Advanced Card Systems Ltd.
Card & Reader Technologies

ACR89U-A2
Handheld
Smart Card Reader

Reference Manual V1.01

Table of Contents

List of Figures

List of Tables

1.0. Introduction

The ACR89U-A2 Handheld Smart Card Reader with NFC tag support is a versatile dual interface smart card reader with PINpad, which can be used to access ISO 7816 MCU cards, ISO 14443 Type A and B, MIFARE®, FeliCa and ISO 18092 or NFC tags. It can operate in both office and field-based environments using it PC-linked and standalone modes, respectively.

For PC-linked Mode, ACR89U-A2 acts as the intermediary device between the PC and the card. The reader, specifically to communicate with a contactless tag, MCU card, SAM card or device peripherals, will carry out a command issued from the PC.

This manual describes the use of ACR89 software programming interface to control the built-in accessories of the ACR89 multi-functional card reader. Built-in accessories are defined to be the keypad, LCD display, LEDs, buzzer and real-time clock, embedded in ACR89. Such components are not controlled through the smart card reader library. In addition, this document provides a guide on implementing PC/SC APDU commands for device contactless tags.

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

2.0. Hardware Design

2.1. Architecture

The architecture of the ACR89U-A2 library can be visualized as the following diagram:

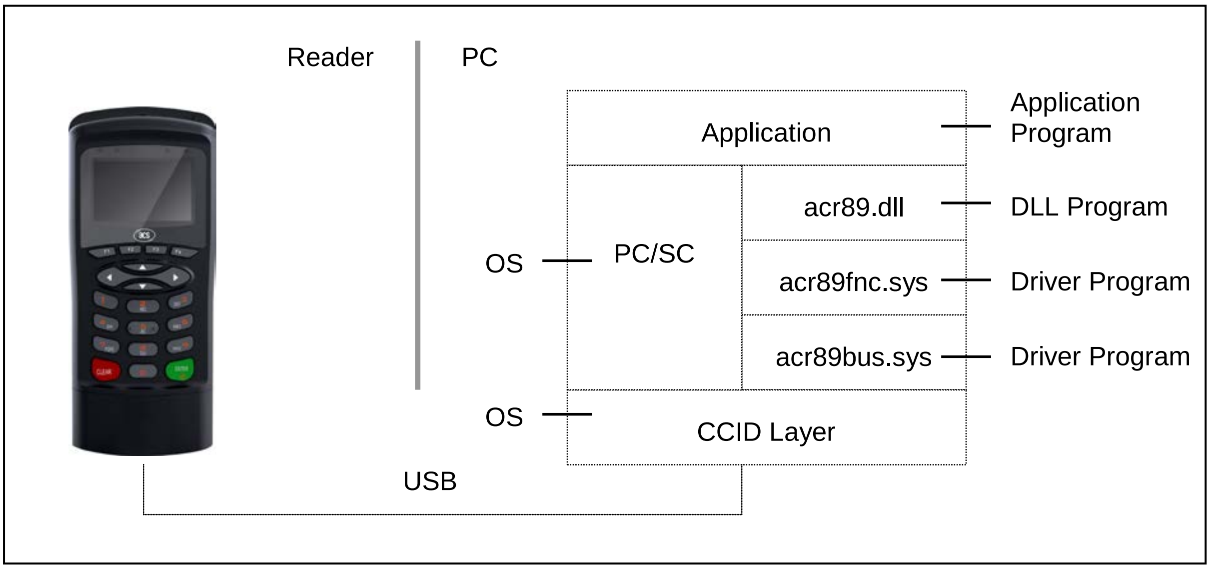

Figure 1: ACR89U-A2 Architecture

2.2. USB Interface

The ACR89U-A2 is connected to a computer through USB following the USB standard.

2.3. Communication Parameters

The ACR89U-A2 is connected to a computer through USB as specified in the USB Specification 2.0., working in full speed mode, i.e. 12 Mbps.

Pin	Signal	Function
1	V_{BUS}	+5 V power supply for the reader
2	D-	Differential signal transmits data between ACR89U-A2 and PC
3	D+	Differential signal transmits data between ACR89U-A2 and PC
4	GND	Reference voltage level for power supply

Table 1: USB Interface Wiring

Note*: In order for the ACR89U-A2 to function properly through USB interface, the device driver should be installed.*

2.4. Endpoints

The ACR89U-A2 uses the following endpoints to communicate with the host computer:

Control Endpoint – For setup and control purposes

Bulk OUT – For commands to be sent from host to ACR89U-A2 (data packet size is 64 bytes)

Bulk IN – For commands to be sent from ACR89U-A2 to host (data packet size is 64 bytes)

Interrupt IN – For card status message to be sent from ACR89U-A2 to host (data packet size is 8 bytes)

2.5. Contact Smart Card Interface

The interface between the ACR89U-A2 and the inserted smart card follows the specifications of ISO 7816-3 with certain restrictions or enhancements to increase the practical functionality of the ACR89U-A2.

2.5.1. Smart Card Power Supply VCC (C1)

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

2.5.2. 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 ACR89U-A2.

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 an 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.

2.5.3. 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).

2.6. Contactless Smart Card Interface

The interface between the ACR89U-A2 and the contactless card follows the specifications of ISO 14443 with certain restrictions or enhancements to increase the practical functionality of the ACR89U-A2.

2.6.1. Carrier Frequency

The carrier frequency for ACR89U-A2 is 13.56 MHz.

2.6.2. Card Polling

The ACR89U-A2 automatically polls the contactless tags that are within the field. ISO 14443-4 Type A, ISO 14443-4 Type B, MIFARE, FeliCa and NFC tags are supported.

3.0. ACR89 USB Communication Protocol

ACR89 interfaces with host (in PC-linked mode) with USB connection. CCID specifications have been released within the industry defining such protocol for the USB chip-card interface devices. CCID covers all the protocols required for operating smart cards and PIN. However, it does not define the protocol for operating other peripheral features that ACR89 also has. Communication protocol for ACR89 reader shall follow the CCID specifications and extend it to support the rest of the reader's features.

3.1. Device Configuration

The configurations and usage of USB end-points on ACR89 shall follow CCID Section 3. An overview is summarized below:

1. *Control Commands* are sent on control pipe (default pipe). These include class-specific requests and USB standard requests. Commands that are sent on the default pipe report information back to the host on the default pipe.
2. *CCID Events* are sent on the interrupt pipe.
3. *CCID Commands* are sent on BULK-OUT endpoint. Each command sent to ACR89 has an associated ending response. Some commands can also have intermediate responses.
4. *CCID Responses* are sent on BULK-IN endpoint. All commands sent to ACR89 have to be sent synchronously. (i.e. *bMaxCCIDBusySlots* is equal to 1 for ACR89).

The supported CCID features by ACR89 are indicated in its Class Descriptor:

Offset	Field	Size	Value	Description
0	*bLength*	1	36h	Size of this descriptor, in bytes
1	*bDescriptorType*	1	21h	CCID Functional Descriptor type
2	*bcdCCID*	2	0100h	CCID Specification Release Number in Binary-Coded decimal
4	*bMaxSlotIndex*	1	04h	Five slots are available on ACR89.
5	*bVoltageSupport*	1	07h	ACR89 can supply 1.8V, 3.0V and 5.0V to its slots
6	*dwProtocols*	4	00000003h	ACR89 supports T=0 and T=1 Protocol
10	*dwDefaultClock*	4	000012C0h	Default ICC clock frequency is 4.8 MHz
14	*dwMaximumClock*	4	000012C0h	Maximum supported ICC clock frequency is 4.8 MHz
18	*bNumClockSupported*	1	00h	Does not support manual setting of clock frequency
19	*dwDataRate*	4	003267h	Default ICC I/O data rate is 12,903 bps
23	*dwMaxDataRate*	4	00032673h	Maximum supported ICC I/O data rate is 206,451 bps
27	*bNumDataRatesSupported*	1	00h	Does not support manual setting of data rates
28	*dwMaxIFSD*	4	00000FEh	Maximum IFSD supported by ACR89 for protocol T=1 is 254

Offset	Field	Size	Value	Description
32	*dwSynchProtocols*	4	00000000h	ACR89 does not support synchronous card
36	*dwMechanical*	4	00000000h	ACR89 does not support special mechanical characteristics
40	*dwFeatures*	4	000204B2h	ACR89 supports the following features: - Automatic parameter configuration based on ATR data - Automatic ICC clock frequency change according to parameters - Automatic baud rate change according to frequency and FI, DI parameters - Automatic PPS made by the ACR89 according to the current parameters - Automatic IFSD - Short APDU level exchange with ACR89
44	*dwMaxCCIDMessageLength*	4	00000110h	Maximum message length accepted by ACR89 is 272 bytes
48	*bClassGetResponse*	1	FFh	Echo class of APDU in Get Response command
49	*bClassEnvelope*	1	FFh	Insignificant (Short APDU exchange level)
50	*wLCDLayout*	2	0815h	8 lines x 21 characters LCD
52	*bPINSupport*	1	03h	ACR89 supports PIN Verification and PIN Modification
53	*bMaxCCIDBusySlots*	1	01h	Only 1 slot can be simultaneously busy

Table 2: ACR89 Supported CCID Features Class Descriptor

Note: *Standard CCID adopts little endian mode.*

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

3.2. CCID Class-Specific Requests

ACR89's USB communication with PC is based on command message format standard of ACR89 reader. This device shall support one CCID Class-specific Request. Class-specific requests are sent via Control Pipe.

3.2.1. Command Summary

Stop any current processing command and return to a state where ACR89 is ready to accept a new command:

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001b	ABORT (01h)	*bSeq, bSlot*	Interface	0000h	None

3.3. CCID Command Pipe Bulk-Out Message

ACR89 reader follows the CCID Bulk-OUT Messages as standard CCID Session 4. In addition, this specification defines some extended commands for operating additional features. This section lists the CCID Bulk-OUT Messages to be supported by ACR89. The extended commands will be introduced in **Section 3.5**.

3.3.1. Command Summary

3.3.1.1. PC_to_RDR_IccPowerOn

Activates the card slot and returns ATR from the card.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	62h	-
1	*dwLength*	4	00000000h	Size of extra bytes of this message
2	*bSlot*	1	-	Identifies the slot number for this command.
5	*bSeq*	1	-	Sequence number for command.
6	*bPowerSelect*	1	-	Voltage that is applied to the ICC: 00h = Automatic Voltage Selection 01h = 5 volts 02h = 3 volts 03h = 1.8 volts
7	*abRFU*	2	-	Reserved for future use.

The response to this message is the *RDR_to_PC_DataBlock* message and the data returned is the Answer To Reset (ATR) data.

3.3.1.2. PC_to_RDR_IccPowerOff

Deactivates the card slot.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	63h	-
1	*dwLength*	4	00000000h	Size of extra bytes of this message
5	*bSlot*	1	-	Identifies the slot number for this command
6	*bSeq*	1	-	Sequence number for command
7	*abRFU*	3	-	Reserved for future use

The response to this message is the *RDR_to_PC_SlotStatus* message.

3.3.1.3. PC_to_RDR_GetSlotStatus

Gets the current status of the slot.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	65h	-
1	*dwLength*	4	00000000h	Size of extra bytes of this message

Offset	Field	Size	Value	Description
5	*bSlot*	1	-	Identifies the slot number for this command
6	*bSeq*	1	-	Sequence number for command
7	*abRFU*	3	-	Reserved for future use

The response to this message is the *RDR_to_PC_SlotStatus* message.

3.3.1.4. PC_to_RDR_XfrBlock

Transfer data block to the ICC.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	6Fh	-
1	*dwLength*	4	-	Size of *abData* field of this message
5	*bSlot*	1	-	Identifies the slot number for this command
6	*bSeq*	1	-	Sequence number for command
7	*bBWI*	1	-	Used to extend the CCIDs Block Waiting Timeout for this current transfer. The CCID will timeout the block after "this number multiplied by the Block Waiting Time" has expired.
8	*wLevelParameter*	2	0000h	RFU (TPDU exchange level)
10	*abData*	Byte array	-	Data block sent to the CCID. Data is sent "as is" to the ICC (TPDU exchange level).

The response to this message is the *RDR_to_PC_DataBlock* message.

3.3.1.5. PC_to_RDR_GetParameters

Gets the slot parameters.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	6Ch	-
1	*dwLength*	4	00000000h	Size of extra bytes of this message
5	*bSlot*	1	-	Identifies the slot number for this command
6	*bSeq*	1	-	Sequence number for command
7	*abRFU*	3	-	Reserved for future use

The response to this message is the *RDR_to_PC_Parameters* message.

3.3.1.6. PC_to_RDR_ResetParameters

Resets the slot parameters to default value.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	6Dh	-
1	*dwLength*	4	00000000h	Size of extra bytes of this message

Offset	Field	Size	Value	Description
5	*bSlot*	1	-	Identifies the slot number for this command
6	*bSeq*	1	-	Sequence number for command
7	*abRFU*	3	-	Reserved for future use

The response to this message is the *RDR_to_PC_Parameters* message.

3.3.1.7. PC_to_RDR_SetParameters

Sets slot parameters.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	61h	-
1	*dwLength*	4	-	Size of extra bytes of this message
5	*bSlot*	1	-	Identifies the slot number for this command
6	*bSeq*	1	-	Sequence number for command.
7	*bProtocolNum*	1	-	Specifies what protocol data structure follows: 00h = Structure for protocol T=0 01h = Structure for protocol T=1 The following values are reserved for future use: 80h = Structure for 2-wire protocol 81h = Structure for 3-wire protocol 82h = Structure for I2C protocol
8	*abRFU*	2	-	Reserved for future use
10	*abProtocolDataStructure*	Byte array	-	Protocol Data Structure

Protocol Data Structure for Protocol T=0 (dwLength=00000005h)

Offset	Field	Size	Value	Description
10	*bmFindexDindex*	1	-	B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor B3-0 – DI - Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor
11	*bmTCCKST0*	1	-	B0 – 0b, B7-2 – 000000b B1 – Convention used (b1=0 for direct, b1=1 for inverse) ***Note:*** *The CCID ignores this bit.*
12	*bGuardTimeT0*	1	-	Extra Guardtime between two characters. Add 0 to 254 etu to the normal guardtime of 12etu. FFh is the same as 00h.
13	*bWaitingIntegerT0*	1	-	WI for T=0 used to define WWT

Offset	Field	Size	Value	Description
14	*bClockStop*	1	-	ICC Clock Stop Support: 00h = Stopping the Clock is not allowed 01h = Stop with Clock signal Low 02h = Stop with Clock signal High 03h = Stop with Clock either High or Low

Protocol Data Structure for Protocol T=1 (*dwLength*=00000007h)

Offset	Field	Size	Value	Description
10	*bmFindexDindex*	1	-	B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor B3-0 – DI - Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor
11	*BmTCCKST1*	1	-	B7-2 – 000100b B0 – Checksum type (b0=0 for LRC, b0=1 for CRC B1 – Convention used (b1=0 for direct, b1=1 for inverse) ***Note:*** *The CCID ignores this bit.*
12	*BGuardTimeT1*	1	-	Extra Guardtime (0 to 254 etu between two characters). If value is FFh, then guardtime is reduced by 1 etu.
13	*BWaitingIntegerT1*	1	-	B7-4 = BWI values 0-9 valid B3-0 = CWI values 0-Fh valid
14	*bClockStop*	1	-	ICC Clock Stop Support: 00h = Stopping the Clock is not allowed 01h = Stop with Clock signal Low 02h = Stop with Clock signal High 03h = Stop with Clock either High or Low
15	*bIFSC*	1	-	Size of negotiated IFSC
16	*bNadValue*	1	00h	Only supports NAD = 00h

The response to this message is the *RDR_to_PC_Parameters* message.

3.3.1.8. PC_to_RDR_Escape

This command allows ACR89 to use the extended features as defined in **Section 3.5**.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	6Bh	-
1	*DwLength*	4	-	Size of *abData* field of this message
5	*Bslot*	1	-	Identifies the slot number for this command
6	*Bseq*	1	-	Sequence number for command
7	*AbRFU*	3	-	Reserved for future use

Offset	Field	Size	Value	Description
10	*AbData*	Byte array	-	Commands specified in **Section 3.5.2.**

The response to this message is the *RDR_to_PC_Escape* message.

This message could return any of the following ACR89 specific errors. Further qualification of error is provided in the extended response.

bmICCStatus	bmCommand Status	bError	Description
3	1	ACR89_ERROR	ACR89 specific error. Refer to *wReturnCode* in ACR89 response.
3	1	INVALID_MODE	ACR89 is operating in a mode that does not support this command
3	1	DEVICE_VOID	ACR89 is not initialized

Table 3: PC_to_RDR_Escape Extended Response

3.3.1.9. PC_to_RDR_Secure (RFU)

The command is reserved for future implementation.

This is a command message to allow entering the PIN for verification or modification on the card directly.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	69h	-
1	*DwLength*	4	-	Size of extra bytes of this message
5	*BSlot*	1	-	Identifies the slot number for this command
6	*BSeq*	1	-	Sequence number for command
7	*BBWI*	1	-	Used to extend the CCIDs Block Waiting Timeout for this current transfer. The CCID will timeout the block after “this number multiplied by the Block Waiting Time” has expired. This parameter is only used for character level exchanges.
8	*wLevelParameter*	2	0000h	RFU (TPDU exchange level)

Offset	Field	Size	Value	Description
10	*bPINOperation*	1	-	Used to indicate the PIN operation: 00h: PIN Verification 01h: PIN Modification 02h: Transfer PIN from secure CCID buffer 03h: Wait ICC response 04h: Cancel PIN function 05h: Re-send last I-Block, valid only if protocol in use is T=1 06h: Send next part of APDU, valid only if protocol in use is T=1
11	*abPINDataStructure*	Byte array	-	PIN Verification Data Structure or PIN Modification Data Structure

The response to this message is the *RDR_to_PC_DataBlock*.

Note: *Refer to standard CCID Session 4.1.11 for detail PIN Verification Data Structure and PIN Modification Data Structure.*

3.3.1.10. PC_to_RDR_Abort

This command is used with the Control Pipe Abort request to tell the CCID to stop any current transfer at the specified slot and return to a state where the slot is ready to accept a new command pipe Bulk-OUT message.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	72h	-
1	*DwLength*	4	00000000h	Size of extra bytes of this message
5	*BSlot*	1	-	Identifies the slot number for this command
6	*BSeq*	1	-	Sequence number for command
7	*AbRFU*	3	000000h	RTF

The response to this message is the *RDR_to_PC_SlotStatus* message.

3.4. CCID Command Pipe Bulk-IN Message

The Bulk-IN messages are used in response to the Bulk-OUT messages. ACR89 shall follow the CCID Bulk-IN Messages as specified in standard CCID session 4. This section lists the CCID Bulk-IN Messages to be supported by ACR89.

3.4.1. Message Summary

3.4.1.1. RDR_to_PC_DataBlock

This message is sent by ACR89 in response to *PC_to_RDR_IccPowerOn*, *PC_to_RDR_XfrBlock* and *PC_to_RDR_Secure* messages.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	80h	Indicates that a data block is being sent from the CCID
1	*dwLength*	4	-	Size of extra bytes of this message
5	*BSlot*	1	-	Same value as in Bulk-OUT message
6	*BSeq*	1	-	Same value as in Bulk-OUT message
7	*bStatus*	1	-	Slot status and error register as defined in **Section 3.7**
8	*bError*	1	-	Slot status and error register as defined in **Section 3.7**
9	*bChainParameter*	1	00h	RFU (TPDU exchange level)
10	*AbData*	Byte array	-	This field contains the data returned by the CCID

3.4.1.2. RDR_to_PC_SlotStatus

This message is sent by ACR89 in response to *PC_to_RDR_IccPowerOff*, *PC_to_RDR_GetSlotStatus*, *PC_to_RDR_Abort* messages and class-specific ABORT request.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	81h	-
1	*dwLength*	4	00000000h	Size of extra bytes of this message
5	*BSlot*	1	-	Same value as in Bulk-OUT message
6	*BSeq*	1	-	Same value as in Bulk-OUT message
7	*bStatus*	1	-	Slot status and error register as defined in **Section 3.7**
8	*bError*	1	-	Slot status and error register as defined in **Section 3.7**
9	*bClockStatus*	1	-	Value: 00h = Clock running 01h = Clock stopped in state L 02h = Clock stopped in state H 03h = Clock stopped in an unknown state All other values are RFU

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

3.4.1.3. RDR_to_PC_Parameters

This message is sent by ACR89 in response to *PC_to_RDR_GetParameters*, *PC_to_RDR_ResetParameters* and *PC_to_RDR_SetParameters* messages.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	82h	-
1	*dwLength*	4	-	Size of extra bytes of this message
5	*bSlot*	1	-	Same value as in Bulk-OUT message
6	*bSeq*	1	-	Same value as in Bulk-OUT message
7	*bStatus*	1	-	Slot status and error register as defined in **Section 3.7**
8	*bError*	1	-	Slot status and error register as defined in **Section 3.7**
9	*bProtocolNum*	1	-	Specifies what protocol data structure follows. 00h = Structure for protocol T=0 01h = Structure for protocol T=1 The following values are reserved for future use: 80h = Structure for 2-wire protocol 81h = Structure for 3-wire protocol 82h = Structure for I2C protocol
10	*abProtocolDataStructure*	Byte array	-	Protocol Data Structure as summarized in standard CCID Session 5.2.3

3.4.1.4. RDR_to_PC_Escape

This message is sent by ACR89 in response to *PC_to_RDR_Escape* message.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	83h	-
1	*dwLength*	4	-	Size of extra bytes of this message
5	*bSlot*	1	-	Same value as in Bulk-OUT message
6	*bSeq*	1	-	Same value as in Bulk-OUT message
7	*bStatus*	1	-	Slot status and error register as defined in **Section 3.7**
8	*bError*	1	-	Slot status and error register as defined in **Section 3.7**
9	*bRFU*	1	00h	RFU
10	*abData*	Byte array	-	Depending on its corresponding extended command, the data responded by ACR89 vary and are specified in **Section 3.5.4**.

3.5. Extended Command Pipe Message Compatible with ACR89

This section defines the extended commands to be accepted by ACR89 for operating additional features that CCID does not cover. These commands are always executed under the command *PC_to_RDR_Escape* Bulk-OUT message and responded with *RDR_to_PC_Escape* Bulk-IN message.

PC Request Message	Code	ACR89 Response Message	Code
PC_to_ACR89_InputKey	12h	ACR89_to_PC_DataBlock	81h
PC_to_ACR89_SetCursor	18h	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_SetBacklight	19h	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_DisplayMessage	1bh	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_DisplayRowGraphic	23h	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_SetContrast	1ch	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_ClearDisplay	1dh	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_ReadRTC	08h	ACR89_to_PC_TimeStamp	84h
PC_to_ACR89_SetRTC	09h	ACR89_to_PC_TimeStamp	84h
PC_to_ACR89_Buzzer	0ah	ACR89_to_PC_Echo	90h
PC_to_ACR89_AccessEeprom	21h	ACR89_to_PC_Datablock	81h
PC_to_ACR89_SetLED	22h	ACR89_to_PC_Echo	90h
PC_to_ACR89_EraseSPIFlash	30h	ACR89_to_PC_ExMemStatus	b0h
PC_to_ACR89_ProgramSPIFlash	33h	ACR89_to_PC_MemoryStatus	b0h
PC_to_ACR89GetSPIFlash	34h	ACR89_to_PC_MemoryPage	b1h
PC_to_ACR89_GetVersion	36h	ACR89_to_PC_VersionInfo	b2h
PC_to_ACR89_AuthoInfo	38h	ACR89_to_PC_AuthInfo	b4h

Table 4: ACR89 Extended Command Pipe Messages

3.5.1. Extended Command Pipe Bulk-OUT Message

The command format defined in this section will be the *abData* field to be filled in the *PC_to_RDR_Escape* message.

Similar to the CCID message structure, the command format consists of fixed length Command Header and variable length Command Data portion. The command header is fixed to 5 bytes in length.

In contrast to CCID/USB practice, big endian will be adopted in extended command portion.

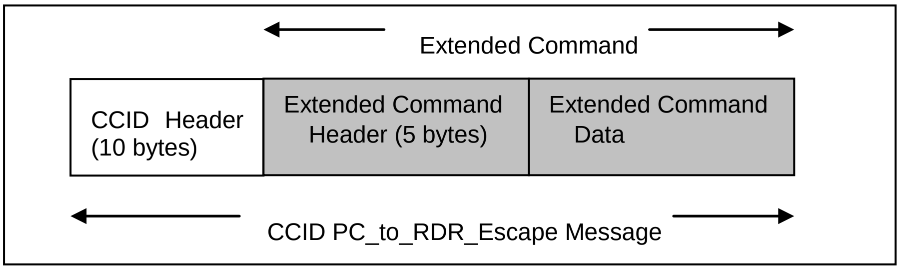

Figure 2: CCID PC_to_RDR_Escape Message

3.5.2. Commands Detail

3.5.2.1. PC_to_ACR89_InputKey

This command accepts key(s) input from the user using keypad. This command context is slot independent.

Offset	Field Name	Type	Size	Value	Description
10	*BCmdCode*	Hex	1	12h	-
11	*wCmdLength*	Hex	2	0002h	Size of command data (in big endian)
13	*AbRfu*	Hex	2	0000h	-
15	*bKeyInputMode*	Bin	1	-	B0 – Input mode (b0=0 for single key input, b0=1 for key string input). In key string input mode, the key string input is considered completed when “Enter” key is pressed. B1 – Keyboard mode (b1=0 for numeric input, b1=1 for alphanumeric input) B3 to b2 – Key display (b2=0 for key display disabled, b2=1 for key display enabled. When b2=1, b3=0 for key display as plaintext, b3=1 for key display as ‘*’) B4 – Key input timeout control (b4=0 for timeout enabled, b4=1 for timeout disabled) B5 – Secure key transfer (b5=0 for plaintext transfer, b5=1 for encrypted key transfer) *This bit is reserved for future implementation*. B6 – 0/1 – disable/enable control key b7 – RFU
16	*bTimeoutValue*	Hex	1	-	Key input timeout time value counted in second. Effective only when key input timeout control bit of *bKeyInputMode* field is 0.

The response to this command is the *ACR89_to_PC_DataBlock* message.

3.5.2.2. PC_to_ACR89_SetCursor

This command sets the LCD position cursor to a new position. This command context is slot independent.

Offset	Field Name	Type	Size	Value	Description
10	*BcmdCode*	Hex	1	18h	-
11	*wCmdLength*	Hex	2	0002h	Size of command data (in big endian)

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

Offset	Field Name	Type	Size	Value	Description
13	*AbRfu*	Hex	2	0000	Reserved for future
15	*bRowPosition*	Hex	1	00h to 07h	New cursor row position
16	*bColumnPosition*	Hex	1	00h to 7Fh	New cursor column position

The response to this command is the *ACR89_to_PC_DisplayStatus* message.

3.5.2.3. PC_to_ACR89_SetBacklight

This command configures the LCD display. This command context is slot independent.

Offset	Field Name	Type	Size	Value	Description
10	*BCmdCode*	Hex	1	19h	-
11	*wCmdLength*	Hex	2	0001h	Size of command data (in big endian)
13	*AbRfu*	Hex	2	0000	Reserved for future
15	*BBacklight*	Hex	1	00h or 01h	00h - turns off backlight 01h - turns on backlight Others values RFU

The response to this command is the *ACR89_to_PC_DisplayStatus* message.

3.5.2.4. PC_to_ACR89_DisplayMessage

This command displays a string of characters from ACR89 build-in font library. The string will be displayed horizontally from the current cursor position. ACR89 will automatically calculate the absolute coordinates from the character position and character size. The cursor will move accordingly. This command context is slot dependent.

Offset	Field Name	Type	Size	Value	Description
10	*BCmdCode*	Hex	1	1Bh	-
11	*wCmdLength*	Hex	2	Var…	Size of command data (in big endian)
13	*AbRfu*	Hex	2	0000h	Reserved for future
15	*bCharCoding*	Hex	1	-	Data encoding format in *abData* field. Character size depends on data format: 00h – ASCII (1 row by 6 column per character) All other values are RFU
16	*AbData*	Ascii	Byte array	-	Character string of encoding format stated in *bCharCoding* field

The response to this command is the *ACR89_to_PC_DisplayStatus* message.

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

3.5.2.5. PC_to_ACR89_DisplayRowGraphic

This command scans a row of graphics to be displayed on LCD.

Offset	Field Name	Type	Size	Value	Description
10	*bCmdCode*	Hex	1	23h	-
11	*wCmdLength*	Hex	2	Var…	Size of command data (in big endian)
13	*abRfu*	Hex	2	0000h	-
15	*bRowPosition*	Hex	1	-	Start position row index. One row is with height of 8 pixels.
16	*bColumnPosition*	Hex	1	-	Start position column index
17	*AbData*	Hex	Var	-	Bitmap data of a row of the graphic to be displayed

The sum of *wCmdLength* and *bColumnPosition* cannot exceed the column number of LCD (128).

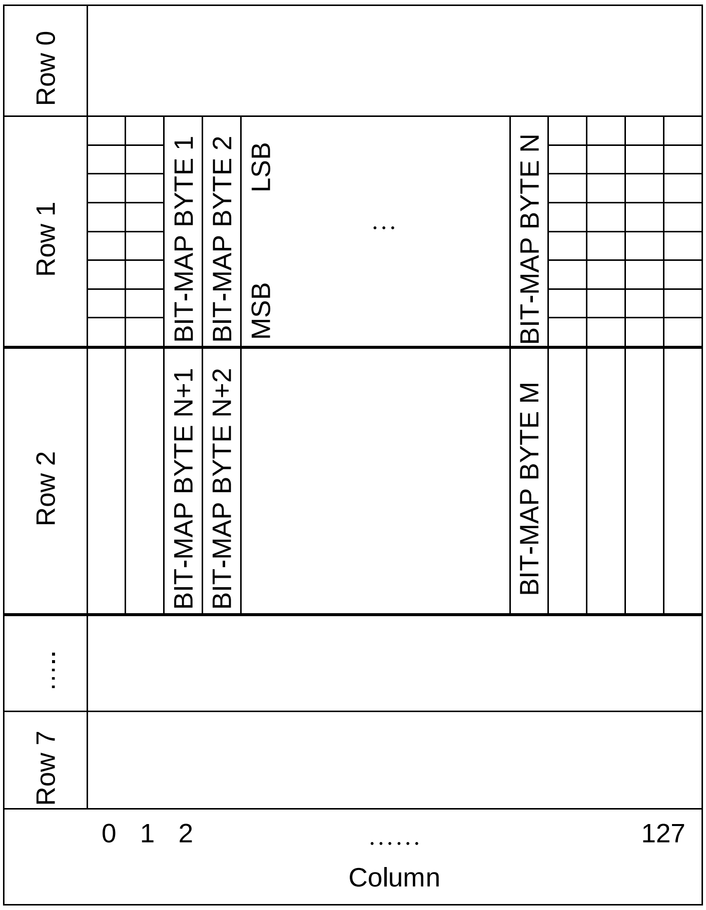

Figure 3: PC_to_ACR89_DisplayGraphic – Bitmap Format

The response to this command is the *ACR89_to_PC_DisplayStatus* message.

3.5.2.6. PC_to_ACR89_SetContrast

This command sets the contracts level of the LCD. This command context is slot independent.

Offset	Field Name	Type	Size	Value	Description

Offset	Field Name	Type	Size	Value	Description
10	*BCmdCode*	Hex	1	1Ch	-
11	*wCmdLength*	Hex	2	0001h	Size of command data (in big endian)
13	*abRfu*	Hex	2	0000	Reserved for future
15	*bContrastLevel*	Hex	1	00h to 0x63h	New LCD contrast level

The value range is between 00h to 63h. Larger the value darkens the contrast. Lower range, on the other hand, brightens the contrast. The whole LCD display and image affects the contrast level.

3.5.2.7. PC_to_ACR89_ClearDisplay

This command clears one or more rows on the LCD display. The cursor will be moved to the position at the starting point of the cleared block after executing this command. This command context is slot independent.

Offset	Field Name	Type	Size	Value	Description
10	*BcmdCode*	Hex	1	1Dh	-
11	*wCmdLength*	Hex	2	0002h	Size of command data (in big endian)
13	*AbRfu*	Hex	2	0000h	Reserved for future
15	*bClearMode*	Hex	1	00h or 01h or 02h	00h = Clear full screen 01h = Clear the row located by the current position cursor 02h = Clear some columns in a row starting from current position cursor All other values RFU
16	*bNumber*	-	1	-	For *bClearMode* = 01h – Number of rows to be cleared For *bClearMode* = 02h – Number of columns to be cleared Not significant otherwise

The response to this command is the *ACR89_to_PC_DisplayStatus* message.

3.5.2.8. PC_to_ACR89_ReadRTC

This command reads the current real time clock value from the built-in real time clock. The RTC increments the value every half second. This command context is slot independent.

Offset	Field Name	Type	Size	Value	Description
10	*BCmdCode*	Hex	1	08h	-
11	*wCmdLength*	Hex	2	0000h	Size of command data (in big endian)
13	*AbRFU*	Hex	2	0000h	-

The response to this command is the *ACR89_to_PC_TimeStamp* message.

3.5.2.9. PC_to_ACR89_SetRTC

This command sets the real time clock value of the build-in real time clock to a specified value. This command context is slot independent.

Offset	Field Name	Type	Size	Value	Description
10	*BCmdCode*	Hex	1	09h	-
11	*wCmdLength*	Hex	2	0006h	Size of command data (in big endian)
13	*AbRFU*	Hex	2	0000	-
15	*bRTCValue*	BCD	6	-	New real time clock value. Format in YY, MM, DD, HH, MI and SS.

The response to this command is the *ACR89_to_PC_TimeStamp* message.

3.5.2.10. PC_to_ACR89_Buzzer

Offset	Field Name	Type	Size	Value	Description
10	*BCmdCode*	Hex	1	0Ah	-
11	*wCmdLength*	Hex	2	0002h	Size of command data (in big endian)
13	*abRfu*	Hex	2	0000h	-
15	*bBuzzerState*	Hex	1	--	01h – Buzzer on 00h - Buzzer off
16	*BbuzzerOnDuration*	Hex	1	-	Buzzer on duration in number of hundredth milliseconds. Effective only when *bBuzzerState* field is 01h. 00h – Activate buzzer and do not turn off the buffer Other value – Activate buzzer for number of hundredth milliseconds and then turn off the buzzer

The response to this command is the *ACR89_to_PC_Echo* message.

3.5.2.11. PC_to_ACR89_AccessEeprom

This command allows user write or read data from the EEPROM. Maximum allow data length is 249 Bytes.

Offset	Field Name	Type	Size	Value	Description
10	*bCmdCode*	Hex	1	21h	-
11	*wCmdLength*	Hex	2	Var…	Size of command data (in big endian)
13	*AbRFU*	Hex	2	0000h	-
15	*bAccessMode*	Ascii	1	-	'W' – write EEPROM 'R' – read EEPROM
16	*BDeviceNumber*	Hex	1	-	00 – Slave EEPROM 01- Chinese Font EEPROM (Rfu)
17	*AbAddress*	Hex	4	-	Address of EEPROM (in big endian)
21	*wDataLength*	Hex	2	Var…	Length of Data (Write/Read) (in big endian)

Offset	Field Name	Type	Size	Value	Description
23	*bEeprom Data*	Hex	Var..	-	EEPROM data

The response to this command is the *ACR89_to_PC_DataBlock* message.

3.5.2.12. PC_to_ACR89_SetLED

The command allows user to switch on/off of Power, slot1 and slot2 on card reader with color red and green.

Offset	Field Name	Type	Size	Value	Description
10	*BcmdCode*	Hex	1	22h	-
11	*WcmdLength*	Hex	2	0003h	Size of command data (in big endian)
13	*AbRFU*	Hex	2	0000h	-
15	*Power LED*	Hex	1	-	Bit0 : 1- Selects Red color Bit1 : 1- Selects Green color Bit2 : 1- Selects Yellow color Bit7 : 0-OFF/1-ON e.g. Turn ON red color 10000001b Turn OFF green color 00000010b Ignore xxxx0000b
16	*Slot1 LED*	Hex	1	-	Bit0 : 1- Selects Red color Bit1 : 1- Selects Green color Bit2 : 1- Selects Yellow color Bit7 : 0-OFF/1-ON
17	*Slot2 LED*	Hex	1	-	Bit0 : 1- Selects Red color Bit1 : 1- Selects Green color Bit2 : 1- Selects Yellow color Bit7 : 0-OFF/1-ON

The response to this command is *ACR89_to_PC_Echo*.

3.5.2.13. PC_to_ACR89_EraseSPIFlash

This command erases flash blocks.

Offset	Field Name	Type	Size	Value.	Description
10	*bCmdCode*	Hex	1	30h	Command Code
11	*bFlashType*	Hex	1	02h	SPI flash
12	*bRFU*	Hex	1	00h	-
13	*bStartBlockNum*	Hex	1	-	Any number not zero, e.g. 01h but less than 08h (if default size of serial flash is used)
14	*bEndBlockNum*	Hex	1	-	Not less than *bStartBlockNum*

The response to this command is the *ACR89_to_PC_ExMemStatus* message.

***Note**: The current size of one flash block is 64k bytes.*

3.5.2.14. PC_to_ACR89_ProgramSPIFlash

This command writes 256 bytes data to a page of the SPI flash.

Offset	Field Name	Type	Size	Value	Description
10	*bCmdCode*	Hex	1	33h	Command Code
11	*AbAddress*	Hex	4	xxxxxx00h	Start address of flash page (in little endian)
15	*AbData*	Hex	256	-	Data write to a flash page
271	*bCheckSum*	Hex	1	-	Checksum of *AbData*

The response to this command is the *ACR89_to_PC_ExMemStatus* message.

3.5.2.15. PC_to_ACR89_GetSPIFlashPage

This command reads 256 bytes data from a page of the SPI flash.

Offset	Field Name	Type	Size	Value	Description
10	*bCmdCode*	Hex	1	34h	Command Code
11	*AbAddress*	Hex	4	xxxxxx00h	Start address of flash page (in little endian)

The response to this command is the *ACR89_to_PC_MemoryPage* message.

3.5.2.16. PC_to_ACR89_GetVersion

This command reads boot loader or application firmware version information.

Offset	Field Name	Type	Size	Value	Description
10	*bCmdCode*	Hex	1	36h	Command Code
11	*bVersionType*	Hex	1	-	01h = boot loader version 02h = application version
12	*AbRFU*	Hex	3	000000h	-

The response to this command is the *ACR89_to_PC_VersionInfo* message.

3.5.2.17. PC_to_ACR89_ AuthInfo

This command reads RomID and RomData.

Offset	Field Name	Type	Size	Value	Description
10	*bCmdCode*	Hex	1	38h	Command Code
11	*AbRFU*	Hex	16	00…00h	-

The response to this command is the *ACR89_to_PC_AuthInfo* message.

3.5.3. Extended Command Pipe Bulk-IN Message

This section defines response messages to the extended commands returned by ACR89 for operating additional features that CCID does not cover. These messages are always responded using *RDR_to_PC_Escape* Bulk-IN message in standard CCID Session 4.2.2.4.

The response format defined in this section will be the *abData* to be filled in the *RDR_to_PC_Escape* messages. Similar to CCID message structure, the response format consists of fixed length Response Header and variable length Response Data portion. The response header is fixed to 5 bytes in length.

In contrast to CCID/USB practice, big endian will be adopted in extended response portion.

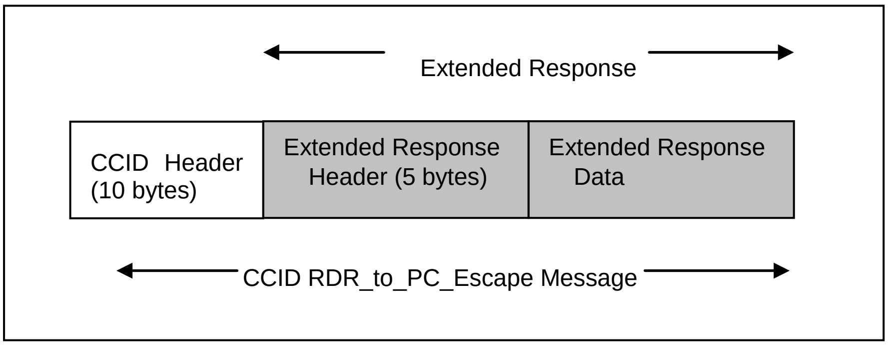

Figure 4: CCID RDR_to_PC_Escape Message

3.5.4. Messages Detail

3.5.4.1. ACR89_to_PC_DataBlock

This message is sent by ACR89 in response to *PC_to_ACR89_InputKey* commands.

For *PC_to_ACR89_InputKey* command, the data returned is the single key or key string captured from the keypad, depending on the key input mode chosen.

Offset	Field Name	Size	Value	Description
10	*BrespType*	1	81h	-
11	*WReturnCode*	2	-	Command response code (in big endian)
13	*WRespLength*	2	Var…	Size of response data (in big endian)
15	*Bdata*	Var …	-	This field contains the data returned by ACR89

3.5.4.2. ACR89_to_PC_DisplayStatus

This message is sent by ACR89 in response to *PC_to_ACR89_DisplaySetCursor, PC_to_ACR89_DisplayMessage, PC_to_ACR89_DisplayRowGraphic* and *PC_to_ACR89_ClearDisplay* commands.

Offset	Field Name	Size	Value	Description
10	*BrespType*	1	83h	-
11	*wReturnCode*	2	-	Command response code (in big endian)
13	*wRespLength*	2	0002h	Size of response data (in big endian)
15	*bRowPosition*	1	00h to 07h	Current cursor row position
16	*bColumnPosition*	1	00h to 7Fh	Current cursor column position

3.5.4.3. ACR89_to_PC_TimeStamp

This message is sent by ACR89 in response to *PC_to_ACR89_ReadRTC* and *PC_to_ACR89_SetRTC* commands.

Offset	Field Name	Size	Value	Description
10	*BRespType*	1	84h	-
11	*wReturnCode*	2	-	Command response code (in big endian)
13	*wRespLength*	2	0006h	Size of response data (in big endian)
15	*bTimeStamp*	6	-	Current real time clock value. Format in YY, MM, DD, HH, MI and SS

3.5.4.4. ACR89_to_PC_Echo

This message is sent by ACR89 in response to *PC_to_ACR89_Buzzer*, *PC_to_ACR89_SetLED* and *PC_to_ACR89_ExitScriptMode* commands.

Offset	Field	Size	Value	Description
10	*bRespType*	1	90h	-

Offset	Field	Size	Value	Description
11	*wReturnCode*	2	90 00h	Command response code, If command success, it returns 90 00h (in big endian)
13	*wRespLength*	2	0000	Size of response data (in big endian)

3.5.4.5. ACR89_to_PC_ExMemStatus

This message is sent by ACR89 in response to *PC_to_ACR89_EraseSPIFlash,* and *PC_to_ACR89_ProgramSPIFlash* command.

Offset	Field Name	Size	Value	Description
10	*bRespType*	1	B0h	-
11	*bReturnState*	1	-	Command return state (please refer to later section).
12	*bErrorCode*	1	-	Error code (please refer to later section).
13	*AbRFU*	2	0000h	-

3.5.4.6. ACR89_to_PC_MemoryPage

This message is sent by ACR89 in response to *PC_to_ACR89_GetSPIFlashPage* commands.

Offset	Field Name	Size	Value	Description
10	*bRespType*	1	B1h	-
11	*bReturnState*	1	-	Command return state (please refer to later section)
12	*bErrorCode*	1	-	Error code (please refer to later section)
13	*AbRFU*	2	0000h	-
15	*AbData*	256	-	Data read from a flash page
271	*bCheckSum*	Hex	1	Checksum of *AbData*

***Note**: There will be no AbData and bCheckSum parts when command failed.*

3.5.4.7. ACR89_to_PC_VersionInfo

This message is sent by ACR89 in response to *PC_to_ACR89_GetVersion* command.

Offset	Field Name	Size	Value	Description
10	*bRespType*	1	B2h	-
11	*bReturnState*	1	-	Command return state (please refer to later section)
12	*bErrorCode*	1	-	Error code (please refer to later section)
13	*wInfoLength*	2	Var	Size of *bInfoData* (in little endian)
15	*bInfoData*	Var	-	Firmware version information (ASCII)

***Note**: The wInfoLength is zero when there is no valid version information.*

3.5.4.8. ACR89_to_PC_AuthInfo

This message is sent by ACR89 in response to *PC_to_ACR89_AuthInfo* commands.

Offset	Field Name	Size	Value	Description
10	*bRespType*	1	B4h	-
11	*bReturnState*	1	-	Command return state (please refer to later section)
12	*bErrorCode*	1	-	Error code (please refer to later section)
13	*AbRFU*	2	0000h	-
15	*AbRomID*	8	-	Unique ID
23	*AbRFU*	48	-	-

***Note**: There will be no parts from offset 15 when command failed.*

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

3.5.5. Extended Command Response Codes and Return States

The table summarizes the response code and the return states for the CCID extended commands used by ACR89.

Response Code	Value	Description
CMD_OKAY	9000h	Command executes successfully
INVALID_PARAMETERS	FFFFh	Wrong parameters in the extended command
INVALID_COMMAND_CODE	FFFEh	Command code in the extended command (offset 10) is invalid
INVALID_COMMAND_LENGTH	FFFDh	Wrong length in the extended command
CANNOT_EXECUTE_COMMAND	FFFCh	Extended command cannot be executed
TIMEOUT	FFFBh	Timeout for executing the extended command
SCRIPT_ERROR	FFFAh	Cannot execute the script

Table 5: Extended Command Response Codes

Return State	Value	Description
CMD_OK	00h	Command executes successfully
CMD_FAIL	01h	Command execution failed

Table 6: Extended Command Return States

Error Code	Value	Description
COMMAND_NOT_SUPPORT	00h	Command code in the extended command (offset 10) is not supported
HARDWARE_ERROR	01h	Hardware error occurred
ACCESS_DENIED	02h	Function is denied according to current configuration
ADDRESS_ERROR	03h	Address parameter is not correct
FRAME_ERROR	04h	Command frame format is not correct
CHECKSUM_ERROR	05h	Check sum for data part is not correct

Table 7: Extended Command Error Codes

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

3.6. CCID Interrupt-IN Message

The Interrupt-IN endpoint is used to notify the host of events that may occur asynchronously and outside the context of a command-response exchange between host and ACR89. ACR89 shall follow the CCID Interrupt-IN Messages as specified in standard CCID session 4. This section lists the CCID Interrupt-IN Messages to be supported by ACR89.

3.6.1. Message Summary

3.6.1.1. RDR_to_PC_NotifySlotChange

This message is sent whenever ACR89 detects a change in the insertion status of an ICC slot.

Offset	Field	Size	Value	Description
0	*bMessageType*	1	50h	-
1	*bmSlotICCState*	-	-	This field is reported on byte granularity. The size is (2 bits * number of slots) rounded up to the nearest byte. Each slot has 2 bits. The least significant bit reports the current state of the slot (0b= no ICC present, 1b = ICC present). The most significant bit reports whether the slot has changed state since the last *RDR_to_PC_NotifySlotChange* message was sent (0b = no change, 1b = change). If no slot exists for a given location, the field returns 00b in those 2 bits. **Example:** A 3 slot CCID reports a single byte with the following format: Bit 0 = Slot 0 current state Bit 1 = Slot 0 changed status Bit 2 = Slot 1 current state Bit 3 = Slot 1 changed status Bit 4 = Slot 2 current state Bit 5 = Slot 2 changed status Bit 6 = 0b Bit 7 = 0b

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

3.7. CCID Error and Status Code

This section is the extension of standard CCID session 12 to tabulate the possible error codes to be used in conjunction with the slot error register in each Bulk-IN message. The table summarizes the CCID defined error codes and the additionally defined error codes for the extended commands used by ACR89.

Error Name	Error Code	Possible Cause
CMD_ABORTED	FFh	Host aborted the current activity
ICC_MUTE	FEh	CCID timed out while talking to the ICC
XFR_PARITY_ERROR	FDh	Parity error while talking to the ICC
XFR_OVERRUN	FCh	Overrun error while talking to the ICC
HW_ERROR	FBh	An all-inclusive hardware error occurred
BAD_ATR_TS	F8h	
BAD_ATR_TCK	F7h	
ICC_PROTOCOL_NOT_SUPPORTED	F6h	
ICC_CLASS_NOT_SUPPORTED	F5h	
PROCEDURE_BYTE_CONFLICT	F4h	
DEACTIVATED_PROTOCOL	F3h	
BUSY_WITH_AUTO_SEQUENCE	F2h	Automatic Sequence Ongoing
PIN_TIMEOUT	F0h	
PIN_CANCELLED	EFh	
CMD_SLOT_BUSY	E0h	A second command was sent to a slot, which was already processing a command
ACR89_ERROR	10h	Error code defined in ACR89 response header instead of this error register
DEVICE_VOID	11h	ACR89 is not initialized. Either in manufacturer mode waiting for vendor personalization or the device has been tampered.
INVALID_SECRET_KEY	12h	Wrong secret key is presented
INVALID_MODE	13h	Tried running a command that the current operation mode does not allow
Reserved for future use		(All the rest unmentioned values)

Table 8: CCID Error and Status Code

4.0. Software Design

4.1. Contactless Smart Card Protocol

4.1.1. ATR Generation

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

4.1.1.1. ATR Format for ISO 14443 Part3 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
	SSh		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

Table 9: ISO 14443 Part 3 ATR Format

Example:

ATR for MIFARE 1K = {3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6Ah}

ATR											
Initial Header	**T0**	**TD1**	**TD2**	**T1**	**Tk**	**Length**	**RID**	**Standard**	**Card Name**	**RFU**	**TCK**
3Bh	8Fh	80h	01h	80h	4Fh	0Ch	A0 00 00 03 06h	03h	00 01h	00 00 00 00h	6Ah

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

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
		FF SAK undefined tags

4.1.1.2. 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 specification. ISO 14443B: The higher layer response from the ATTRIB response (ATQB). Refer to the ISO 14443-3 specification.
	XX XX XXh	Tk	
4+N	UUh	TCK	Exclusive-oring of all the bytes T0 to Tk

Table 10: ISO 14443 Part 4 ATR Format

Example 1: Consider the ATR from MIFARE DESFire as follows:

MIFARE 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: Consider the ATR from ST19XRC8E, which is as follows:

ST19XRC8E (ATR) = 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

4.1.2. Pseudo APDUs for Contactless Interface

4.1.2.1. Direct Transmit with ACR89U-A2 Format

To send a Pseudo APDU (Contactless Chip and Tag commands), and the Response Data will be returned.

Direct Transmit Command Format (Length of the Contactless Chip and Tag Command + 5 Bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Direct Transmit	FFh	00h	00h	00h	Number of Bytes to send	Contactless Chip and TAG Command	Data

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.

Direct Transmit Response Format (Contactless Chip and Tag Response + Data + 2 Bytes)

Item	Command	Data			Meaning
1	D4 40	Tg		[DataOut[]]	Tag Exchange Data
2	D4 4A	MaxTg	BrTy	[InitiatorData[]]	Tag Polling

Where:

Tg 1 Byte. A byte containing the logical number of the relevant target. This byte also contains the More Information (MI) bit (bit 6). When the MI bit is set to 1, this indicates that the host controller wants to send more data which is all the data contained in the DataOUT[] array. This bit is only valid for a TPE target.

DataOut 0-262 Bytes. An array of raw data (from 0 up to 262 bytes) to be sent to the target by the contactless chip.

MaxTg Maximum number of targets to be initialized by the contactless chip. The chip is capable of handling a maximum of two targets at once, so this field should not exceed 02h.

BrTy Baud rate and the modulation type to be used during the initialization.

00h: 106 kbps type A (ISO/IEC 14443 Type A),

01h: 212kbps (FeliCa polling),

02h: 424kbps (FeliCa polling),

03h: 106kbps type B (ISO/IEC 14443-3B),

04h: 106kbps Innovision Jewel tag

InitiatorData[ ] An array of data to be used during the initialization of the target(s). Depending on the Baud Rate specified, the content of this field is different.

106 Kbps type A The field is optional and is present only when the host controller wants to initialize a target with a known UID.

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

In this case, *InitiatorData[ ]* contains the UID of the card (or part of it). The UID must include the cascade tag CT if it is cascaded level 2 or 3.

Cascade Level 1

UID1	UID2	UID3	UID4

Cascade Level 2

UID1	UID2	UID3	UID4	UID5	UID6	UID7

Cascade Level 3

UID1	UID2	UID3	UID4	UID5	UID6	UID7	UID8	UID9	UID10

106 Kbps type B In this case, InitiatorData[ ] is formatted as following:

AFI (1byte)	[Polling Method]

AFI The AFI (Application Family Identifier) parameter represents the type of application targeted by the device IC and is used to pre-select the PICCs before the ATQB.

This field is mandatory.

Polling Method This field is optional. It indicates the approach to be used in the ISO/IEC 14443-3B initialization:

- If bit 0 = 1: Probabilistic approach (option 1) in the ISO/IEC 14443-3B initialization,

- If bit 0 = 0: Timeslot approach (option 2) in the ISO/IEC 14443-3B initialization,

- If this field is absent, the timeslot approach will be used.

212/424 Kbps In this case, this field is mandatory and contains the complete pay load information that should be used in the polling request command (5bypes, length bytes is excluded).

106 Kbps InnoVision Jewel tag This field is not used.

Data Out Contactless Chip and Tag Response.

Contactless Chip and Tag Response returned by the reader.

Direct Transmit Response Format

Response	Data Out				
Result	D5 41	Status	[DataIn[ ]]		SW1 SW2
	D5 4B	NbTg	[TargetData1[ ]]	[TargetData2[ ]]	

Where:

Status 1 Byte; A byte indicating if the process has been terminated successfully or not. When in either DEP or ISO/IEC 14443-4 PCD mode, this byte also indicates if NAD (Node Address) is used and if the transfer of data is not completed with bit More Information.

DataIn 0-262 Bytes; An array of raw data received by the contactless chip.

NbTg The number of initialized Targets (minimum 0, maximum 2 targets).

TargetDatai[] The "i" in TargetDatai[] refers to "1" or "2." This contains the information about the detected targets and depends on the baud rate selected. The following information is given for one target, it is repeated for each target initialized (NbTg times).

106 Kbps Type A

Tg	SENS_RES10 (2 bytes)	SEL_RES (1 byte)	NFCIDLength (1 byte)	NFCID1[] (NFCIDLength bytes)	[ATS[]] (ATSLength bytes11)

106 Kbps Type B

Tg	ATQB Response (12 bytes)	ATTRIB_RES Length (1 byte)	ATTTRIB_RES[] (ATTRIB_RES Length)

212/424 Kbps

Tg	POL_RES length	01h (response code)	NFCID2t	Pad	SYST_CODE (optional)
1 byte	1 byte	1 byte	8 bytes	8 bytes	2 bytes
	POL_RES (18 or 20 bytes)				

106 Kbps Innovision Jewel tag

Tg	SENS_RES (2 bytes)	JEWELID[] (4 bytes)

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

Results	SW1 SW2	Meaning
Success	90 00h	The operation is completed successfully.
Error	63 00h	The operation has failed.
Time Out Error	63 01h	The TAG does not response.
Checksum Error	63 27h	The checksum of the Response is wrong.
Parameter Error	63 7Fh	The TAG Command is wrong.

Table 11: Direct Transmit Response Codes

4.1.2.2. Get Data

The Get Data command will return the serial number or ATS of the "connected PICC."

Get UID APDU Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Get Data	FFh	CAh	00h 01h	00h	00h (Full Length)

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

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

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

Response	Data Out		
Result	ATS	SW1	SW2

Get Data Response Code

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

Example 1:

To get the serial number of the connected PICC:

UINT8 GET_UID[5]={FFh, CAh, 00h, 00h, 00h}

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

UINT8 GET_ATS[5]={FFh, CAh, 01h, 00h, 00h};

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

4.1.2.3.1. Load Authentication Keys

The "Load Authentication Keys command" will load the authentication keys into the reader. The authentication keys are used to authenticate the particular sector of the MIFARE 1K/4K Memory Card. Two kinds of locations for authentication keys are provided, volatile and non-volatile.

Load Authentication Keys APDU Format (11 bytes)

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

Where:

Key Structure 1 Byte.

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

Other = Reserved.

Key Number 1 Byte.

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

Key 6 Bytes. The key value loaded into the reader.

E.g. {FF FF FF FF FF FFh}.

Load Authentication Keys Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Load Authentication Keys Response Codes

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

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

4.1.2.3.2. Authentication for MIFARE 1K/4K

The "Authentication command" uses the keys stored in the reader to do authentication with the MIFARE 1K/4K card (PICC). Two types of authentication keys used: TYPE_A and TYPE_B.

Load Authentication Keys APDU Format (6 bytes)

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

Load Authentication Keys APDU Format (10 bytes)

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

Authenticate Data Bytes (5 bytes)

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

Where:

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

Key Type 1 Byte.

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

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

Key Number 1 Byte.

00h ~ 01h = Key Location.

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

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

Load Authentication Keys Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Load Authentication Keys Response Codes

Results	SW1 SW2	Meaning
Success	90 00h	The operation is completed successfully.
Error	63 00h	The operation has 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	1K Bytes
Sector 1	04h ~ 06h	07h	
..			
..			
Sector 14	38h ~ 0Ah	3Bh	
Sector 15	3Ch ~ 3Eh	3Fh	

Table 12: MIFARE 1K Memory Map

Sectors (Total of 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	2K Bytes
Sector 1	04h ~ 06h	07h	
...			
...			
Sector 30	78h ~ 7Ah	7Bh	
Sector 31	7Ch ~ 7Eh	7Fh	

Sectors (Total of 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	2K Bytes
Sector 33	90h ~ 9Eh	9Fh	
...			
...			
Sector 38	E0h ~ EEh	EFh	
Sector 39	F0h ~ FEh	FFh	

Table 13: MIFARE 4K Memory Map

Example 1:

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

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

Example 2:

Similar to the previous example, if we authenticate Block 04h with the following characteristics: TYPE A, non-volatile, 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 14: MIFARE Ultralight Memory Map

4.1.2.3.3. Read Binary Blocks

The Read Binary Blocks command is used for retrieving multiple data blocks from the PICC. The data block/trailer block must be authenticated first.

Read Binary APDU Format (5 Bytes)

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

Where:

Block Number 1 Byte. The block to be accessed.

Number of Bytes to Read 1 Byte. Maximum 16 bytes.

Read Binary Block Response Format (N + 2 Bytes)

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

Read Binary Response Codes

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

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

APDU = {FF B0 00 04 10}

Example 2: Read 4 bytes from binary Page 04h (MIFARE Ultralight)

APDU = {FF B0 00 04 04}

Example 3: Read 16 bytes from binary Page 04h (MIFARE Ultralight) (Pages 4, 5, 6 and 7 will be read)

APDU = {FF B0 00 04 10}

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

4.1.2.3.4. Update Binary Blocks

The Update Binary Blocks command is used for writing multiple data blocks into the PICC. The data block/trailer block must be authenticated first.

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

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

Where:

Block Number	1 byte. This is the starting block to be updated.
Number of Bytes to Update	1 byte. 16 bytes for MIFARE 1K/4K 4 bytes for MIFARE Ultralight
Block Data	4 or 16 bytes. The data to be written in to binary block/blocks.

Update Binary Block Response Codes (2 Bytes)

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

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

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

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}

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

The Value Block Operation command is used for manipulating value-based transactions (e.g., increment a value of the value block, etc.).

Value Block Operation APDU Format (10 Bytes)

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

Where:

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

VB_OP 1 Byte.

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

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

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

VB_Value 4 Bytes. The value of this data, which is a signed long integer (4 bytes), is used for value manipulation.

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

VB_Value			
MSB			LSB
FFh	FFh	FF	FCh

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

VB_Value			
MSB			LSB
00h	00h	00h	01h

Value Block Operation Response Format (2 Bytes)

Response	Data Out	
Result	SW1	SW2

Value Block Operation Response Codes

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

4.1.2.3.6. Read Value Block

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

Read Value Block APDU Format (5 bytes)

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

Where:

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

Read Value Block Response Format (4 + 2 bytes)

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

Where:

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

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

Value			
MSB			LSB
FFh	FFh	FFh	FC

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

Value			
MSB			LSB
00h	00h	00h	01h

Read Value Block Response Codes

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

4.1.2.3.7. Copy Value Block

The Copy Value Block command is used to copy a value from a value block to another value block.

Copy Value Block APDU Format (7 Bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Copy Value Block Operation	FFh	D7h	00h	Source Block Number	02h	03h	Target Block Number

Where:

Source Block Number 1 Byte. The value of the source value block will be copied to the target value block.

Target Block Number 1 Byte. This is the value block to be restored. The source and target value blocks must be in the same sector.

Copy Value Block Response Format (2 Bytes)

Response	Data Out	
Result	SW1	SW2

Copy Value Block Response Codes

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

Example 1: Store a value "1" 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 "5"

APDU = {FF D7 00 05 05 01 00 00 00 05h}
Answer: 90 00h [$9000]

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

All ISO 14443-4 compliant cards (PICCs) would understand the ISO 7816-4 APDUs. The ACR89U-A2 Reader needs to communicate with the ISO 14443-4 compliant cards through exchanging ISO 7816-4 APDUs and Responses. ACR89U-A2 will handle the ISO 14443 Parts 1-4 Protocols internally.

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

ISO 7816-4 APDU Format

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

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

Response	Data Out		
Result	Response Data	SW1	SW2

Common ISO 7816-4 Response Codes

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

Typical sequence may be:

1. Present the Tag and Connect the PICC Interface.
2. 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 3Ah

In which,

The Application Data of ATQB = 00 00 00 00h, protocol information of ATQB = 33 81 81h. It is an ISO 14443-4 Type B tag.

Step 2: Send an APDU, Get Challenge.

<< 00 84 00 00 08h

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

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

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

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 [$9000]

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 "ACR89U PICC Interface" with T=1 protocol.

Step 2. Access the PICC by exchanging APDUs.

..

Step N. Disconnect the "ACR89U PICC Interface". Shut down the application.

Remarks:

The antenna can be switched off in order to save the power.

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

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

The MIFARE DESFire supports ISO 7816-4 APDU Wrapping and Native modes. Once the MIFARE DESFire Tag is activated, the first APDU sent to the MIFARE 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 (MIFARE DESFire)

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

Class = 90h; INS = 0Ah (MIFARE 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 [$91AF]

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

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

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

To get the version of the MIFARE DESFire card.

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

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

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 [$91AF]

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 [$9100]

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 (MIFARE DESFire)

APDU = {0A 00h}

Answer: AF 25 9C 65 0C 87 65 1D D7h [$1DD7]

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

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

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

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

To get the version of the MIFARE DESFire card.

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

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

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

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 [$2604]

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 C. Access FeliCa Tags (ISO 18092)

Typical sequence may be:

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

Step 1: Connect the Tag.

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

In which,

F0 11h = FeliCa 212K

Step 2: Read the memory block without using Pseudo APDU.

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

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

Or

Step 2: Read the memory block using Pseudo APDU.

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

In which,

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

D4 40 01h 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 AAh [90 00h]

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

***Note:** The NFC ID can be obtained by using the APDU "FF CA 00 00 00h." Please refer to the FeliCa specification for more detailed information.*

Appendix D. Access NFC Forum Type 1 Tags (ISO 18092)

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 9Fh

In which,

F0 04h = Topaz

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

<< 01 08h

>> 18h [90 00h]

In which, Response Data = 18h

Or

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

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

In which,

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

D4 40 01h is the Data Exchange Command.

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

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

In which, Response Data = 18h

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

<< 00h

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

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

<< 53 08 FFh

>> FFh [90 00h]

In which, Response Data = FFh

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

HR0	HR1
11h	xx h

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

Reserved for internal use
User Block Lock & Status
OTP bits

Figure 5: Topaz Memory Map

Memory Address = Block No * 8 + Byte No

Example 1: Memory Address 08h = 1 x 8 + 0 = Block 1: Byte-0 = Data0

Example 2: Memory Address 10h = 2 x 8 + 0 = Block 2: Byte-0 = Data8