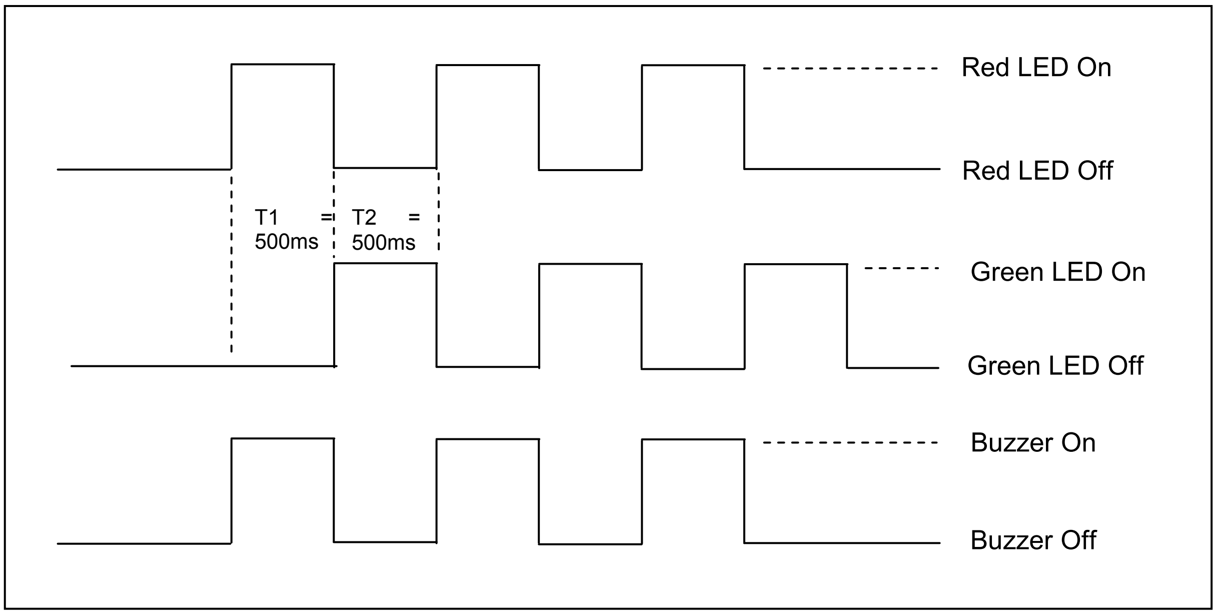
Response = “90 00h”

Example 7: To make Red and Green LED blink in turns at 1Hz 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//



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

T1 Duration = 500ms = 0x05h

T2 Duration = 500ms = 0x05h

Number of repetition = 0x03h

Link to Buzzer = 0x01h

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