



Datenblatt / Specifications



ACR122 NFC Contactless Smart Card Reader

Table of Contents

1. Introduction	3
1.1. USB Interface	3
2. Implementation	4
2.1. Smart Card Reader Interface Overview	5
3. PICC Interface Description	6
3.1. ATR Generation	6
3.1.1. ATR format for ISO 14443 Part 3 PICCs.....	6
3.1.2. ATR format for ISO 14443 Part 4 PICCs.....	7
4. PICC Commands for General Purposes	8
4.1. Get Data	8
5. PICC Commands (T=CL Emulation) for Mifare Classic Memory Cards	9
5.1. Load Authentication Keys	9
5.2. Authentication	9
5.3. Read Binary Blocks	13
5.4. Update Binary Blocks	14
5.5. Value Block Related Commands	15
5.5.1. Value Block Operation.....	15
5.5.2. Read Value Block.....	16
5.5.3. Restore Value Block.....	17
6. Pseudo-APDUs	18
6.1. Direct Transmit	18
6.2. Bi-Color LED and Buzzer Control	19
6.3. Get the Firmware Version of the reader	20
6.4. Get the PICC Operating Parameter	21
6.5. Set the PICC Operating Parameter	21
6.6. Set Timeout Parameter	22
6.7. Set Buzzer Output Enable for Card Detection	22
7. Basic Program Flow for Contactless Applications	23
7.1. How to Access PC/SC-Compliant Tags (ISO 14443-4)?	25
7.2. How to Access DESFire Tags (ISO 14443-4)?	26
7.3. How to Access FeliCa Tags (ISO 18092)?	27
7.4. How to Access NFC Forum Type 1 Tags (ISO 18092), e.g. Jewel and Topaz Tags?	28
8. Appendix E. Sample Codes for Setting the LED	30

1. 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. 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	VBUS	+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. 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 1 for more details.

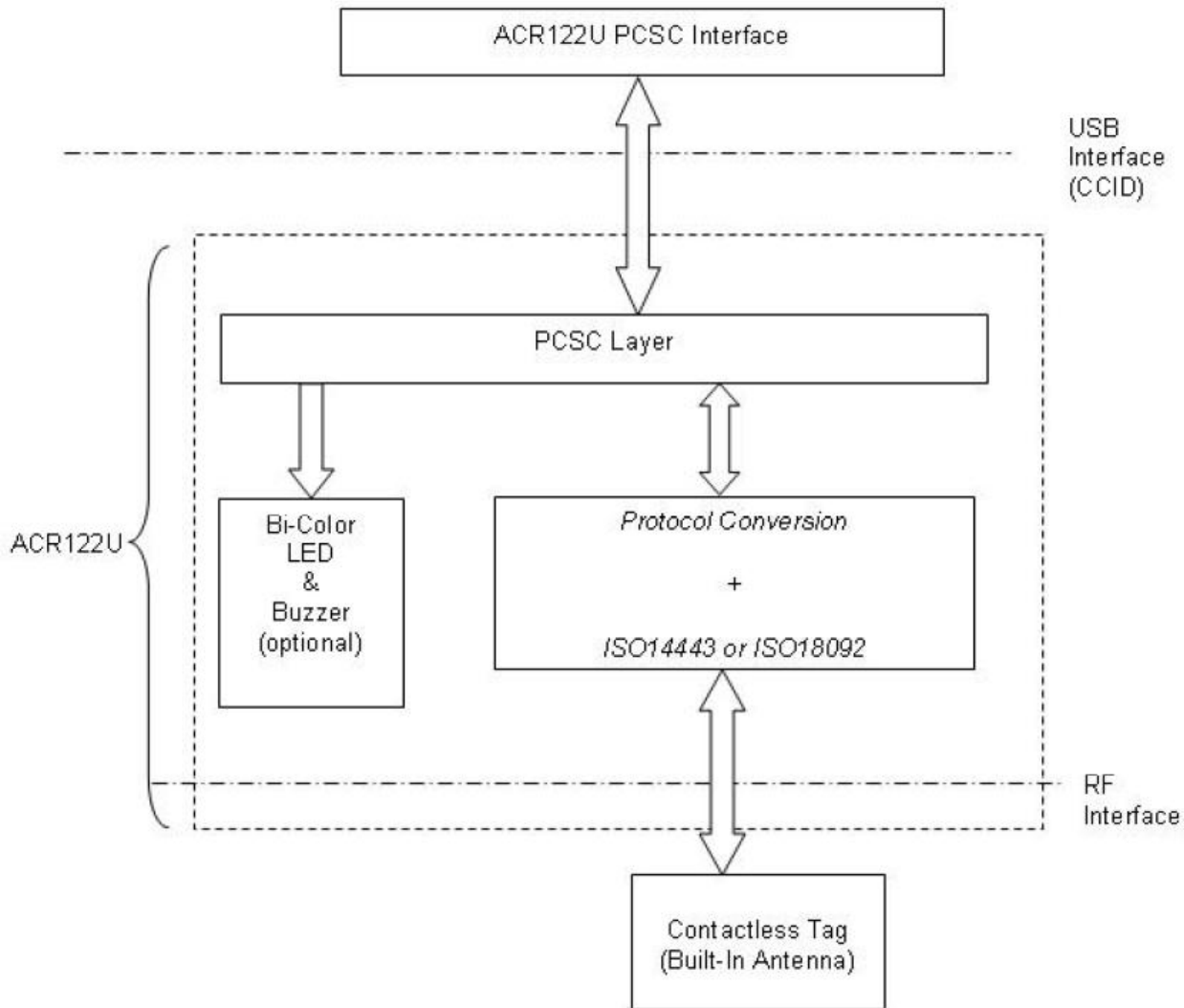


Figure 1: Communication Flow Chart of ACR122U

2.1. 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. 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	3B	Initial Header	
1	8N	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	80	TD1	Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following. Lower nibble 0 means T = 0
3	01	TD2	Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1
4 To 3+N	80	T1	Category indicator byte, 80 means A status indicator may be present in an optional COMPACT-TLV data object
	4F	TK	Application identifier Presence Indicator
	0C		Length
	RID		Registered Application Provider Identifier (RID) # A0 00 00 03 06
	SS		Byte for standard
	C0 .. C1		Bytes for card name
	00 00 00 00		RFU
4+N	UU	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 0000 00 6A}

ATR											
Initial Header	T0	TD1	TD2	T1	Tk	Length	RID	Standard	Card Name	RFU	TCK
3B	8F	80	01	80	4F	0C	A0 00 00 03 06	03	00 01	00 00 00 00	6A

Where: **Length (YY)** = 0C
RID = A0 00 00 03 06 (PC/SC Workgroup)
Standard (SS) = 03 (ISO14443A, Part 3)
Card Name (C0 .. C1) = [00 01] (Mifare 1K)

Where, Card Name (C0 .. C1)
 00 01: Mifare 1K
 00 02: Mifare 4K
 00 03: Mifare Ultralight
 00 26: Mifare Mini

 F0 04: Topaz and Jewel
 F0 11: FeliCa 212K
 F0 12: Felica 424K
 ...FF [SAK]: Undefined

3.1.2. ATR format for ISO 14443 Part 4 PICCs

Byte	Value (Hex)	Designation	Description
0	3B	Initial Header	
1	8N	T0	Higher nibble 8 means: no TA1, TB1, TC1 only TD1 is following. Lower nibble N is the number of historical bytes (HistByte 0 to HistByte N-1)
2	80	TD1	Higher nibble 8 means: no TA2, TB2, TC2 only TD2 is following. Lower nibble 0 means T = 0
3	01	TD2	Higher nibble 0 means no TA3, TB3, TC3, TD3 following. Lower nibble 1 means T = 1
4 To 3+N	xx xx xx xx	T1 Tk	Historical Bytes: ISO14443A: The historical bytes from ATS response. Refer to the ISO14443-4 specification. ISO14443B: The higher layer response from the ATTRIB response (ATQB). Refer to the ISO14443-3 specification.
4+N	UU	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 00

ATR						
Initial Header	T0	TD1	TD2	ATS		TCK
				TD1	Tk	
3B	86	80	01	06	75 77 81 02 80	00

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

Note: Use the APDU "FF CA 01 00 00" to distinguish the ISO14443A-4 and ISO14443B-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 55

ATR						
Initial Header	T0	TD1	TD2	ATQB		
				T11	Tk	TCK
3B	86	80	01	50	12 23 45 56 12 53 54 4E 33 81 C3	55

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 C3 is 12 bytes long with no CRC-B

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

4. 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	FF	CA	00 01	00	00 (Full Length)

Get UID Response Format (UID + 2 Bytes) if P1 = 0x00

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

Get ATS of a ISO 14443 A card (ATS + 2 Bytes) if P1 = 0x01

Response	Data Out		
Result	ATS	SW1	SW2

Response Codes

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

Example:

- To get the serial number of the “connected PICC”
UINT8 GET_UID[5]={0xFF, 0xCA, 0x00, 0x00, 0x04};
- To get the ATS of the “connected ISO 14443 A PICC”
UINT8 GET_ATS[5]={0xFF, 0xCA, 0x01, 0x00, 0x04};

5. 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	FF	82	Key Structure	Key Number	06	Key (6 bytes)

Where:

- **Key Structure:** 1 Byte.
0x00 = Key is loaded into the reader volatile memory.
Other = Reserved.
- **Key Number:** 1 Byte.
0x00 ~ 0x01 = 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 FF}

Load Authentication Keys Response Format (2 Bytes)

Response	Data Out	
Result	SW1	SW2

Response Codes

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

Example:

Load a key {FF FF FF FF FF FF} into the key location 0x00.
APDU = {FF 82 00 00 06 FF FF FF FF FF}

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	Lc	Data In
Authentication	FF	86	00	00	05	Authenticate Data Bytes

Authenticate Data Bytes (5 Bytes)

Byte1	Byte 2	Byte 3	Byte 4	Byte 5
Version 0x01	0x00	Block Number	Key Type	Key Number

Where:

- **Block Number:** 1 Byte. This is the memory block to be authenticated.
- **Key Type:** 1 Byte
0x60 = Key is used as a TYPE A key for authentication.
0x61 = Key is used as a TYPE B key for authentication.
- **Key Number:** 1 Byte
0x00 ~ 0x01 = Key Location.

Note: For Mifare 1K Card, it has totally 16 sectors and each sector consists of 4 consecutive blocks. E.g. Sector 0x00 consists of Blocks {0x00, 0x01, 0x02 and 0x03}; Sector 0x01 consists of Blocks {0x04, 0x05, 0x06 and 0x07}; the last sector 0x0F consists of Blocks {0x3C, 0x3D, 0x3E and 0x3F}.

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	00	The operation completed successfully.
Error	63	00	The operation failed.

Memory Map

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 ~ 0x02	0x03
Sector 1	0x04 ~ 0x06	0x07
...		
...		
Sector 14	0x38 ~ 0x0A	0x3B
Sector 15	0x3C ~ 0x3E	0x3F

Mifare 4K 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 ~ 0x02	0x03
Sector 1	0x04 ~ 0x06	0x07
...		
...		
Sector 30	0x78 ~ 0x7A	0x7B
Sector 31	0x7C ~ 0x7E	0x7F

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 ~ 0x8E	0x8F
Sector 33	0x90 ~ 0x9E	0x9F
...		
...		
Sector 38	0xE0 ~ 0xEE	0xEF
Sector 39	0xF0 ~ 0xFE	0xFF

Mifare Ultralight 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

Example:

1. To authenticate the Block 0x04 with a {TYPE A, key number 0x00}. For PC/SC V2.01, Obsolete.
APDU = {FF 88 00 04 60 00};
2. To authenticate the Block 0x04 with a {TYPE A, key number 0x00}. For PC/SC V2.07
alaAPDU = {FF 86 00 00 05 01 00 04 60 00}

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	FF	B0	00	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

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.	00	The operation completed successfully
Error	63	00	The operation failed.

Example:

1. Read **16 bytes** from the binary block **0x04** (Mifare 1K or 4K)
APDU = {FF B0 00 04 10}
2. Read **4 bytes** from the binary Page **0x04** (Mifare Ultralight)
APDU = {FF B0 00 04 04}
3. Read **16 bytes** starting from the binary Page **0x04** (Mifare Ultralight) (Pages 4, 5, 6 and 7 will be read)
APDU = {FF B0 00 04 10}

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	FF	D6	00	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	00	The operation completed successfully
Error	63	00	The operation failed.

Example:

1. Update the binary block 0x04 of Mifare 1K/4K with Data {00 01 .. 0F}
APDU = {FF D6 00 04 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F}
2. Update the binary block 0x04 of Mifare Ultralight with Data {00 01 02 03}
APDU = {FF D6 00 04 04 00 01 02 03}

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	FF	D7	00	Block Number	05	VB_OP	VB_Value (4 Bytes) {MSB .. LSB}

Where:

- Block Number:** 1 Byte. The value block to be manipulated.
 VB_OP: 1 Byte.
 0x00 = Store the VB_Value into the block. The block will then be converted to a valueblock.
 0x01 = Increment the value of the value block by the VB_Value. This command is only valid for value block.
 0x02 = 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 = {0xFF, 0xFF, 0xFF, 0xFC}

VB_Value			
MSB			LSB
FF	FF	FF	FC

Example 2: Decimal 1 = {0x00, 0x00, 0x00, 0x01}

VB_Value			
MSB			LSB
00	00	00	01

Value Block Operation Response Format (2 Bytes)

Response	Data Out	
Result	SW1	SW2

Response Codes

Results	SW1	SW2	Meaning
Success	90	00	The operation completed successfully.
Error	63	00	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	FF	B1	00	Block Number	04

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 (4bytes).

Example 1: Decimal -4 = {0xFF, 0xFF, 0xFF, 0xFC}

Value			
MSB			LSB
FF	FF	FF	FC

Example 2: Decimal 1 = {0x00, 0x00, 0x00, 0x01}

Value			
MSB			LSB
00	00	00	01

Response Codes

Results	SW1	SW2	Meaning
Success	90	00	The operation completed successfully.
Error	63	00	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	FF	D7	00	Source Block Number	02	03	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	00	The operation completed successfully.
Error	63	00	The operation failed.

Example:

1. Store a value “1” into block 0x05
APDU = {FF D7 00 05 05 00 00 00 01}
Answer: 90 00
2. Read the value block 0x05
APDU = {FF B1 00 05 00}
Answer: 00 00 00 01 90 00 [9000]
3. Copy the value from value block 0x05 to value block 0x06
APDU = {FF D7 00 05 02 03 06}
Answer: 90 00 [9000]
4. Increment the value block 0x05 by “5”
APDU = {FF D7 00 05 05 01 00 00 00 05}
Answer: 90 00 [9000]

6. 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	0xFF	0x00	0x00	0x00	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	0xFF	0x00	0x40	LED State Control	0x04	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

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.
 0x00: The buzzer will not turn on
 0x01: The buzzer will turn on during the T1 Duration
 0x02: The buzzer will turn on during the T2 Duration
 0x03: 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	90	Current LED State	The operation completed successfully.
Error	63	00	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 4 Current LED State (1 Byte)

Note:

- A. The LED State operation will be performed after the LED Blinking operation is completed.
- B. The LED will not be changed if the corresponding LED Mask is not enabled.
- C. 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.
- D. 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).
- E. To control the buzzer only, just set the P2 "LED State Control" to zero.
- F. To make the buzzer operating, the "number of repetition" must be greater than zero.
- G. 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	0xFF	0x00	0x48	0x00	0x00

Response Format (10 Bytes)

Response	Data Out
Result	Firmware Version

E.g. Response = 41 43 52 31 32 32 55 32 30 31 (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	0xFF	0x00	0x50	0x00	0x00

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	0xFF	0x00	0x51	New PICC Operating Parameter	0x00

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	ISO14443 Type B		1 = Detect 0 = Skip
0	ISO14443 Type A #To detect the Mifare Tags, the Auto ATS Generation must be disabled first.		1 = Detect 0 = Skip

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	0xFF	0x00	0x41	Timeout Parameter (Unit: 5 sec.)	0x00

Where:

P2: Timeout Parameter.

- 0x00: No Timeout check
- 0x01 – 0xFE: Timeout with 5 second unit
- 0xFF: Wait until the contactless chip responds

Response Format (8 Bytes)

Results	SW1	SW2	Meaning
Success	90	00	The operation completed successfully.
Error	63	00	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	0xFF	0x00	0x52	PollBuzzStatus	0x00

Where:

P2: PollBuzzStatus.

- 0x00: Buzzer will NOT turn ON when a card is detected
- 0xFF: 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. 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 1 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.

NOTE:

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 00
 - Turn on the antenna power: FF 00 00 00 04 D4 32 01 01
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.

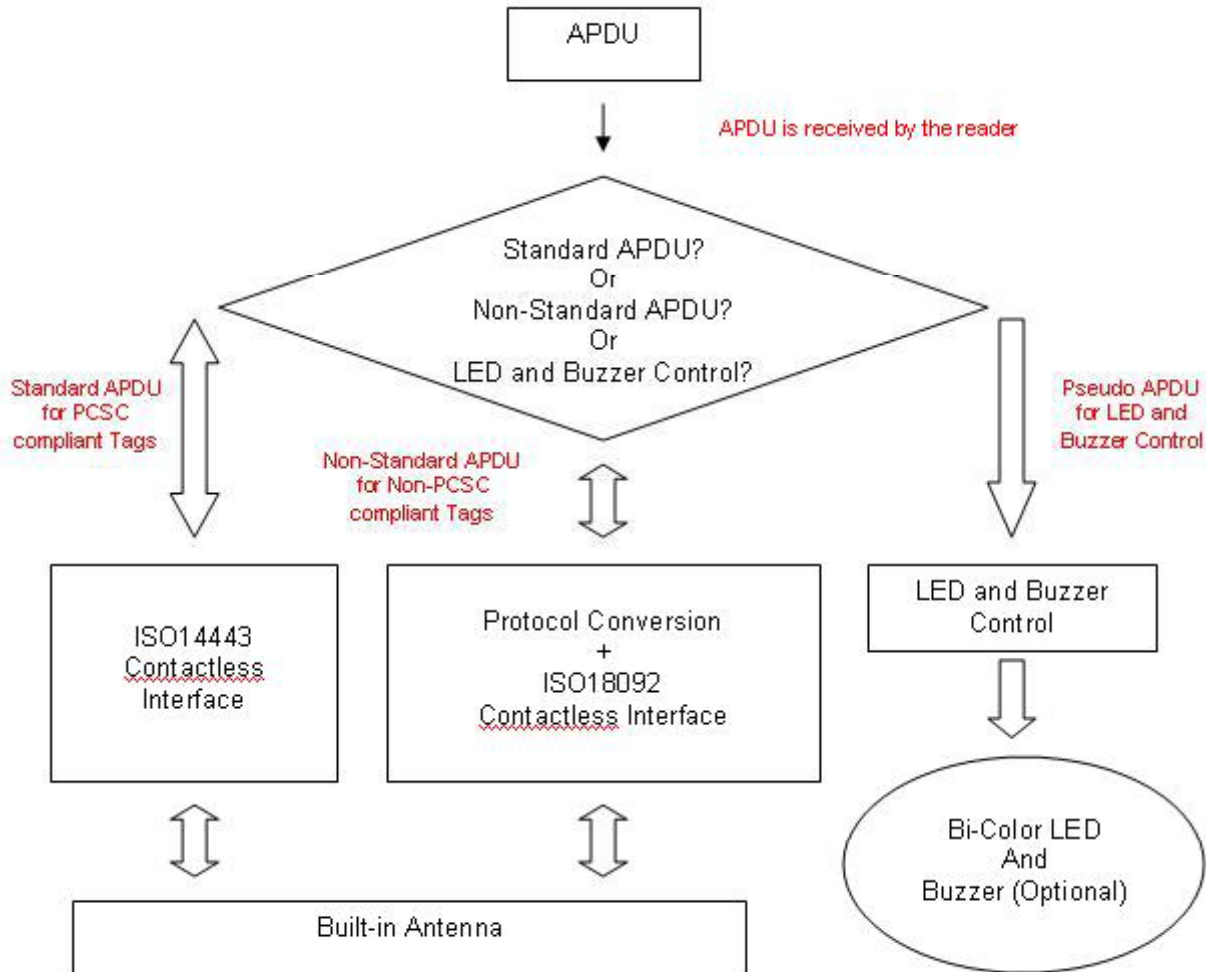


Figure 2 Basic Program Flow for Contactless Applications

- 1) For the ACR122U PICC Interface, ISO7816 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	00	The operation completed successfully.
Error	63	00	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 08

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

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

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 00}

Class = 0x90; INS = 0x0A (DESFire Instruction); P1 = 0x00; P2 = 0x00

Lc = 0x01; Data In = 0x00; Le = 0x00 (Le = 0x00 for maximum length)

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

The Status Code [91 AF] 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 00} to get the first frame. INS=0x60

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

Step 2: Send an APDU {90 AF 00 00 00} to get the second frame. INS=0xAF

Answer: 04 01 01 00 06 18 05 91 AF [\$91AF]

Step 3: Send an APDU {90 AF 00 00 00} to get the last frame. INS=0xAF

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

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 00}

Answer: AF 25 9C 65 0C 87 65 1D D7[\$1DD7]

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 {60} to get the first frame. INS=0x60

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

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

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

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

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

Note: In DESFire Native Mode, the status code [90 00] 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 00] 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:

- Present the FeliCa Tag and Connect the PICC Interface
- 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 8A

In which,

F0 11 = FeliCa 212K

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

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

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

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 00

In which,

[13] is the length of the Pseudo Data "D4 40 01.. 80 00"

D4 40 01 is the Data Exchange Command

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

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

Note:

The **NFC ID** can be obtained by using the APDU "FF CA 00 00 00"

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:

- Present the Topaz Tag and Connect the PICC Interface
- 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 9F

In which, F0 04 = Topaz

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

<< 01 08

>> 18 [90 00]

In which, Response Data = 18

Or

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

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

In which,

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

D4 40 01 is the DataExchange Command.

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

>> D5 41 00 18 [90 00]

In which, Response Data = 18

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

<< 00

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

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

<< 53 08 FF

>> FF [90 00]

In which, Response Data = FF

Topaz Memory Map.

Memory Address = Block No * 8 + Byte No

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

E.g. Memory Address 10 (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

8. Appendix E. Sample Codes for Setting the LED

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

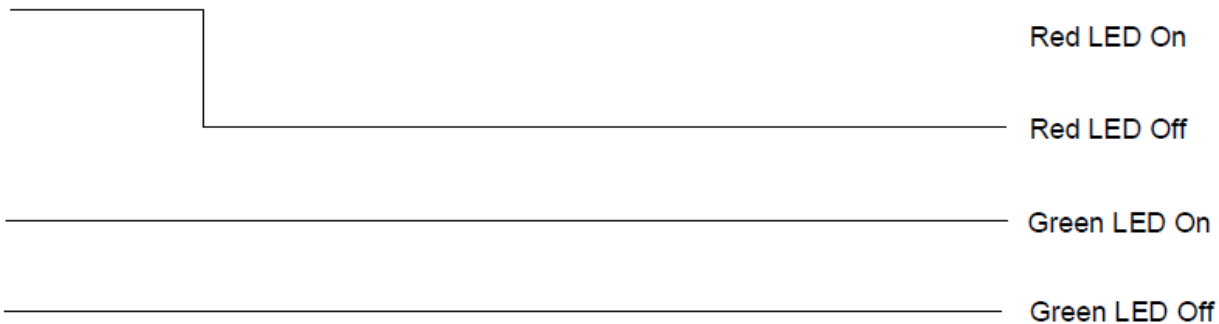
```
// Assume both Red and Green LEDs are OFF initially //
// Not link to the buzzer //
APDU = "FF 00 40 00 04 00 00 00 00"
Response = "90 00". RED and Green LEDs are OFF.
```

- **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 00"
Response = "90 03". RED and Green LEDs are ON,
To turn off both RED and Green LEDs, APDU = "FF 00 40 0C 04 00 00 00 00"
```

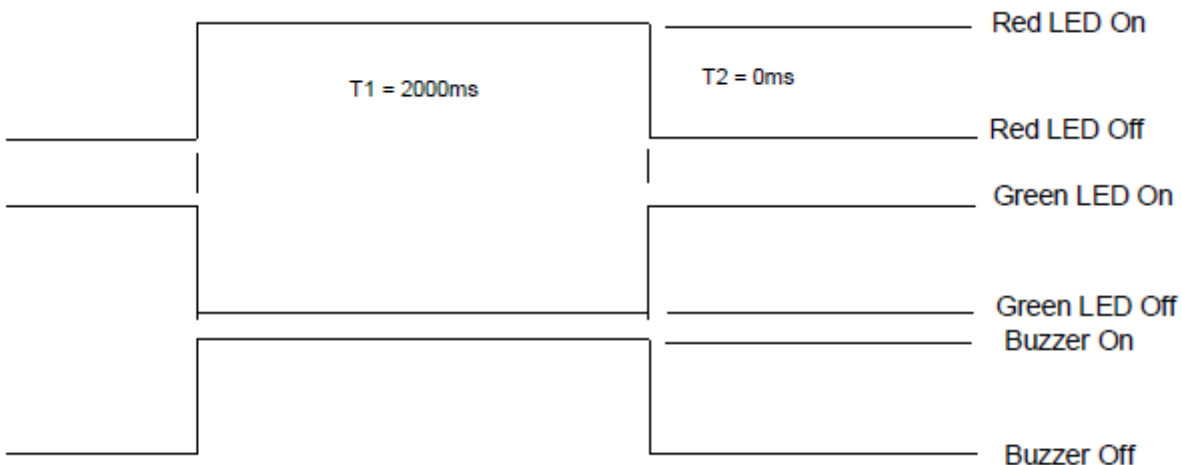
- **Example 3: To turn off the RED Color LED only, and leave the Green Color LED unchanged**

```
// Assume both Red and Green LEDs are ON initially //
// Not link to the buzzer //
APDU = "FF 00 40 04 04 00 00 00 00"
Response = "90 02". Green LED is not changed (ON); Red LED is OFF,
```



- **Example 4: To turn on the Red LED for 2 sec. 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. //
```

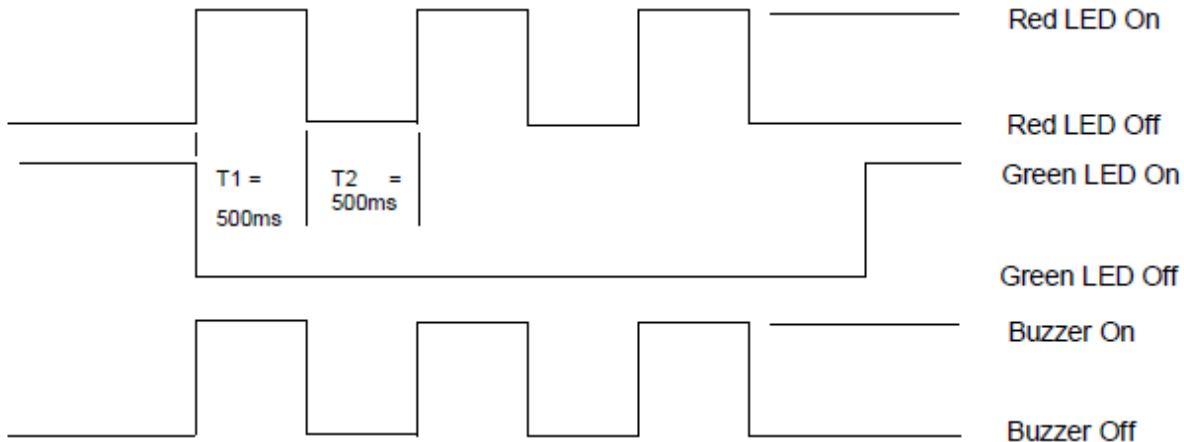


1Hz = 1000ms Time Interval = 500ms ON + 500 ms OFF
 T1 Duration = 2000ms = 0x14
 T2 Duration = 0ms = 0x00
 Number of repetition = 0x01
 Link to Buzzer = 0x01

APDU = "FF 00 40 50 04 14 00 01 01"
 Response = "90 02"

Example 5: To make the Red LED blink at 1Hz , 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 //

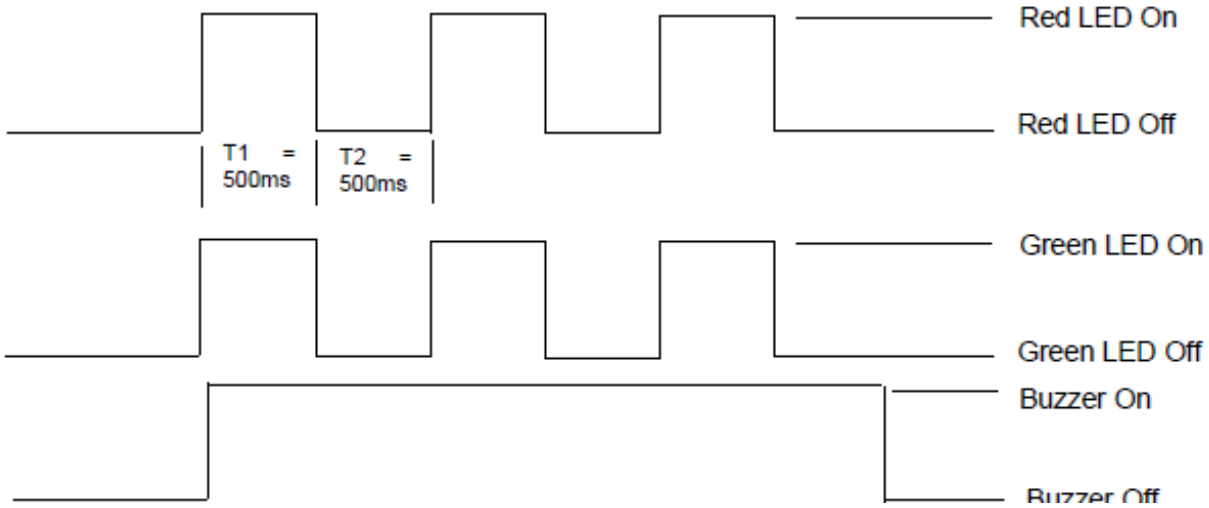


1Hz = 1000ms Time Interval = 500ms ON + 500 ms OFF
 T1 Duration = 500ms = 0x05
 T2 Duration = 500ms = 0x05
 Number of repetition = 0x03
 Link to Buzzer = 0x01

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

Example 6: To make the Red and Green LEDs blink at 1Hz 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 = 0x05
 T2 Duration = 500ms = 0x05
 Number of repetition = 0x03
 Link to Buzzer = 0x03

APDU = "FF 00 40 F0 04 05 05 03 03"
 Response = "90 00"

