Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1222L NFC Reader with LCD

Application Programming Interface V2.01

Version History

Date	Changes	Version
2011-10-17	• Initial Release	1.00
2012-09-13	Updated document formatting • Updated Section 6.0: Peripherals Control • Added Appendix H: ACR122U Compatible Commands • Deleted screenshots in Appendices B and D • Removed "Example of LED & Buzzer Control Command" from Appendix	1.01
2014-09-17	• Updated Photo and Product description • Updated Section 2.0: Features • Updated Section 6.2: Corrected Buzzer duration from unit=10 to unit=100 • Added Sections 6.8 & 6.9: Set/Read LED and Buzzer Behaviors • Updated RoHS2 certification	1.02
2015-05-28	• 2015-05-28Updated Section 1.0: Introduction • Updated Section 2.0: Features • Updated Document Formatting • Updated Sections 4.1.2.4 to 5.1.2.4 • Updated Appendix B	1.03
2017-07-17	• Updated Section 2.0 Features	1.04
2022-03-04	• Updated Section 6.14: LCD control command	2.00

Advanced Card Systems Ltd.
Card & Reader Technologies

Table of Contents

List of Figures

List of Tables

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 three Secure Access Module (SAM) slots which can be used with ISO 7816 compliant Class A SAM cards to add a layer of security for contactless smart card applications.

ACR1222L serves as an intermediary device between the computer and the smart card. The reader is connected to the computer via USB port and carries out the commands whether 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.

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

2.0. Features

- USB Full Speed Interface
- CCID-compliance
- Smart Card Reader:
    - Contactless Interface:
        - 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)
    - SAM Interface:
        - Three SAM slots
        - Supports ISO 7816-acompliant Class A SAM cards
- Application Programming Interface:
    - Supports PC/SC
    - Supports CT-API (through wrapper on top of PC/SC)
- Built-in Peripherals:
    - Two-line graphic LCD with multi-language support (i.e. Chinese, English, Japanese and several European languages)
    - Four user-controllable LEDs
    - User-controllable buzzer
- USB Firmware Upgradability
- Supports Android™ 3.1 and later [1]
- Compliant with the following standards:
    - ISO 14443
    - ISO 7816 (SAM slot)
    - PC/SC
    - CCID
    - CE
    - FCC
    - RoHS 2
    - REACH
    - KC
    - VCCI
    - Microsoft WHQL

[1] Uses an ACS-defined Android Library

3.0. Architecture

3.1 Reader Block Diagram

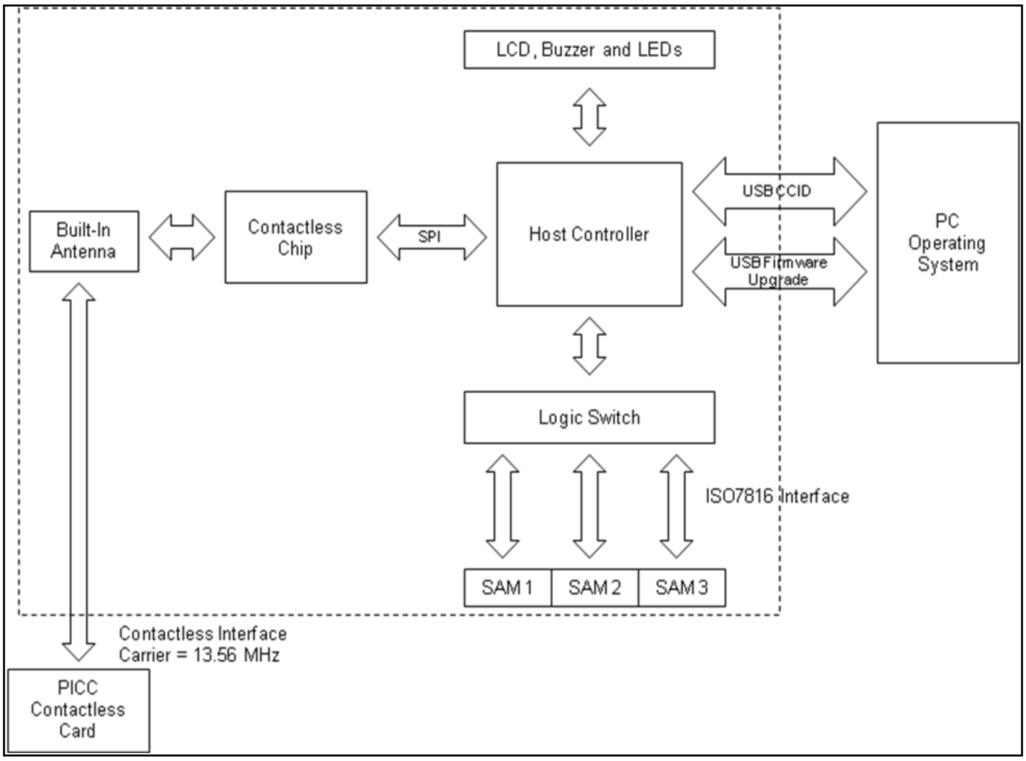


Figure 1: Reader Block Diagram

3.2 Communication among the PC/SC Driver, PICC and SAM

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

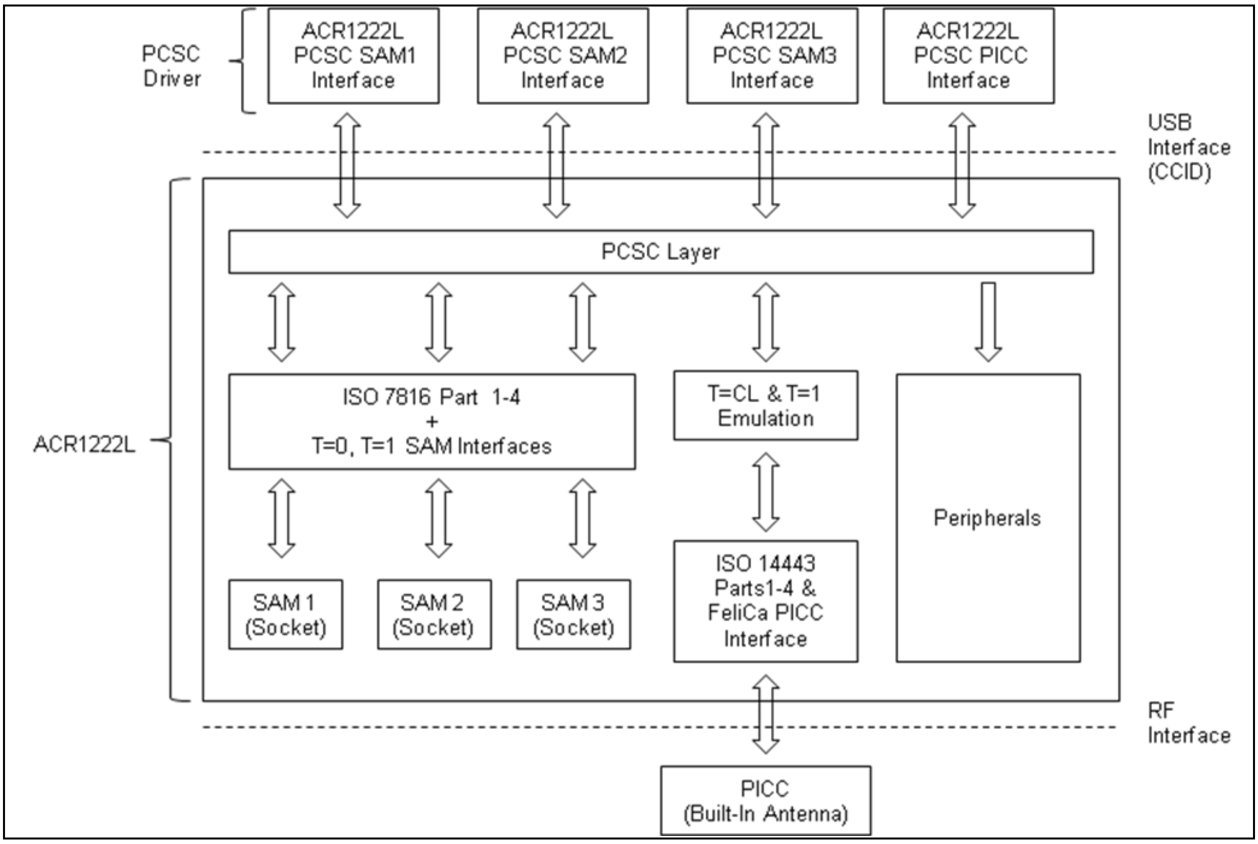


Figure 2: ACR1222L Architecture

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.

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 be sent from host to ACR1222L (data packet size is 64 bytes)

Bulk IN For response to be sent from ACR1222L to host (data packet size is 64 bytes)

Interrupt IN For card status message to be 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 computer 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 Present" and "Card Removal" events.

User-controllable Monotone Buzzer.

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

Table 2: 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. Contactless Smart Card Protocol

5.1 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 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
	C0h .. 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

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}

Where:

Length (YY) = 0Ch

RID = A0 00 00 03 06h (PC/SC Workgroup)

Standard (SS) = 03h (ISO 14443A, Part 3)

Card Name (C0 ... C1) = 00 01h (MIFARE 1K)

00 02h: MIFARE 4K

00 03h: MIFARE Ultralight

00 26h: MIFARE Mini

F0 04h: Topaz and Jewel

F0 11h: FeliCa 212K

F0 12h FeliCa 424K

FF 28h: JCOP 30

FFh [SAK]: undefined tags

5.1.2 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 ISO 14443-4 specifications. ISO 14443B: The higher layer response from the ATTRIB response (ATQB). Refer to the ISO 14443-3 specifications.
	XXh XXh XXh	Tk	
4+N	UUh	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 80h (6 bytes of ATR)

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

APDU Command = FF CA 01 00 00h

APDU Response = 06 75 77 81 02 90 00h

ATS = 06 75 77 81 02 80h

Example 2:

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

Application Data of ATQB = 12 53 54 4Eh

Protocol Info of ATQB = 33 81 C3h

5.2 Pseudo APDUs for Contactless Interface

The following Pseudo APDUs are used for exchanging data with non-PC/SC compliant tags. Pseudo APDUs can be sent through the PICC Interface is the tag is already connected or sent using Escape Command if the tag is not yet presented.

5.2.1 Direct Transmit via PC_to_RDR_Escape

Command

Command	Class	INS	P1	P2	Lc	Data In
Direct Transmit	E0h	00h	00h	24h	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	E1h	00h	00h	00h	Number of Bytes to be Received	Contactless Chip and Tag Response

Where:

Data Out: Contactless Chip and Tag Response

Contactless Chip and Tag Response returned by the reader

5.2.2 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	FFh	CAh	00h 01h	00h	00h (Max Length)

Response if P1 = 00h

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

Response if P1 = 01h

Response	Data Out		
Result	ATS	SW1	SW2

Response Codes

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

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

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

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

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

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

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

5.2.3.1 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	FFh	82h	Key Structure	Key Number	06h	Key (6 bytes)

Where:

Key Structure: 1 Byte

00h = Key is loaded into the reader volatile memory

Other = Reserved

Key Number: 1 Byte

00h – 01h = Key Location

The keys will 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 FFh}

Response

Response	Data Out	
Result	SW1	SW2

Where:

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

= 63 00h means the operation failed

Example:

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

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

5.2.3.2 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)	FFh	88h	00h	Block Number	Key Type	Key Number

Command

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

Where:

Authenticate Data Bytes: 5 Bytes

Byte1	Byte 2	Byte 3	Byte 4	Byte 5
Version 01h	00h	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 00h consists of Blocks {00h, 01h, 02h and 03h}; Sector 01h consists of Blocks {04h, 05h, 06h and 07h}; the last sector 0F consists of Blocks {3Ch, 3Dh, 3Eh and 3Fh}.*

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

Key Type: 1 Byte

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

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

Key Number: 1 Byte

00h ~ 01h = Key Location

Response Format

Response	Data Out	
Result	SW1	SW2

Where:

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

= 63 00h means 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	00h ~ 02h	03h
Sector 1	04h ~ 06h	07h
..		
..		
Sector 14	38h ~ 0Ah	3Bh
Sector 15	3Ch ~ 3Eh	3Fh

1K Bytes

Table 3: 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	00h ~ 02h	03h
Sector 1	04h ~ 06h	07h
..		
..		
Sector 30	78h ~ 7Ah	7Bh
Sector 31	7Ch ~ 7Eh	7Fh

2K Bytes

Table 4: MIFARE 4K Memory Map

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	80h ~ 8Eh	8Fh
Sector 33	90h ~ 9Eh	9Fh
..		
..		
Sector 38	E0h ~ EEh	EFh
Sector 39	F0h ~ FEh	FFh

2K Bytes

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

APDU = { FF 88 00 04 60 00h }

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

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

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

Byte Number	0	1	2	3	Page
Serial Number	SN0	SN1	SN2	BCC0	0
Serial Number	SN3	SN4	SN5	SN6	1
Internal / Lock	BCC1	Internal	Lock0	Lock1	2
OTP	OPT0	OPT1	OTP2	OTP3	3
Data read/write	Data0	Data1	Data2	Data3	4
Data read/write	Data4	Data5	Data6	Data7	5
Data read/write	Data8	Data9	Data10	Data11	6
Data read/write	Data12	Data13	Data14	Data15	7
Data read/write	Data16	Data17	Data18	Data19	8
Data read/write	Data20	Data21	Data22	Data23	9
Data read/write	Data24	Data25	Data26	Data27	10
Data read/write	Data28	Data29	Data30	Data31	11
Data read/write	Data32	Data33	Data34	Data35	12
Data read/write	Data36	Data37	Data38	Data39	13
Data read/write	Data40	Data41	Data42	Data43	14
Data read/write	Data44	Data45	Data46	Data47	15

512 bits Or 64 bytes

Table 5: MIFARE Ultralight Memory Map

5.2.3.3 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	FFh	B0h	00h	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: 10h (16 bytes). The starting block only (Single Block Mode)

Example 2: 40h (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 00h means the operation is completed successfully

= 63 00h means the operation failed

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

APDU = { FF B0 00 04 10h }

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

APDU = { FF B0 00 80 F0h }

5.2.3.4 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	FFh	D6h	00h	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: 10h (16 bytes). The starting block only. (Single Block Mode)

Example 2: 30h (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 00h means the operation is completed successfully

= 63 00h means the operation failed

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

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

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

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

5.2.3.5 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	FFh	D7h	00h	Block Number	05h	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

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

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

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

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

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

VB_Value			
MSB			LSB
FFh	FFh	FFh	FCh

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

VB_Value			
MSB			LSB
00h	00h	00h	01h

Response

Response	Data Out	
Result	SW1	SW2

Where:

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

= 63 00h means the operation failed

5.2.3.6 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	FFh	B1h	00h	Block Number	00h

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 FCh }

VB_Value			
MSB			**LSB**
FFh	FFh	FFh	FCh

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

VB_Value			
MSB			**LSB**
00h	00h	00h	01h

Response

Response	Data Out	
Result	SW1	SW2

Where:

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

= 63 00h means the operation failed

5.2.3.7 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	FFh	D7h	00h	Source Block Number	02h	03h	Target Block Number

Where:

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

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

Response

Response	Data Out	
Result	SW1	SW2

Where:

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

= 63 00h means the operation failed

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

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

Example 2: Read the value block 05h

APDU = {FF B1 00 05 00h }

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

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

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

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

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

All ISO 14443-4 compliant cards (PICCs) 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 **PICC Commands (T=CL Emulation) for MIFARE 1K/4K Memory Cards.**

Command

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

Response

Response	Data Out		
Result	Response Data	SW1	SW2

Where:

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

= 63 00h 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 00h."*

Example: ISO 7816-4 APDU

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

APDU = { 80 B2 80 00 08h }

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

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

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

6.0. Peripherals Control

The reader's peripherals control commands are implemented by using PC_to_RDR_Escape. The vendor IOCTL for the escape commands is 3500.

6.1 Get Firmware Version

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

Command

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

Response

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

Example:

Response = E1 00 00 00 12 41 43 52 31 32 32 32 4C 2D 55 20 56 33 31 33 2E 30 31 h

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

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

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

6.2 Buzzer Control

This command is used to control the buzzer output.

Command

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

Where:

Buzzer On Duration: 1 Byte

01 – FFh = Duration (unit: 10 ms)

Response

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

Where:

Timer: 1 Byte

The value is the MCU's timer and is not to be used in the application.

6.3 Get Serial Number of the reader

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

Command

Command	Class	INS	P1	P2	Lc
Get Serial Number	E0h	00h	00h	33h	00h

Response

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	10h	Serial Number (16 bytes)

Where:

Serial Number: 16 Bytes

The value is the serial number of the reader.

6.4 Read the PICC Operating Parameter

This command is used to check current PICC Operating Parameter.

Command

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

Response

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

Where:

Operating Parameter: 1 Byte

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

6.5 Set PICC Operating Parameter

The command is used to set the PICC Operating Parameter.

Command

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

Response

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

Where:

Operating Parameter: 1 Byte. Default value = 03h

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

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

6.6 2 LEDs Control

This command is used to control the first 2 LEDs.

Command

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

Response

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

Where:

LED Status: 1 Byte

LED Status	Description	Description
Bit 0	Green LED	1 = ON 0 = OFF
Bit 1	Blue LED	1 = ON 0 = OFF
Bit 2 – 7	RFU	RFU

6.7 4 LEDs Control

This command is used to control the 4 LEDs.

Command

Command	Class	INS	P1	P2	Lc
4 LEDs Control	FFh	00h	44h	bLEDsState	00h

Where:

P2: 1 Byte. bLEDsState.

CMD	Item	Description
Bit 0	LED_0 State Green LED	1 = On; 0 = Off
Bit 1	LED_1 State Blue LED	1 = On; 0 = Off
Bit 2	LED_2 State Orange LED	1 = On; 0 = Off
Bit 3	LED_3 State Red LED	1 = On; 0 = Off
Bits 4 – 7	RFU	RFU

Data Out: SW1 SW2.

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

6.8 Set Default LED and Buzzer Behaviors

This command is used to set the default behavior of the LEDs and buzzer.

Note: *This command is supported by firmware version 312 and above only.*

Command

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

Where:

Default Behavior: Default value = 8Fh (1 Byte)

Operating Parameter	Description	Option
Bit 0	RFU	RFU
Bit 1	PICC Polling Status LED	To show the PICC polling status. 1 = Enable 0 = Disable
Bit 2	RFU	RFU
Bit 3	RFU	RFU
Bit 4	Card Insertion and Removal Events Buzzer	To make a beep whenever a card insertion or removal event is detected (for PICC). 1 = Enable 0 = Disable
Bit 5	Contactless Chip Reset Indication Buzzer	To make a beep when the contactless chip is reset. 1 = Enable 0 = Disable
Bit 6	RFU	RFU
Bit 7	Card Operation Blinking LED	To blink the LED whenever the PICC is being accessed. 1 = Enable 0 = Disable

Response

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Default Behavior

6.9 Read Default LED and Buzzer Behaviors

This command is used to set the read the current default behaviors of LEDs and buzzer.

Note: *This command is supported by firmware version 312 and above only.*

Command

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

Response

Response	Class	INS	P1	P2	Le	Data Out
Result	E1h	00h	00h	00h	01h	Default Behavior

Where:

Behavior: 1 Byte

Operating Parameter	Description	Option
Bit 0	RFU	RFU
Bit 1	PICC Polling Status LED	To show the PICC polling status. 1 = Enable 0 = Disable
Bit 2	RFU	RFU
Bit 3	RFU	RFU
Bit 4	Card Insertion and Removal Events Buzzer	To make a beep whenever a card insertion or removal event is detected (for PICC). 1 = Enable 0 = Disable
Bit 5	Contactless Chip Reset Indication Buzzer	To make a beep when the contactless chip is reset. 1 = Enable 0 = Disable
Bit 6	RFU	RFU
Bit 7	Card Operation Blinking LED	To blink the LED whenever the PICC is being accessed. 1 = Enable 0 = Disable

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

6.10 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	FFh	00h	4Ah	00h	00h	Data Len (MSB)	Data Len (LSB)	Data

Where:

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	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation failed.

6.11 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	FFh	00h	4Bh	00h	00h	Data Len (MSB)	Data Len (LSB)	Data

Where:

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	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation failed.

6.12 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	FFh	00h	4Ch	00h	00h	Data Len (MSB)	Data Len (LSB)

Where:

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

6.13 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	FFh	00h	4Dh	00h	00h	Data Len (MSB)	Data Len (LSB)

Where:

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

6.14 LCD Control Command

6.14.1 Clear LCD

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

Command

Command	Class	INS	P1	P2	Lc
Clear LCD	FFh	00h	60h	00h	00h

Response

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

6.14.2 LCD Display (ASCII Mode)

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

Command

Command	Class	INS	P1	P2	Lc	Data In (Max. 16Bytes)
LCD Display	FFh	00h	68h	LCD XY Position	LCD Message Length	LCD Message

Where:

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:

Note: *The length of the LCD message should multiple of 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		LCD XY POSITION
2nd LINE	40		41		42		43		44		45		46		47		

Where:

Lc: LCD Message Length

The length of the LCD message (max. 10h); 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.

Note: *The length of the LCD message should multiple of 2*

Data In: LCD Message

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

Please follow the Font table below for the LCD Character Index.

Note: *Size of the Characters in ASCII Fonts table is 8x16*

\	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	→	←

ASCII Font table

Response

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

Example 1:

Display 2 char ("56") in display position 5 & 6 (XY Position 02h):

Send escape command:

```
<< FF 00 68 02 02 35 56
>> [90 00]
```

Example 2:

Display 2 char ("56") in display position 6 & 7 (have to fill all the data in line 1 block 02 & 03):

Send escape command:

```
<< FF 00 68 02 04 20 35 36 20
>> [90 00]
```


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

6.14.3 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	FFh	00h	69h	LCD XY Position	LCD Message Length	LCD Message

Where:

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:

<table>
<tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>DISPLAY POSITION</th></tr>
<tr><td>1st LINE</td><td colspan="2">00</td><td colspan="2">01</td><td colspan="2">02</td><td colspan="2">03</td><td colspan="2">04</td><td colspan="2">05</td><td colspan="2">06</td><td colspan="2">07</td><td rowspan="2">LCD XY POSITION</td></tr>
<tr><td>2nd LINE</td><td colspan="2">40</td><td colspan="2">41</td><td colspan="2">42</td><td colspan="2">43</td><td colspan="2">44</td><td colspan="2">45</td><td colspan="2">46</td><td colspan="2">47</td></tr>
</table>

Lc: LCD Message Length

The length of the LCD message (max. 10h); 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 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	90h	00h	The operation is completed successfully.
Error	63h	00h	The operation failed.

6.14.4 LCD Display (Graphic Mode)

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

Command

Command	Class	INS	P1	P2	Lc	Data In (must be 32 Bytes)
LCD Display	FFh	00h	6Ah	Block Index	20h	Pixel Data {MSB .. LSB}

Where:

P2: Block Index

To set which block to start to update the LCD Display

<table>
<tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>DISPLAY POSITION</th></tr>
<tr><td>1st LINE</td><td colspan="2">00</td><td colspan="2">01</td><td colspan="2">02</td><td colspan="2">03</td><td colspan="2">04</td><td colspan="2">05</td><td colspan="2">06</td><td colspan="2">07</td><td rowspan="2">LCD XY POSITION</td></tr>
<tr><td>2nd LINE</td><td colspan="2">40</td><td colspan="2">41</td><td colspan="2">42</td><td colspan="2">43</td><td colspan="2">44</td><td colspan="2">45</td><td colspan="2">46</td><td colspan="2">47</td></tr>
</table>

Lc: Pixel Data Length

The length of the pixel data (must be 20h) (16x16 pixels, total 32 bytes)

Data In: Pixel Data

The pixel data to be sent to LCD for display.

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

<table>
<tr><th rowspan="2"></th><th rowspan="2">Lines no.</th><th colspan="16">P2 = 0x00h</th><th>…</th><th colspan="6">P2 = 0x07h</th></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>…</th><th>15</th><th>…</th><th>8</th><th>7</th><th>…</th><th>0</th></tr>
<tr><td rowspan="16">1st Line</td><td>0</td><td colspan="8">date byte 1</td><td colspan="8">date byte 2</td><td></td><td colspan="3">date byte 1</td><td colspan="3">date byte 2</td></tr>
<tr><td>1</td><td colspan="8">date byte 3</td><td colspan="8">date byte 4</td><td></td><td colspan="3">date byte 3</td><td colspan="3">date byte 4</td></tr>
<tr><td>2</td><td colspan="8">date byte 5</td><td colspan="8">date byte 6</td><td></td><td colspan="3">date byte 5</td><td colspan="3">date byte 6</td></tr>
<tr><td>3</td><td colspan="8">date byte 7</td><td colspan="8">date byte 8</td><td></td><td colspan="3">date byte 7</td><td colspan="3">date byte 8</td></tr>
<tr><td>4</td><td colspan="8">date byte 9</td><td colspan="8">date byte 10</td><td></td><td colspan="3">date byte 9</td><td colspan="3">date byte 10</td></tr>
<tr><td>5</td><td colspan="8">date byte 11</td><td colspan="8">date byte 12</td><td></td><td colspan="3">date byte 11</td><td colspan="3">date byte 12</td></tr>
<tr><td>6</td><td colspan="8">date byte 13</td><td colspan="8">date byte 14</td><td></td><td colspan="3">date byte 13</td><td colspan="3">date byte 14</td></tr>
<tr><td>7</td><td colspan="8">date byte 15</td><td colspan="8">date byte 16</td><td></td><td colspan="3">date byte 15</td><td colspan="3">date byte 16</td></tr>
<tr><td>8</td><td colspan="8">date byte 17</td><td colspan="8">date byte 18</td><td></td><td colspan="3">date byte 17</td><td colspan="3">date byte 18</td></tr>
<tr><td>9</td><td colspan="8">date byte 19</td><td colspan="8">date byte 20</td><td></td><td colspan="3">date byte 19</td><td colspan="3">date byte 20</td></tr>
<tr><td>10</td><td colspan="8">date byte 21</td><td colspan="8">date byte 22</td><td></td><td colspan="3">date byte 21</td><td colspan="3">date byte 22</td></tr>
<tr><td>11</td><td colspan="8">date byte 23</td><td colspan="8">date byte 24</td><td></td><td colspan="3">date byte 23</td><td colspan="3">date byte 24</td></tr>
<tr><td>12</td><td colspan="8">date byte 25</td><td colspan="8">date byte 26</td><td></td><td colspan="3">date byte 25</td><td colspan="3">date byte 26</td></tr>
<tr><td>13</td><td colspan="8">date byte 27</td><td colspan="8">date byte 28</td><td></td><td colspan="3">date byte 27</td><td colspan="3">date byte 28</td></tr>
<tr><td>14</td><td colspan="8">date byte 29</td><td colspan="8">date byte 30</td><td></td><td colspan="3">date byte 29</td><td colspan="3">date byte 30</td></tr>
<tr><td>15</td><td colspan="8">date byte 31</td><td colspan="8">date byte 32</td><td></td><td colspan="3">date byte 31</td><td colspan="3">date byte 32</td></tr>
<tr><td>…</td><td colspan="24"></td></tr>
<tr><td>2nd Line</td><td>31</td><td colspan="8">date byte 31</td><td colspan="8">date byte 32</td><td></td><td colspan="3">date byte 31</td><td colspan="3">date byte 32</td></tr>
</table>

Response

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

Example:

Graphic mode on XY position = 44 using below picture

Font Data	CGRAM Address	CGRAM DATA (High Byte)								CGRAM DATA (Low Byte)								CGRAM DATA High Byte	CGRAM DATA Low Byte
B15~B0	AC6~AC0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		
A0A0H	0x00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00	0x00
		0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0x1F	0xFF
		0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0x10	0x01
		0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0x10	0x01
		0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0x60	0x01
		1	0	0	0	1	1	0	1	1	0	1	1	0	1	1	1	0x8D	0xB7
		1	0	0	0	1	1	0	1	1	0	1	1	0	1	1	1	0x8D	0xB7
		1	0	0	0	1	1	0	1	1	0	1	1	0	1	1	1	0x8D	0xB7
		1	0	0	0	1	1	0	1	1	0	1	1	0	1	1	1	0x8D	0xB7
		1	0	0	0	1	1	0	1	1	0	1	1	0	1	1	1	0x8D	0xB7
		1	0	0	0	1	1	0	1	1	0	1	1	0	1	1	1	0x8D	0xB7
		0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0x60	0x01
		0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0x10	0x01
		0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0x10	0x01
		0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0x1F	0xFF
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00	0x00

Display the above graphic in XY position 44h:

Send escape command:

```
<<  FF 00 6A 44 20 00 00 1F FF 10 01 10 01 60 01 8D B7 8D B7 8D B7 8D B7 8D
    B7 8D B7 60 01 10 01 10 01 1F FF 00 00

>> [90 00]

XY Position = 44, 32 bytes graphical pixel data
```

6.14.5 LCD Create Word

This command is used to create new customized word, total 50 characters can be created. The created character will save to RAM, and the data will lose after disconnect to PC.

Command

Command	Class	INS	P1	P2	Lc	Data In (must be. 32 Bytes)
LCD Display	FFh	00h	65h	Word Index	20h	Pixel Data {MSB .. LSB}

Where:

P2: Word Index

To save the pixel data to the GB Code.

The data will save to the code index in GB Table, and the saved character can be displayed by using LCD Display (GB Mode).

The matching table of Character index and the GB character index:

Character index	GB character index
0x00	A0B0
0x01	A0B1
⋮	⋮
0x32	A0E2

Pixel data position:

Lines no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	date byte 1								date byte 2							
1	date byte 3								date byte 4							
2	date byte 5								date byte 6							
3	date byte 7								date byte 8							
4	date byte 9								date byte 10							
5	date byte 11								date byte 12							
6	date byte 13								date byte 14							
7	date byte 15								date byte 16							
8	date byte 17								date byte 18							
9	date byte 19								date byte 20							
10	date byte 21								date byte 22							
11	date byte 23								date byte 24							
12	date byte 25								date byte 26							
13	date byte 27								date byte 28							
14	date byte 29								date byte 30							
15	date byte 31								date byte 32							

Example:

CGRAM DATA (High Byte)								CGRAM DATA (Low Byte)								CGRAM DATA High Byte	CGRAM DATA Low Byte
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0x04	0x20
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0x04	0x20
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0xFF	0xFE
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0x04	0x20
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0x04	0x20
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0x02	0x00
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0x01	0x00
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0x00	0x80
0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0x04	0x88
0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0x24	0x04
0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0x24	0x04
0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0x24	0x02
0	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0x24	0x1 2
0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0x44	0x10
0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0x03	0xF0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00	0x00

Set GB Character (A0B0) using above data:

Send escape command:

```
<<  FF 00 65 00 20 04 20 04 20 FF FE 04 20 04 20 02 00 01 00 00 80 04 88 24
    04 24 04 24 02 24 12 44 10 03 F0 00 00

>> [90 00]

Character index = 00, 32 bytes pixel data
```

6.14.6 Scroll Current LCD Display

In API 2.0, the scrolling function is removed. Please refer to Appendix I.1 for the alternative method.

6.14.7 LCD Contrast Control

This command is used to control the LCD contrast.

Command

Command	Class	INS	P1	P2	Lc
LCD Contrast Control	FFh	00h	6Ch	Contrast Control	00h

Where:

Contrast Control: 1 Byte

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

Response

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

6.14.8 LCD Backlight Control

This command controls the LCD Backlight.

Command

Command	Class	INS	P1	P2	Lc
LCD Backlight Control	FFh	00h	64h	Backlight Control	00h

Where:

Backlight Control: 1 Byte

CMD	Description
00h	LCD Backlight Off
FFh	LCD Backlight On

Response

Results	SW1	SW2	Meaning
Success	90h	00h	The operation is successfully completed.
Error	63h	00h	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 **PICC Commands (T=CL Emulation) for MIFARE 1K/4K Memory Cards.**

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 00h means the operation is completed successfully

= 63 00h 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 00h"*

Example: ISO 7816-4 APDU

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

APDU ={80 B2 80 00 08h}

Class = 80h
INS = B2h
P1 = 80h
P2 = 00h
Lc = None
Data In = None
Le = 08h

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

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

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

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

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

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

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

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

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

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

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

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

To get the version of the DESFire card:

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

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

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

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

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

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

Example 3: MIFARE DESFire Native Command.

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

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

APDU = {0A 00h}

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

In which, the first byte "AFh" is the status code returned by the DESFire Card.

The Data inside the blanket [$1D D7h] can simply be ignored by the application.

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

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

To get the version of the DESFire card:

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

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

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

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

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

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

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

Appendix 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 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 00h". Please refer to the FeliCa specification for more detailed information.*

Appendix E. 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
```


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

Topaz Memory Map.

Memory Address = Block No * 8 + Byte No

e.g. Memory Address 08h = 1 x 8 + 0 = Block 1: Byte-0 = Data0

e.g. Memory Address 10h = 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)

Step 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

Step 2: Get a 'random' for the SAM Card.

```
<< 80 84 00 00 08
>> 5F 9F 97 C6 93 61 B5 AD 90 00[$9000]
```

Step 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
```

Step 4: Write a date to the file in step 3.

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

Step 5: Read a date from a file.

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


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

Appendix H. ACR122U Compatible Commands

Appendix H.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	FFh	00h	00h	00h	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 00h means the operation is completed successfully

= 63 00h means the operation failed

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

Appendix H.2. Get Firmware Version

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

Command

Command	Class	INS	P1	P2	Le
Get Response	FFh	00h	48h	00h	00h

Response

Response	Data Out
Result	Firmware Version

E.g. Response = 41 43 52 31 32 32 32 4C 2D 55 20 56 33 31 33 2E 30 31h
= ACR1222L-U V313.01 (ASCII)

Appendix H.3. Get 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	FFh	00h	50h	00h	00h

Response

Response	Data Out
Result	PICC Operating Parameter

Where:

PICC Operating Parameter. Default Value = FFh.

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

Appendix H.4. 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	FFh	00h	51h	New PICC Operating Parameter	00h

Response

Response	Data Out
Result	PICC Operating Parameter

Where:

PICC Operating Parameter: Default Value = FFh

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

Appendix I. LCD Program example

Appendix I.1. LCD horizontal Scrolling

The scrolling function can be achieved by frequently update the display data on LCD. (The follow displays are only used for illustrating the program concept, the results are not identical to the real LCD display)

Example 1:

Assume the sample string “0123456789abcdef 01“is stored in the application program.

To achieve the right scrolling function, shift toward right by 1 character (8 pixels) in each update: (Note: only ASCII mode characters can shift 8 pixels in scrolling)

1. Write the first data to the LCD using **LCD Display (ASCII Mode)**

   Send escape command:

   ```
   << FF 00 68 00 10 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65 66
   >> [90 00]
   ```

2. To shift toward right by 8 pixels (one ASCII character size):

   Send escape command:

   ```
   << FF 00 68 00 10 31 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65
   >> [90 00]
   ```

3. To shift toward right by 8 pixels (one ASCII character size):

   Send escape command:

   `<< FF 00 68 00 10 30 31 30 31 32 33 34 35 36 37 38 39 61 62 63 64`

   `>> [90 00]`

4. To shift toward right by 8 pixels (one ASCII character size):

   Send escape command:

   `<< FF 00 68 00 10 20 30 31 30 31 32 33 34 35 36 37 38 39 61 62 63`

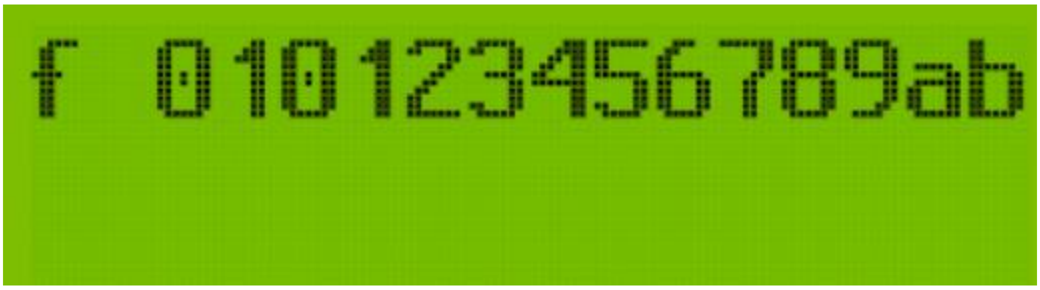
   `>> [90 00]`

5. To shift toward right by 8 pixels (one ASCII character size):

   Send escape command:

   `<< FF 00 68 00 10 66 20 30 31 30 31 32 33 34 35 36 37 38 39 61 62`

   `>> [90 00]`



6. To complete one cycle, repeat the shifing process until shifting back to step 1.

To slow down the scrolling speed, add a >100ms delay between each escape command.

Example 2:

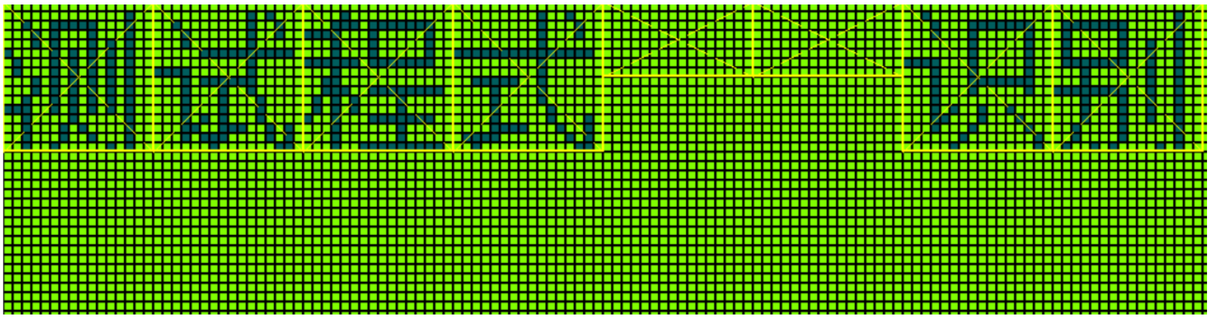
Assume the sample string “测试程式 识别“ is stored in the application program:

To achieve the left scrolling function, shift toward left by 1 GB character (16 pixels) in each update:

1. Write the first data to the LCD using **LCD Display (GB Mode)**

   Send escape command:

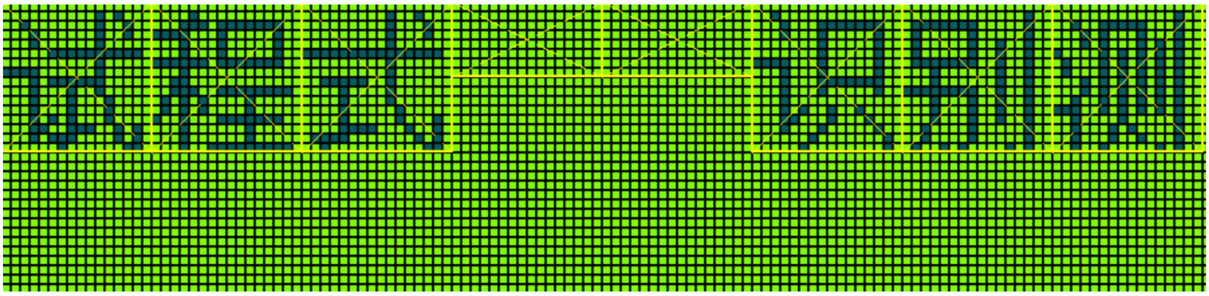
   ```
   << FF 00 69 00 10 B2 E2 CA D4 B3 CC CA BD A1 A1 A1 A1 CA B6 B1 F0
   >> [90 00]
   ```



2. To shift toward left by 16 pixels (one GB character size):

   Send escape command:

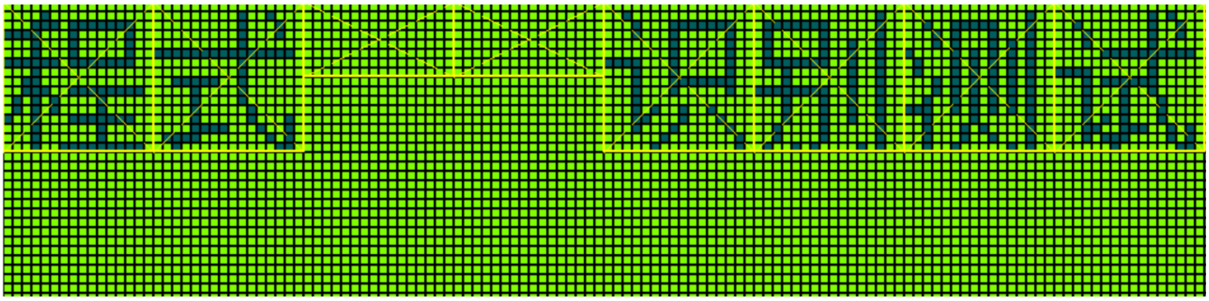
   ```
   << FF 00 69 00 10 CA D4 B3 CC CA BD A1 A1 A1 A1 CA B6 B1 F0 B2 E2
   >> [90 00]
   ```



3. To shift toward left by 16 pixels (one GB character size):

   Send escape command:

   ```
   << FF 00 69 00 10 B3 CC CA BD A1 A1 A1 A1 CA B6 B1 F0 B2 E2 CA D4
   >> [90 00]
   ```



4. To complete one cycle, repeat the shifing process until shifting back to step 1.