Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1222L NFC Reader with LCD

Application Programming Interface

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

Table of Contents

Figures

1.0. Introduction

The ACR1222L NFC Reader with LCD is a PC-linked device that is used for accessing contactless cards. Its contactless interface is used to access ISO 14443-4 Types A and B cards, Mifare, FeliCa and ISO 18092 or NFC tags. ACR1222L also has a Secure Access Module (SAM) interface that ensures a high level of security in contactless smart card applications.

ACR1222L serves as an intermediary device between the PC and the smart card. The reader is connected to the PC via its USB port and carries out the PC's commands--whether the command is used in order to communicate with a contactless tag or SAM card, or control the device peripherals (LCD, LED or buzzer).

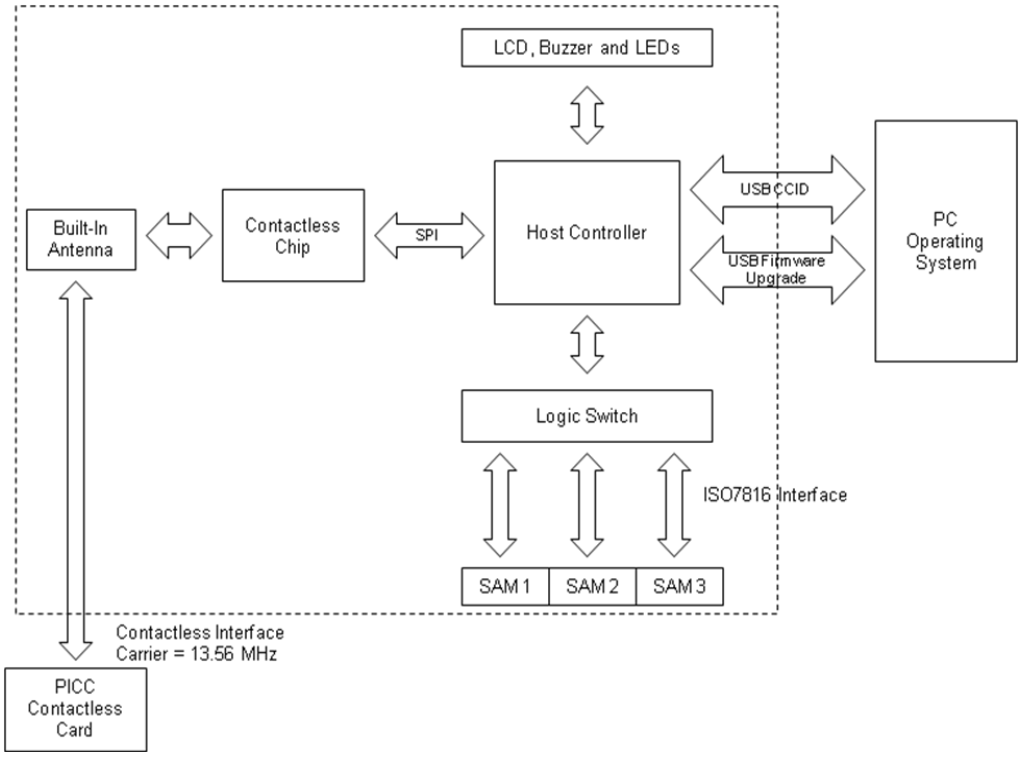
This API document provides a detailed guide on implementing PC/SC APDU commands for device peripherals and contactless tags following the PC/SC Specifications.

2.0. Features

- PC/SC for both Contactless and SAM interfaces
- CCID Compliance
- 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
- Support for new Mifare Ultralight C and DESFire EV1
- Built-in anti-collision feature (only one tag is accessed at any time)
- Two-line graphic LCD with interactive operability (i.e. scroll up and down, left and right etc) and multi-language support (i.e. Chinese, English, Japanese and some European languages)
- Three ISO 7816 compliant SAM slots
- Four User-controllable LEDs
- User-controllable buzzer
- USB Firmware Upgradability
- USB Full Speed (12 Mbps)

3.0. Architecture

3.1. Reader Block Diagram



Reader Block Diagram

3.2. Communication between the PC/SC Driver and the PICC & SAM

The protocol between the ACR1222L and the PC is using CCID protocol. All the communication between PICC and SAM are PC/SC compliant.

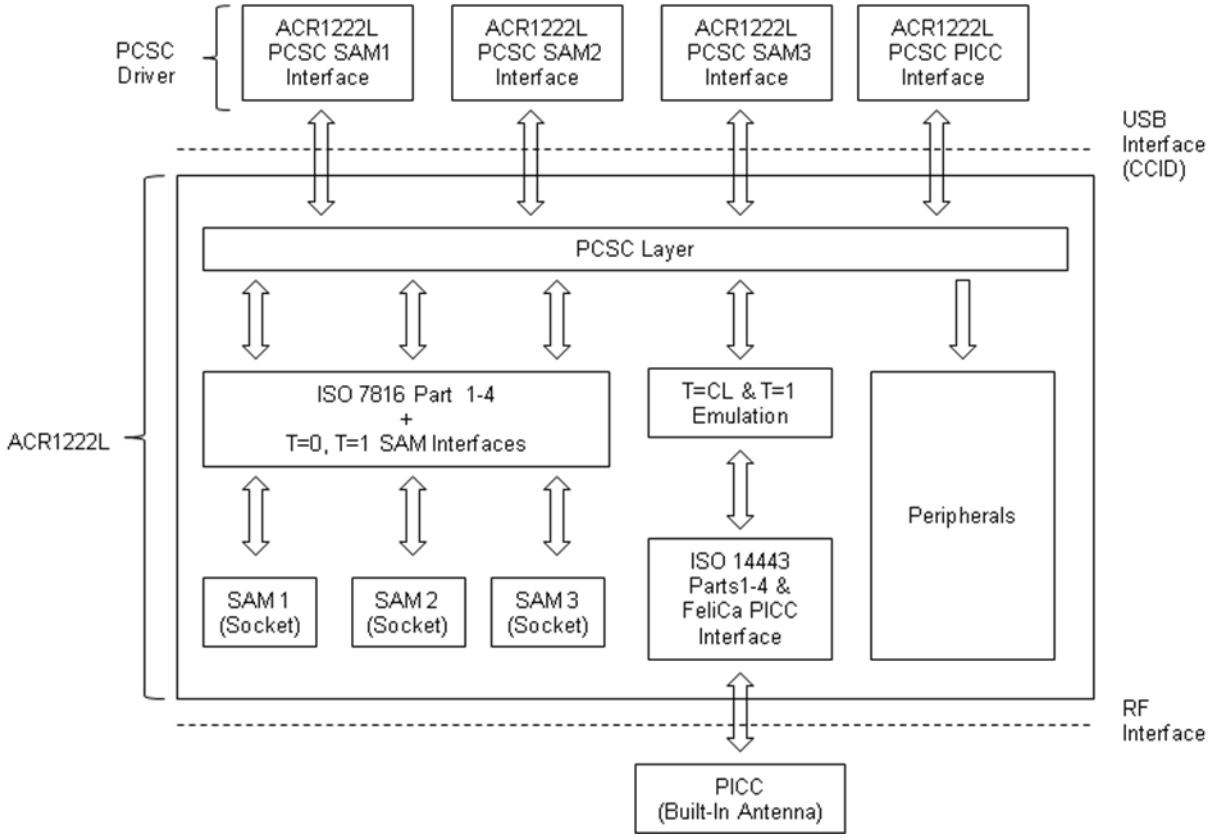


Figure 1: ACR1222L Architecture

4.0. Hardware Design

4.1. USB

The ACR1222L connects to a computer through a USB following the USB standard.

4.1.1. Communication Parameters

The ACR1222L connects to a computer through USB as specified in the USB Specification 2.0. The ACR1222L is working in full speed mode, i.e. 12 Mbps.

USB Interface Wiring

Pin	Signal	Function
1	V_{BUS}	+5V power supply for the reader
2	D-	Differential signal transmits data between ACR1222L and PC.
3	D+	Differential signal transmits data between ACR1222L and PC.
4	GND	Reference voltage level for power supply

Table 1: USB Interface Wiring

Note: In order for the ACR1222L to function properly through USB interface, the device driver should be installed.

4.1.2. Endpoints

The ACR1222L uses the following endpoints to communicate with the host computer:

Control Endpoint	For setup and control purpose
Bulk OUT	For command to send from host to ACR1222L (data packet size is 64 bytes)
Bulk IN	For response to send from ACR1222L to host (data packet size is 64 bytes)
Interrupt IN	For card status message to sent from ACR1222L to host (data packet size is 8 bytes)

4.2. Contact Smart Card Interface

The interface between the ACR1222L and the inserted smart card follows the ISO 7816-3 specifications with certain restrictions or enhancements to increase the practical functionality of the ACR1222L.

4.2.1. Smart Card Power Supply VCC (C1)

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

4.2.2. Card Type Selection

Before activating the inserted card, the controlling PC always needs to select the card type through the proper command sent to the ACR1222L. This includes both memory card and MCU-based cards.

For MCU-based cards the reader allows to select the preferred protocol, T=0 or T=1. However, this selection is only accepted and carried out by the reader through the PPS when the card inserted in the reader supports both protocol types. Whenever a MCU-based card supports only one protocol type, T=0 or T=1, the reader automatically uses that protocol type, regardless of the protocol type selected by the application.

4.2.3. Interface for Microcontroller-based Cards

For microcontroller-based smart cards only the contacts C1 (VCC), C2 (RST), C3 (CLK), C5 (GND) and C7 (I/O) are used. A frequency of 4 MHz is applied to the CLK signal (C3).

4.3. Contactless Smart Card Interface

The interface between the ACR1222L and the Contactless follows the ISO 14443 specifications with certain restrictions or enhancements to increase the practical functionality of the ACR1222L.

4.3.1. Carrier Frequency

The carrier frequency for ACR1222L is 13.56 MHz.

4.3.2. Card Polling

The ACR1222L automatically polls the contactless cards that are within the field. ISO 14443-4 Type A, ISO 14443-4 Type B and Mifare, FeliCa and NFC tags are supported.

4.4. User Interface

4.4.1. Buzzer

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

User-controllable Monotone Buzzer.

Events	Buzzer
1. The reader powered up and initialization success.	Beep
2. Card Insertion Event (PICC)	Beep
3. Card Removal Event (PICC)	Beep

Table 1: Buzzer Event

4.4.2. LED

- 4 x user-controllable single-color LEDs
- LED colors are: Green, Blue, Orange and Red (from left to right)

5.0. Software Design

5.1. Contactless Smart Card Protocol

5.1.1. ATR Generation

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

5.1.1.1. 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	80	T1	Category indicator byte, 80 means A status indicator may be present in an optional COMPACT-TLV data object
To	4F	Tk	Application identifier Presence Indicator
3+N	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	RFU # 00 00 00 00
4+N	UU	TCK	Exclusive-oring of all the bytes T0 to Tk

Example:

ATR for Mifare 1K = {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 (ISO 14443A, Part 3)
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 28: JCOP 30
		FF [SAK]: undefined tags

5.1.1.2. 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	T1	Historical Bytes: ISO 14443A: The historical bytes from ATS response. Refer to the ISO 14443-4 specifications. ISO 14443B: The higher layer response from the ATTRIB response (ATQB). Refer to the ISO 14443-3 specifications.
	XX XX XX	Tk	
4+N	UU	TCK	Exclusive-oring of all the bytes T0 to Tk

Example 1: Consider the ATR from DESFire as follows:

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

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

APDU Command = `FF CA 01 00 00`

APDU Response = `06 75 77 81 02 90 00`

ATS = `06 75 77 81 02 80`

Example 2:

ATR for ST19XRC8E = `3B 88 80 01 12 53 54 4E 33 81 C3 00 23`

Application Data of ATQB = `12 53 54 4E`

Protocol Info of ATQB = `33 81 C3`

5.1.2. Pseudo APDUs for Contactless Interface

5.1.2.1. Direct Transmit via PC_to_RDR_XfrBlock/PC_to_RDR_Escape

This command is used to send Pseudo APDU (Contactless Chip and Tag commands), and the Response Data will be returned.

Command

Command	Class	INS	P1	P2	Lc	Data In
`Direct Transmit`	`FF`	`00`	`00`	`00`	Number of Bytes to send	Contactless Chip and Tag Command

Where:

Lc: 1 Byte. Number of Bytes to Send

- Maximum 255 bytes

Data In: Contactless Chip or Tag Command

The data to be sent to the Contactless Chip and Tag

Response

Response	Data Out	
Result	Contactless Chip and Tag Response	SW1 SW2

Where:

Contactless Chip and Tag Response

Contactless Chip and Tag Response returned by the reader.

SW1, SW2 = `90 00` means the operation is completed successfully

= `63 00` means the operation failed

= `63 27` means the checksum of the Response is wrong

5.1.2.2. Direct Transmit via PC_to_RDR_Escape

Command

Command	Class	INS	P1	P2	Lc	Data In
Direct Transmit	E0	00	00	24	Number of Bytes to send	Contactless Chip and Tag Command

Where:

Lc: 1 Byte. Number of Bytes to Send.
- Maximum of 255 bytes.

Data In: Contactless Chip and Tag Command.
The data to be sent to the Contactless Chip and Tag

Response

Response	Class	INS	P1	P2	Le	Data Out
Result	E1	00	00	00	Number of Bytes to send	Contactless Chip and Tag Response

Where:

Data Out: Contactless Chip and Tag Response.
Contactless Chip and Tag Response returned by the reader.

5.1.2.3. Get Data

This command is used to return the serial number or ATS of the “connected PICC”.

Command

Command	Class	INS	P1	P2	Le
Get Data	FF	CA	00 01	00	00 (Max Length)

Response if P1 = 00

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

Response if P1 = 01

Response	Data Out		
Result	ATS	SW1	SW2

Response Codes

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

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

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

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

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

5.1.2.4. PICC Commands (T=CL Emulation) for Mifare 1K/4K MEMORY Cards

5.1.2.4.1. Load Authentication Keys

This command is used to load the authentication keys into the reader. The authentication keys are used to authenticate the specified sector of the MIfare 1K/4K Memory Card. Volatile authentication key location is provided.

Command

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.

00 = Key is loaded into the reader volatile memory.

Other = Reserved.

Key Number: 1 Byte.

00 - 01 = Key Location

The keys will be erased when 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}

Response

Response	Data Out	
Result	SW1	SW2

Where:

SW1, SW2 = 90 00 means the operation is completed successfully

= 63 00 means 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 FF}

5.1.2.4.2. Authentication for Mifare 1K/4K

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

Command

Command	Class	INS	P1	P2	P3	Data In
Authentication 6 Bytes (Obsolete)	FF	88	00	Block Number	Key Type	Key Number

Command

Command	Class	INS	P1	P2	Lc	Data In
Authentication 10 Bytes	FF	86	00	00	05	Authenticate Data Bytes

Where:

Authenticate Data Bytes: 5 Bytes.

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

Block Number: 1 Byte.

The memory block to be authenticated.

Note: For Mifare 1K Card, it has a total of 16 sectors and each sector consists of 4 consecutive blocks. For example, Sector 00 consists of Blocks {00, 01, 02 and 03}; Sector 01 consists of Blocks {04, 05, 06 and 07}; the last sector 0F consists of Blocks {3C, 3D, 3E and 3F}.

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

Key Type: 1 Byte.

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

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

Key Number: 1 Byte.

`00` ~ `01` = Key Location.

Response Format

Response	**Data Out**	
Result	SW1	SW2

Where:

SW1, SW2 = `90 00` means the operation is completed successfully

= `63 00` means the operation failed

Mifare 1K 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`

1K Bytes

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`

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

2K Bytes

Example1: To authenticate Block 04 with the following characteristics: Type A, key number 00, from PC/SC V2.01 (Obsolete).

APDU = { FF 88 00 04 60 00 }

Example2: Similar to the previous example, to authenticate Block 04 with the following characteristics: Type A, key number 00, from PC/SC V2.07.

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

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

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

512 bits
Or
64 bytes

5.1.2.4.3. Read Binary Blocks

This command is used to retrieve multiple “data blocks” from the PICC. The data block/trailer must be authenticated first before executing the “Read Binary Blocks” command.

Command

Command	Class	INS	P1	P2	Le
Read Binary Blocks	FF	B0	00	Block Number	Number of Bytes to Read

Where:

Block Number: 1 Byte. Starting Block.

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

Maximum of 16 bytes for Mifare Ultralight.

Maximum of 48 bytes for Mifare 1K. (Multiple Blocks Mode; 3 consecutive blocks)

Maximum of 240 bytes for Mifare 4K. (Multiple Blocks Mode; 15 consecutive blocks)

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

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

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

Response

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

Where:

SW1, SW2 = 90 00 means the operation is completed successfully

= 63 00 means the operation failed

Example 1: Read 16 bytes from the binary block 04 (Mifare 1K or 4K)

APDU = { FF B0 00 04 10 }

Example 2: Read 240 bytes starting from the binary block 80 (Mifare 4K). Block 80_H to Block $8E_H$ (15 blocks)

APDU = { FF B0 00 80 F0 }

5.1.2.4.4. Update Binary Blocks

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

Command

Command	Class	INS	P1	P2	Lc	Data In
Update Binary Blocks	FF	D6	00	Block Number	Number of Bytes to Update	Block Data (Multiple of 16 Bytes)

Where:

Block Number: 1 Byte. Starting Block.

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

Maximum of 16 bytes for Mifare Ultralight.

Maximum of 48 bytes for Mifare 1K. (Multiple Blocks Mode; 3 consecutive blocks)

Maximum of 240 bytes for Mifare 4K. (Multiple Blocks Mode; 15 consecutive blocks)

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

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

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

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

Response

Response	Data Out	
Result	SW1	SW2

Where:

SW1, SW2 = 90 00 means the operation is completed successfully

= 63 00 means the operation failed

Example 1: Update the binary block 04 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 }

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

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

5.1.2.4.5. Value Block Operation (Increment, Decrement, Store)

This command is used to manipulate value-based transactions (e.g. increment a value block, etc.).

Command

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. Value Block to be manipulated

VB_OP: 1 Byte. Value block operation.

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

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

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

VB_Value: 4 Byte. The value used for manipulation. The value is a signed long integer.

Example 1: Decimal - 4 = { FF FF FF FC }

VB_Value			
MSB			LSB
FF	FF	FF	FC

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

VB_Value			
MSB			LSB
00	00	00	01

Response

Response	Data Out	
Result	SW1	SW2

Where:

SW1, SW2 = 90 00 means the operation is completed successfully

= 63 00 means the operation failed

5.1.2.4.6. Read Value Block

This command is used to retrieve the value from the value block. This command is only valid for value blocks.

Command

Command	Class	INS	P1	P2	Le
Read Value Block	FF	B1	00	Block Number	00

Where:

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

Response

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

Response

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

Where:

Value. 4 Bytes. The value returned from the cards. The value is a signed long integer.

Example 1: Decimal - 4 = { FF FF FF FC }

VB_Value			
MSB			LSB
FF	FF	FF	FC

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

VB_Value			
MSB			LSB
00	00	00	01

Response

Response	Data Out	
Result	SW1	SW2

Where:

SW1, SW2 = 90 00 means the operation is completed successfully

= 63 00 means the operation failed

5.1.2.4.7. Copy Value Block

This command is used to copy a value from a value block to another value block.

Command

Command	Class	INS	P1	P2	Lc	Data In	
Value Block Operation	FF	D7	00	Source Block Number	02	03	Target Block Number

Where:

Source Block Number: 1 Byte. Block number where the value will come from and copied to the target value block.

Target Block Number: 1 Byte. Block number where the value from the source block will be copied to. The source and target value blocks must be in the same sector.

Response

Response	Data Out	
Result	SW1	SW2

Where:

SW1, SW2 = 90 00 means the operation is completed successfully

= 63 00 means the operation failed

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

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

Example 2: Read the value block 05

APDU = {FF B1 00 05 00 }

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

APDU = {FF D7 00 05 02 03 06 }

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

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

5.1.2.5. Access PC/SC Compliant Tags (ISO 14443-4)

Basically, all ISO 14443-4 compliant cards (PICCs) would understand the ISO 7816-4 APDUs. The ACR1222L Reader needs to communicate with the ISO 14443-4 compliant cards through exchanging ISO 7816-4 APDUs and Responses. ACR1222U will handle the ISO 14443 Parts 1-4 Protocols internally.

The 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".

Command

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

Response

Response	Data Out		
Result	Response Data	SW1	SW2

Where:

SW1, SW2 = 90 00 means the operation is completed successfully

= 63 00 means 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**

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

In which,

The Application Data of ATQB = 00 00 00 00, protocol information of ATQB = 33 81 81. It is an ISO14443-4 Type B tag.

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

<< 00 84 00 00 08

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

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

Example: ISO 7816-4 APDU

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

APDU = { 80 B2 80 00 08 }

Class = 80; INS = B2; P1 = 80; P2 = 00;

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

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

5.2. Peripherals Control

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

5.2.1. Get Firmware Version

This command is used to get the reader's firmware message.

Command

Command	Class	INS	P1	P2	Lc
`Get Firmware Version`	`E0`	`00`	`00`	`18`	`00`

Response

Response	Class	INS	P1	P2	Le	Data Out
Result	`E1`	`00`	`00`	`00`	Number of Bytes to Received	Firmware Version

E.g. Response = `E1 00 00 00 11 41 43 52 31 32 32 32 4C 2D 55 20 56 33 30 37 2E 31`

Firmware Version (HEX) = `41 43 52 31 32 32 32 4C 2D 55 20 56 33 30 37 2E 31`

Firmware Version (ASCII) = "ACR1222L-U V307.1"

OR Using ACR122U Command

Command

Command	Class	INS	P1	P2	Le
`Get Response`	`FF`	`00`	`48`	`00`	`00`

Response

Response	Data Out
Result	Firmware Version

E.g. Response = **`41 43 52 31 32 32 32 4C 2D 55 20 56 33 30 37 2E 31`** (Hex)

= ACR1222L-U V307.1 (ASCII)

5.2.2. Pseudo APDU for LEDs and Buzzer Control

This command is used to control the states of the LED_0, LED_1 and Buzzer.

Command

Command	Class	INS	P1	P2	Lc	Data In (4 Bytes)
LEDs and Buzzer Control	FF	00	40	LED State Control	04	Blinking Duration Control

Where:

P2: 1 Byte. LED State Control.

CMD	Item	Description
Bit 0	Final LED_1 State	1 = On; 0 = Off
Bit 1	Final LED_0 State	1 = On; 0 = Off
Bit 2	LED_1 State Mask	1 = Update the State 0 = No change
Bit 3	LED_0 State Mask	1 = Update the State 0 = No change
Bit 4	Initial LED_1 Blinking State	1 = On; 0 = Off
Bit 5	Initial LED_0 Blinking State	1 = On; 0 = Off
Bit 6	LED_1 Blinking Mask	1 = Blink 0 = Not Blink
Bit 7	LED_0 Blinking Mask	1 = Blink 0 = Not Blink

Data In: 4 Bytes. Blinking Duration Control.

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

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

00: The buzzer will not turn on

01: The buzzer will turn on during the T1 Duration

02: The buzzer will turn on during the T2 Duration

03: 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 is completed successfully.
Error	63	00	The operation failed.

Current LED State: 1 Byte.

Status	Item	Description
Bit 0	Current LED_1 LED	1 = On; 0 = Off
Bit 1	Current LED_0 LED	1 = On; 0 = Off
Bits 2 – 7	Reserved	Reserved

Remark:

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 blink 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.

5.2.3. Pseudo APDU for LEDs Control Enable

This command is used to set the LEDs behavior when a card is detected.

Command

Command	Class	INS	P1	P2	Lc
LED Control	FF	00	43	bLEDCtrl	00

Where:

P2: 1 Byte. bLEDCtrl.

CMD	Description
00	The LEDs will turn on when a card is detected
FF	The LEDs will not turn on when a card is detected

Data Out: SW1 SW2.

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

5.2.4. Pseudo APDU for LEDs Control

This command is used to control 4 LEDs.

Command

Command	Class	INS	P1	P2	Lc
LED Control	FF	00	44	bLEDsState	00

Where:

P2: 1 Byte. bLEDsState.

CMD	Item	Description
Bit 0	LED_0 State	1 = On; 0 = Off
Bit 1	LED_1 State	1 = On; 0 = Off
Bit 2	LED_2 State	1 = On; 0 = Off
Bit 3	LED_3 State	1 = On; 0 = Off
Bits 4 – 7	Reserved	Reserved

Data Out: SW1 SW2.

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

5.2.5. Pseudo APDU for Buzzer Control

This command is used to control the buzzer.

Command

Command	Class	INS	P1	P2	Lc	Data In (3 Bytes)
Buzzer Control	FF	00	42	00	03	Buzzer Control

Data In: 4 Bytes. Buzzer Control.

Byte 0	Byte 1	Byte 2
T1 Duration On State (Unit = 100ms)	T2 Duration Off State (Unit = 100ms)	Number of repetition

Data Out: SW1 SW2.

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

5.2.6. Get the PICC Operating Parameter

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

Command

Command	Class	INS	P1	P2	Le
Get the PICC Operating Parameter	FF	00	50	00	00

Response

Response	Data Out
Result	PICC Operating Parameter

Where:

PICC Operating Parameter. Default Value = FF

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 ISO 14443-4 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

5.2.7. Set the PICC Operating Parameter

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

Command

Command	Class	INS	P1	P2	Le
Set the PICC Operating Parameter	FF	00	51	New PICC Operating Parameter	00

Response

Response	Data Out
Result	PICC Operating Parameter

Where:

PICC Operating Parameter. Default Value = FF

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

5.2.8. Set Serial Number of the reader

This command is used to set the serial number of the reader.

Command

Command	Class	INS	P1	P2	Le	Data
Set Serial Number	E0	00	00	26	14	Sn

Response

Response	Class	INS	P1	P2	Le	Data	
Result	E1	00	00	00	02	SW1	SW2

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

5.2.9. Get Serial Number of the reader

This command is used to retrieve the serial number of the reader.

Command

Command	Class	INS	P1	P2	Le
Get Response	FF	00	49	00	00

Response

Response	Data Out
Result	Sn

5.2.10. Set Timeout Parameter

This command is used to set the time out parameter of the reader response time. The default value is 5 seconds.

Command

Command	Class	INS	P1	P2	Le
Get Response	FF	00	41	Time out Parameter (unit: 5 sec)	00

P2: Time out Parameter

00 – No time out check

01 ~ FE – Time out with 5 sec unit

FF – Wait until the reader responds

Response

Response	Data Out	
Result	SW1	SW2

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

5.2.11. Store 1st Data Storage Area

This command is used to store a data to 1st Data Storage Area (up to 256 Bytes)

Command

Command	Class	INS	P1	P2	Lc	Data		
Store 1st Data Storage	FF	00	4A	00	00	Data Len (MSB)	Data Len (LSB)	Data

Data Len (MSB): The high byte of the data length

Data Len (LSB): The low byte of the data length

Response

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

5.2.12. Store 2nd Data Storage Area

This command is used to store a data to 2nd Data Storage Area (up to 256 Bytes)

Command

Command	Class	INS	P1	P2	Lc	Data		
Store 2nd Data Storage	FF	00	4B	00	00	Data Len (MSB)	Data Len (LSB)	Data

Data Len (MSB): The high byte of the data length

Data Len (LSB): The low byte of the data length

Store 2nd Data Storage Response Format (2 bytes)

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

5.2.13. Read 1st Data Storage Area

This command is used to read a data from 1st Data Storage Area (up to 256 Bytes)

Command

Command	Class	INS	P1	P2	Lc	Data	
Read 1st Data Storage	FF	00	4C	00	00	Data Len (MSB)	Data Len (LSB)

Data Len (MSB): The high byte of the data length

Data Len (LSB): The low byte of the data length

Response

Results	Data
Result	Data return from the 1st Data Storage Area

5.2.14. Read 2nd Data Storage Area

This command is used to read a data from 2nd Data Storage Area (up to 256 Bytes)

Command

Command	Class	INS	P1	P2	Lc	Data	
Read 2nd Data Storage	FF	00	4D	00	00	Data Len (MSB)	Data Len (LSB)

Data Len (MSB): The high byte of the data length

Data Len (LSB): The low byte of the data length

Response

Results	Data
Result	Data return from the 2nd Data Storage Area

5.2.15. LCD Control Command

5.2.15.1. Clear LCD

This command is used to clear all contents shown on the LCD.

Command

Command	Class	INS	P1	P2	Lc
Clear LCD	FF	00	60	00	00

Response

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

5.2.15.2. LCD Display (ASCII Mode)

This APDU is used to display LCD message in ASCII Mode.

Command

Command	Class	INS	P1	P2	Lc	Data In (Max. 16Bytes)
LCD Display	FF	Option Byte	68	LCD XY Position	LCD Message Length	LCD Message

Where”

INS: 1 Byte. Option Byte.

CMD	Item	Description
Bit 0	Character Bold Font	1 = Bold; 0 = Normal
Bit 1 - 3	Reserved	Reserved
Bit 4 - 5	Table Index	00 = Fonts Set A 01 = Fonts Set B 10 = Fonts Set C
Bits 6 – 7	Reserved	Reserved

P2: 1 Byte. LCD XY Position.

The Character to be displayed on the LCD position specified by DDRAM Address

Please follow the DDRAM table below for the LCD character position’s representation:

For Fonts Set 1 and 2,

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	DISPLAY POSITION
1st LINE	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	LCD XY POSITION
2nd LINE	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	

For Fonts Set 3,

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	DISPLAY POSITION
1st LINE	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	LCD XY POSITION
2nd LINE	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	
3rd LINE	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	
4th LINE	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	

Lc: LCD Message Length

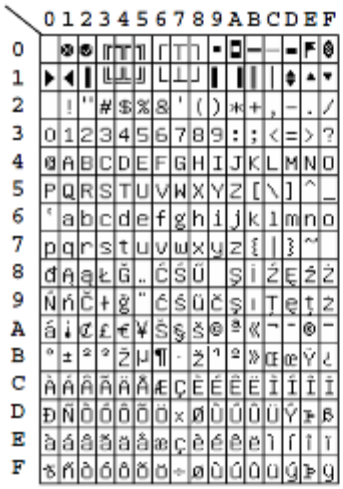
The length of the LCD message (max. 10); If the message length is longer than the number of character that the LCD screen's can be shown, then the redundant character will not be shown on the LCD

Data In: LCD Message.

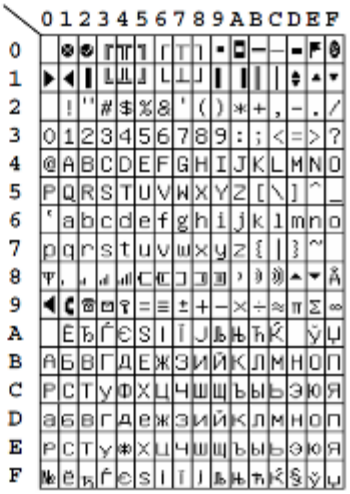
The message to be displayed on the LCD, maximum 16 Character for each line.

Please follow the Font tables (selected by INS Bit 4 - 5) below for the LCD Character Index

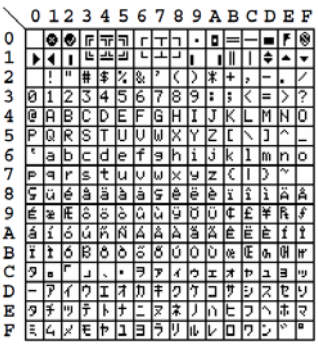
Note: Size of the Characters in Fonts Set A and Fonts Set B is 8x16, but size of the Characters in Fonts Set C is 8x8



Character Set A



Character Set B



Character Set C

Response

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

5.2.15.3. Pseudo APDU for LCD Display (GB Mode)

This command is used to display LCD message in GB Mode.

Command

Command	Class	INS	P1	P2	Lc	Data In (Max. 16 Bytes)
LCD Display	FF	Option Byte	69	LCD XY Position	LCD Message Length	LCD Message

Where:

INS: 1 Byte. Option Byte.

CMD	Item	Description
Bit 0	Character Bold Font	1 = Bold; 0 = Normal
Bit 1 - 7	Reserved	Reserved

P2: LCD XY Position

The Character to be displayed on the LCD position specified by DDRAM Address

Please follow the DDRAM table below for the LCD character position's representation:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	DISPLAY POSITION
FIRST LINE	00		01		02		03		04		05		06		07		LCD XY POSITION
SECOND LINE	40		41		42		43		44		45		46		47		

Lc: LCD Message Length

The length of the LCD message (max. 10); If the message length is longer than the number of characters that the LCD screen can show, then the redundant character will not be shown on the LCD

The length of the LCD message should multiple of 2 because each Chinese Character (GB code) should be contain two bytes

Data In: LCD Message

The data to be sent to LCD, maximum of 8 (2 x 8 bit each character) characters for each line

Please follow the Fonts table of GB Coding

Response

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

5.2.15.4.

5.2.15.5. LCD Display (Graphic Mode)

This command is used to display LCD message in Graphic Mode.

Command

Command	Class	INS	P1	P2	Lc	Data In (max. 128 Bytes)
LCD Display	FF	00	6A	Line Index	Pixel Data Length	Pixel Data

Where:

P2: Line Index.

To set which line to start to update the LCD Display

Refer to Below LCD Display Position

Lc: Pixel Data Length

The length of the pixel data (max. 0x80)

Data In: Pixel Data

The pixel data to be sent to LCD for display

LCD Display Position (Total LCD Size: 128x32):

	Byte 0x00 (X = 0x00)								Byte 0x01 (X = 0x01)								…	Byte 0x0F (X = 0x0F)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	…	7	6	5	4	3	2	1	0
0x00																									
0x01																									
0x02																									
0x03																									
0x04																									
0x05																									
0x06																									
0x07																									
0x08																									
0x09																									
…	…																								
0x1F																									

Response

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

5.2.15.6. Scrolling LCD Current Display

This command is used to set the scrolling feature of the current LCD display:

Command

Command	Class	INS	P1	P2	Lc	Data In (6 Bytes)
Scrolling LCD	FF	00	6D	00	06	Scroll Ctrl

Where:

Scroll Ctrl: 6 Bytes. Scrolling Control Format.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
X Position	Y Position	Scrolling Range (Horizontal)	Scrolling Range (Vertical)	Refresh Speed Ctrl	Scrolling Direction

X Position: Horizontal Start Up Position, Ref to LCD Display Position Below

Y Position: Vertical Start Up Position, Ref to LCD Display Position Below

LCD Display Position (Total LCD Size: 128x32):

	Byte 0x00 (X = 0x00)								Byte 0x01 (X = 0x01)								…	Byte 0x0F (X = 0x0F)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	…	7	6	5	4	3	2	1	0
0x00																									
0x01																									
0x02																									
0x03																									
0x04																									
0x05																									
0x06																									
0x07																									
0x08																									
0x09																									
…	…																								
0x1F																									

Scrolling Range (Horizontal): How many 8 pixels in Horizontal after X position will be scrolled

Scrolling Range (vertical): How many pixels in Vertical after Y position will be scrolled

Refresh Speed Ctrl:

Bit0~Bit3 – how many pixel move pre scrolling

Bit4~Bit7 – Scrolling period

Bit7	Bit6	Bit5	Bit4	Scrolling period
0	0	0	0	1 Unit
0	0	0	1	3 Units
0	0	1	0	5 Units
0	0	1	1	7 Units
0	1	0	0	17 Units
0	1	0	1	19 Units
0	1	1	0	21 Units
0	1	1	1	23 Units
1	0	0	0	129 Units
1	0	0	1	131 Units
1	0	1	0	133 Units
1	0	1	1	135 Units
1	1	0	0	145 Units
1	1	0	1	147 Units
1	1	1	0	149 Units
1	1	1	1	151 Units

Scrolling Direction: the Scrolling Direction

Bit1	Bit0	Scrolling Direction
0	0	From Left to Right
0	1	From Right to Left
1	0	From Top to Bottom
1	1	From Bottom to Top

Response

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

5.2.15.7. Pause LCD Scrolling

This command is used to pause the LCD scrolling. To resume the scrolling, send the scrolling LCD command again.

Command

Command	Class	INS	P1	P2	Lc
Pause LCD Scrolling	FF	00	6E	00	00

Response

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

5.2.15.8. Stop LCD Scrolling

This command is used to stop the LCD Scrolling set. The LCD display comes back to normal display position.

Command

Command	Class	INS	P1	P2	Lc
Stop Scrolling LCD	FF	00	6F	00	00

Response

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

5.2.15.9. LCD Contrast Control

This command is used to control the LCD contrast.

Command

Command	Class	INS	P1	P2	Lc
LCD Contrast Control	FF	00	6C	Contrast Control	00

Where:

Contrast Control: 1 Byte.

The value range is between 00 to 0F. Larger value brightens the contrast. Lower range, on the other hand, darkens the contrast.

Response

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

5.2.15.10. LCD Backlight Control

This command controls the LCD Backlight.

Command

Command	Class	INS	P1	P2	Lc
LCD Backlight Control	FF	00	64	Backlight Control	00

Where:

Backlight Control: 1 Byte.

CMD	Description
00	LCD Backlight Off
FF	LCD Backlight On

Response

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

Appendix A. Basic Program Flow for Contactless Applications

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

Step 1. Connect the ACR1222L PICC Interface with T=1 protocol.

Step 2. Access the PICC by exchanging APDUs.

..

Step N. Disconnect the ACR122L PICC Interface and close the application.

Appendix B. Access PCSC Compliant Tags (ISO 14443-4)

All ISO 14443-4 compliant cards (PICCs) would understand the ISO 7816-4 APDUs. The ACR1222L needs to communicate with the ISO 14443-4 compliant cards through exchanging ISO 7816-4 APDUs and Responses. ACR1222L 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 Command

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

ISO 7816-4 Response

Response	Data Out		
Result	Response Data	SW1	SW2

Where:

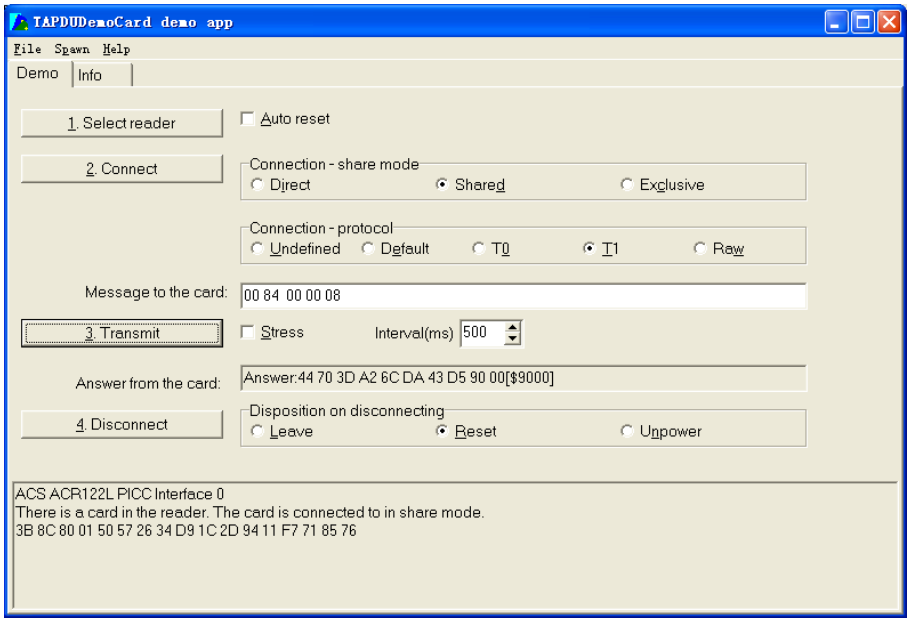
SW1, SW2 = `90 00` means the operation is completed successfully

= `63 00` means 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



The ATR of the tag is 3B 8C 80 01 50 57 26 34 D9 1C 2D 94 11 F7 71 85 76

In which,

The ATQB = 50 57 26 34 D9 1C 2D 94 11 F7 71 85. It is an ISO14443-4 Type B tag.

Step 2) Send an APDU, Get Challenge.

<< 00 84 00 00 08

>> 44 70 3D A2 6C DA 43 D5 [90 00]

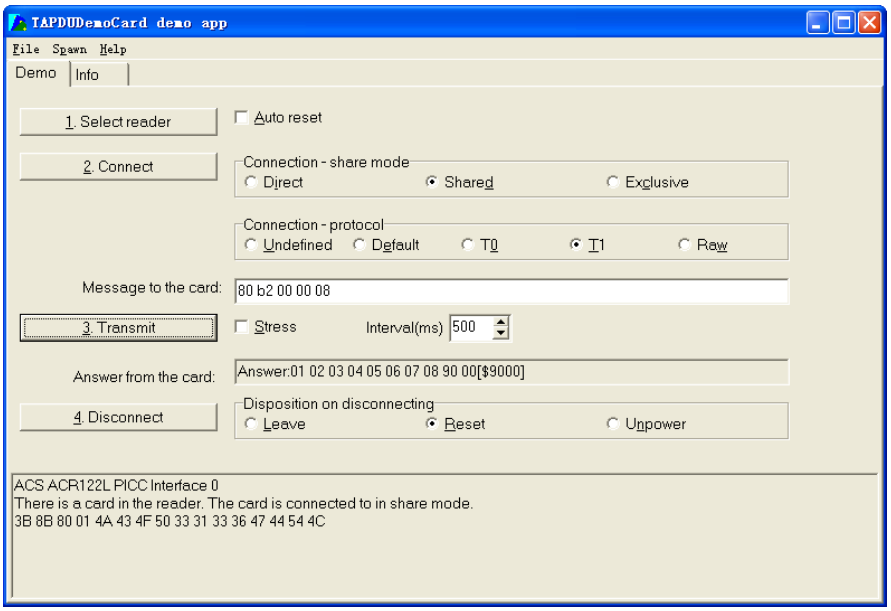
Note: For ISO 14443-4 Type A tags, the ATS can be obtained by using the APDU "FF CA 01 00 00"

Example: ISO 7816-4 APDU

To read 8 bytes from an ISO 14443-4 TypeA PICC.

APDU ={80 B2 80 00 08}

Class	=	80
INS	=	B2
P1	=	80
P2	=	00
Lc	=	None
Data In	=	None
Le	=	08



Answer: 01 02 03 04 05 06 07 08 [90 00]

Appendix C. Access DESFire Tags (ISO 14443-4)

The 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 = 90; INS = 0A (DESFire Instruction); P1 = 00; P2 = 00

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

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

> **Note:** 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=60

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

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

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

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.

Appendix D. 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 [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 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.

Appendix E. 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	

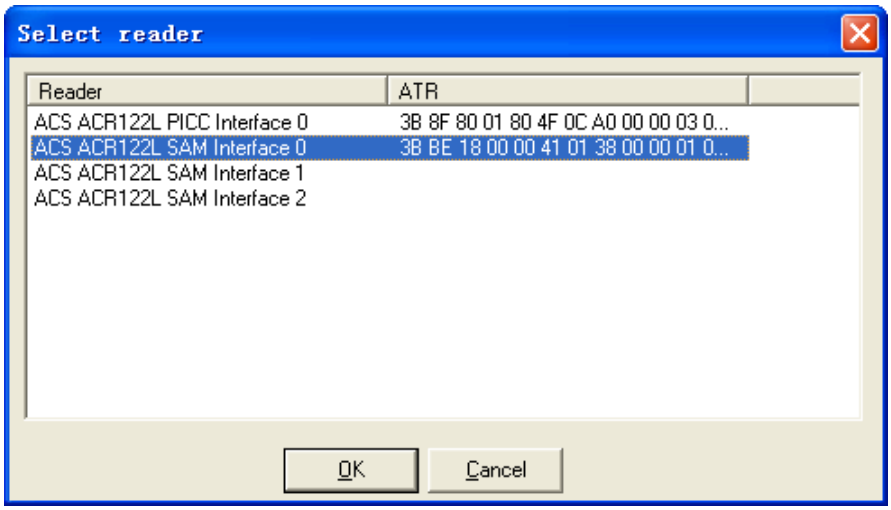
Reserved for internal use
User Block Lock & Status
OTP bits

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

Appendix F. Basic Program Flow for SAM 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.

Step 1. Connect the ACR1222L SAM Interface N(N = 0, 1, 2) with T=0 or T=1protocol.



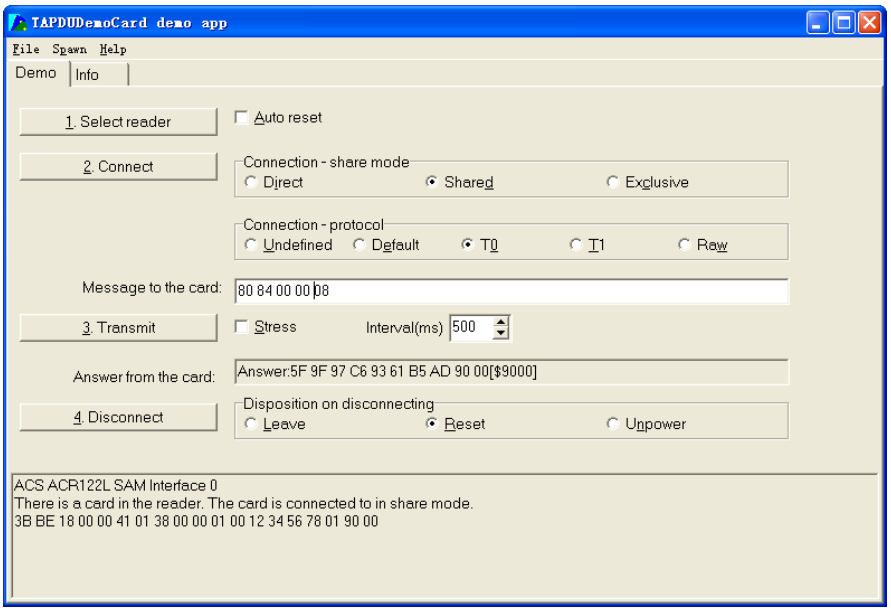
Step 2. Access the PICC by exchanging APDUs.

..

Step N. Disconnect the ACR1222L SAM Interface N(N = 0, 1, 2). Close the application.

Appendix G. Access ACOS3 SAM Cards (ISO 7816)

1) Connect the Tag.



The ATR of the tag is 3B BE 18 00 00 41 01 38 00 00 01 00 12 34 56 78 01 90 00

In which,

TD1 = 00 and TD2 is absent ,So the SAM Card is a T=0 SAM Card

2) Get a 'random' for the SAM Card.

<< 80 84 00 00 08

>> 5F 9F 97 C6 93 61 B5 AD 90 00[$9000]

3) Create a file on SAM Card and open it.

<<80 20 07 00 08 41 43 4F 53 54 45 53 54

>>90 00[$9000]

<<80 A4 00 00 02 FF 02

>>90 00[$9000]

<<80 D2 00 00 04 00 00 01 00

>>90 00[$9000]

<<80 A4 00 00 02 FF 04

>>90 00[$9000]

<<80 D2 00 00 06 ff 01 00 00 55 55

>>90 00[$9000]

<<80 A4 00 00 02 55 55

>>91 00[$9000]

File name is 55 55

4) Write a date to the file in 3)step.

```
<<80 d2 00 00 08 01 02 03 04 05 06 07 08
>>90 00[$9000]
```

5) Read a date from a file.

```
<<80 b2 00 00 08
>>01 02 03 04 05 06 07 08 90 00[$9000]
```

Appendix H. Example of LED & Buzzer Control Command

Example 1: To read the existing LED State.

Assume that both LED_0 and LED_1 are OFF initially.

Not linked to the buzzer.

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

Response = "90 00". LED_0 and LED_1 are OFF.

Example 2: To turn on LED_0 and LED_1

Assume that both LED_0 and LED_1 are OFF initially.

Not link to the buzzer.

APDU = "FF 00 40 0F 04 00 00 00 00"

Response = "90 03". LED_0 and LED_1 are ON,

To turn off both LED_0 and LED_1,

APDU = "FF 00 40 0C 04 00 00 00 00"

Example 3: To turn off the LED_0 only, and left the LED_1 unchanged

Assume that both LED_0 and LED_1 are ON initially

Not link to the buzzer

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

Response = "90 02". LED_1 is not changed (ON); LED_0 is OFF.

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

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

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

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

T1 Duration = 2000ms = 14

T2 Duration = 0ms = 00

Number of repetition = 01

Link to Buzzer = 01

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

Response = "90 02"

Example 5: To blink the Red LED of 1Hz for 3 times. After that, resume to initial state

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

If 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 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 = 500 ms 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`"

Response = "`90 02`"

Example 6: To blink the LED_0 and LED_1 of 1Hz for 3 times

Assume that both LED_0 and LED_1 are initially OFF.

Both Initial LED_0 and LED_1 Blinking States are ON

The buzzer will turn on during both the T1 and T2 duration

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

T1 Duration = 500ms = `05`

T2 Duration = 500ms = `05`

Number of repetition = `03`

Link to Buzzer = `03`

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

Response = "`90 00`"

Example 7: To blink the LED_0 and LED_1 in turn of 1Hz for 3 times

Assume that both LED_0 and LED_1 are initially OFF.

The Initial LED_0 Blinking State is ON; The Initial LED_1 Blinking States is OFF.

The buzzer will turn on during the T1 duration.

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

T1 Duration = 500ms = `05`

T2 Duration = 500ms = `05`

Number of repetition = `03`

Link to Buzzer = `01`

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