Advanced Card Systems Ltd.
Card & Reader Technologies

ACR122S Serial NFC Reader

Communication Protocol

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

Table of Contents

1.0. Introduction 4

1.1. Serial Interface 4
1.2. Bi-Color LED 4
1.3. Buzzer 4
1.4. SAM Interface 4
1.5. Built-In Antenna 4

2.0. Communication between the Host and the Contactless Interface, SAM and Peripherals 5

3.0. Serial Interface (CCID-Liked FRAME Format) 6

3.1. Protocol Flow Examples 7

4.0. SAM Interface 9

4.1. To Activate the SAM Interface 9
4.2. To Deactivate the SAM Interface 10
4.3. To Do Data Exchange Through the SAM Interface 11

5.0. Pseudo-APDUs for Contactless Interface and Peripherals Control 13

5.1. Direct Transmit 13
5.2. Pseudo-APDU for Bi-Color LED and Buzzer Control 14
5.3. Pseudo-APDU for Changing the Communication Speed 21
5.4. Pseudo-APDU for Topaz512 and Jewel96 26
5.5. Get the Firmware Version of the Reader 34
5.6. Basic Program Flow for FeliCa Applications 34
5.7. Basic Program Flow for NFC Forum Type 1 Tags Applications 35
5.8. Basic Program Flow for MIFARE Applications 35
5.8.1. How to Handle Value Blocks of MIFARE 1K/4K Tag? 36
5.8.2. How to Access MIFARE Ultralight Tags? 39

Appendix A. Topaz 41

Appendix B. Topaz512 42

Appendix C. Jewel64 44

Appendix D. Jewel96 45

Figures

Figure 1: Communication Flowchart of ACR122S 5

Figure 2: Tag Address 32

Figure 3: Tag Address 33

Tables

Table 1: PIN Configuration 4

Table 2: ACR122S Command Frame Format 6

Table 3: ACR122S Status Frame Format 6

Table 4: ACR122S Response Frame Format 6

Table 5: ACR122S Command Frame Format 9

Table 6: HOST_to_RDR_IccPowerOn Format 9

Table 7: ACR122S Response Frame Format 9
Table 8: RDR_to_HOST_DataBlock Format 9
Table 9: ACR122S Command Frame Format 10
Table 10: HOST_to_RDR_IccPowerOff Format 10
Table 11: ACR122S Response Frame Format 10
Table 12: RDR_to_HOST_DataBlock Format 11
Table 13: ACR122S Command Frame Format 11
Table 14: HOST_to_RDR_XfrBlock Format 11
Table 15: ACR122S Response Frame Format 11
Table 16: RDR_to_HOST_DataBlock Format 12
Table 17: Direct Transmit Command Format (Length of the PN532_TAG Command + 5 Bytes) 13
Table 18: Direct Transmit Response Format (Response Length + 2 Bytes) 13
Table 19: Status Code 14
Table 20: Bi-Color LED and Buzzer Control Command Format (9 Bytes) 14
Table 21: Bi-Color LED and Buzzer Control Format (1 Byte) 14
Table 22: Bi-Color LED Blinking Duration Control Format (4 Bytes) 14
Table 23: Status Code 15
Table 24: Current LED State (1 Byte) 15
Table 25: Baud Rate Control Command Format (9 Bytes) 21
Table 26: Status Code 21
Table 27: Topaz512 and Jewel96 Command Format 26
Table 28: Direct Transmit Response Format (Response Length + 2 Bytes) 27
Table 29: Status Code 27
Table 30: Get Firmware Version Command Format (5 Bytes) 34
Table 31: Get Firmware Version Response Format (10 bytes) 34

1.0. Introduction

The ACR122S is a contactless smart card reader/writer used for accessing ISO 14443-4 Type A and B, Mifare, ISO 18092 or NFC, and FeliCa tags using the Serial Interface. This document will discuss the command set in implementing a smart card application using the ACR122S.

1.1. Serial Interface

The ACR122S is connected to a Host through the RS232C Serial Interface at 9600 bps, 8-N-1.

Pin	Signal	Function
1	VCC	+5V power supply for the reader (Max 200mA, Normal 100mA)
2	TXD	The signal from the reader to the host
3	RXD	The signal from the host to the reader
4	GND	Reference voltage level for power supply

Table 1: PIN Configuration

1.2. Bi-Color LED

A user-controllable Bi-Color LED with Red and Green Color is provided.

- The Green Color LED will be blinking if the “Card Interface” is not connected.
- The Green Color LED will be turned on if the “Card Interface” is connected.
- The Green Color LED will be flashing if the “Card Interface” is operating.
- The Red Color LED is controlled by the application only.

1.3. Buzzer

A user-controllable Buzzer with a default State of OFF is provided.

1.4. SAM Interface

One SAM socket is provided.

1.5. Built-In Antenna

- 3 turns symmetric loop antenna, Center-tapped.
- The estimated size = 60mm x 48mm.
- The loop inductance should be around ~ 1.6uH to 2.5uH
- Operating Distance for different Tags ~ up to 50mm (depends on the Tag)
- Only one Tag can be accessed at any one time.

2.0. Communication between the Host and the Contactless Interface, SAM and Peripherals

The Contactless interface and peripherals are accessed through the use of Pseudo-APDUs.

The SAM interface is accessed through the use of standard APDUs.

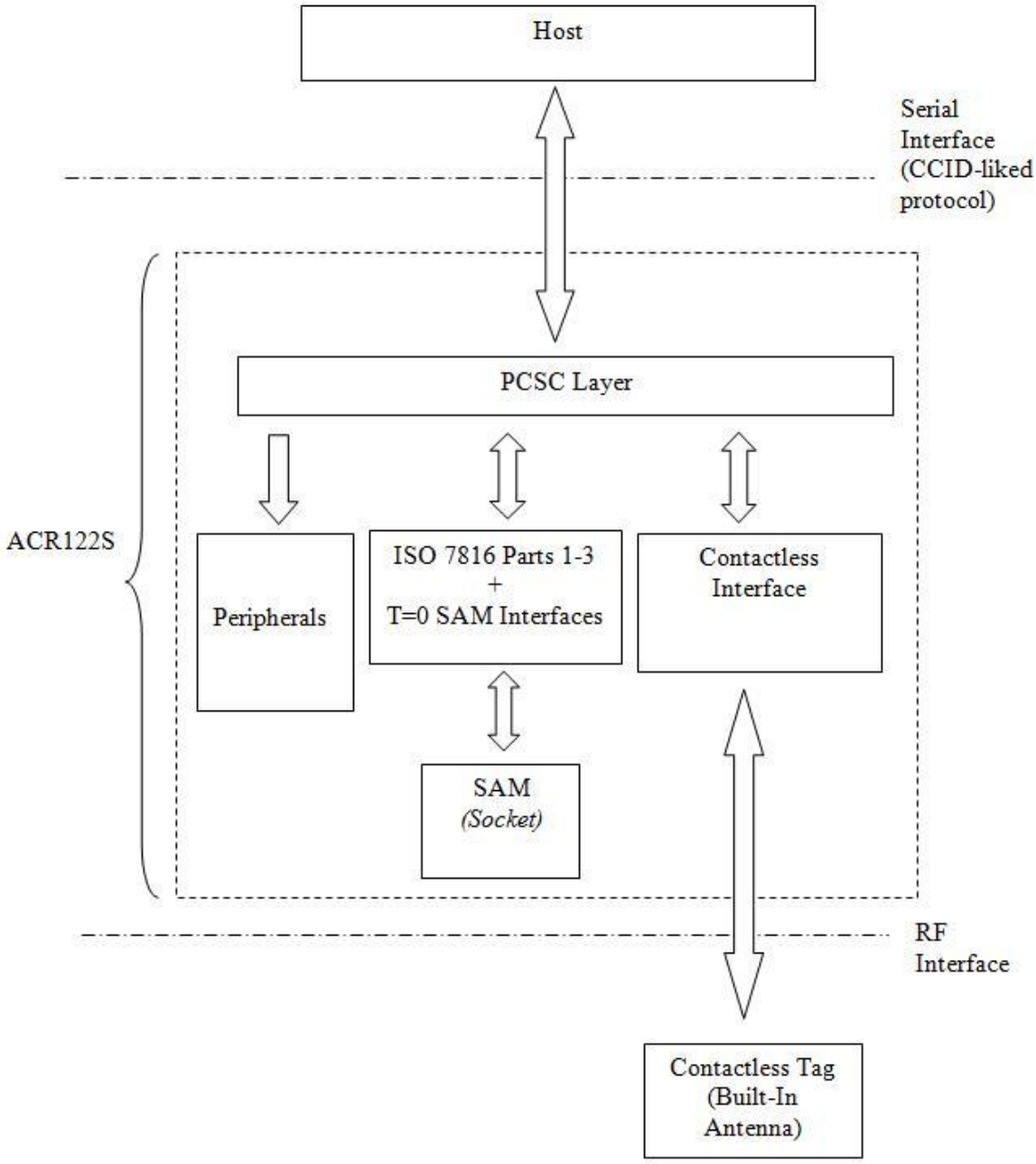


Figure 1: Communication Flowchart of ACR122S

3.0. Serial Interface (CCID-Liked FRAME Format)

Communication setting: 9600 bps, 8-N-1

The communication protocol between the Host and ACR122S is very similar to the CCID protocol.

STX (0x02)	Bulk-OUT Header	APDU Command Or Parameters	Checksum	ETX (0x03)
1 Byte	10 Bytes	M Bytes (If applicable)	1 Byte	1 Byte

Table 2: ACR122S Command Frame Format

STX (0x02)	Status	Checksum	ETX (0x03)
1 Byte	1 Byte	1 Byte	1 Byte

Table 3: ACR122S Status Frame Format

STX (0x02)	Bulk-IN Header	APDU Response Or abData	Checksum	ETX (0x03)
1 Byte	10 Bytes	N Bytes (If applicable)	1 Byte	1 Byte

Table 4: ACR122S Response Frame Format

Checksum = XOR {Bulk-OUT Header, APDU Command or Parameters}

Checksum = XOR {Bulk-IN Header, APDU Response or abData}

In general, we would make use of three types of Bulk-OUT Header:

- HOST_to_RDR_IccPowerOn: To activate the SAM interface. The ATR of the SAM will be returned if available.
- HOST_to_RDR_IccPowerOff: To deactivate the SAM interface.
- HOST_to_RDR_XfrBlock: To exchange APDUs between the Host and ACR122S.

The SAM interface must be activated in order to use the Contactless interface and Peripherals. In short, all the APDUs are exchanged through the SAM Interface.

Similarly, two types of Bulk-IN Header are used:

- RDR_to_HOST_DataBlock: In response to the “HOST_to_RDR_IccPowerOn” and “HOST_to_RDR_XfrBlock” Frames.
- RDR_to_HOST_SlotStatus: In response to the “HOST_to_RDR_IccPowerOff” Frame.

RDR = ACR122S; HOST = Host Controller

HOST_to_RDR = Host Controller -> ACR122S

RDR_to_HOST = ACR122S -> Host Controller

3.1. Protocol Flow Examples

1) Activate a SAM

	HOST		RDR
1. HOST sends a frame	→	02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03	
2. RDR sends back a positive status frame immediately		02 00 00 03 (positive status frame)	←
3. RDR sends back the response of the command		.. after some processing delay .. 02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03	←

2) Activate a SAM (Incorrect Checksum, HOST)

	HOST		RDR
1. HOST sends a corrupted frame	→	02 62 00 00 00 00 00 01 01 00 00 [Incorrect Checksum] 03	
2. RDR sends back a negative status frame immediately		02 FF FF 03 (negative status frame)	←
3. HOST sends the frame again.	→	02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03	
4. RDR sends back a positive status frame immediately		02 00 00 03 (positive status frame)	←
5. RDR sends back the response of the command		.. after some processing delay .. 02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03	←

3) Activate a SAM (Incorrect Checksum, RDR)

	HOST		RDR
1. HOST sends a frame	→	02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03	
2. RDR sends back a positive status frame immediately		02 00 00 03 (positive status frame)	←
		.. after some processing delay ..	
3. RDR sends back the response (corrupted) of the command		02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Incorrect Checksum] 03	←
4. HOST sends a NAK frame to get the response again.	→	02 00 00 00 00 00 00 00 00 00 00 00 03 (NAK)	
5. RDR sends back the response of the command		02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03	←

Note:

If the frame sent by the HOST is correctly received by the RDR, a positive status frame = {02 00 00 03} will be sent to the HOST immediately to inform the HOST the frame is correctly received. The HOST has to wait for the response of the command. The RDR will not receive any more frames while the command is being processed.

In case of errors, a negative status frame will be sent to the HOST to indicate the frame is either corrupted or incorrectly formatted.

- CheckSum Error Frame = {02 FF FF 03}
- Length Error Frame = {02 FE FE 03}. The length “dDwLength” is greater than 0x0105 bytes.
- ETX Error Frame = {02 FD FD 03}. The last byte is not equal to ETX “0x03”.

The NAK Frame is only used by the HOST to get the last response.

{02 00 00 00 00 00 00 00 00 00 00 00 03} // 11 zeros

4.0. SAM Interface

The ACR122S comes with a SAM interface.

4.1. To Activate the SAM Interface

STX (0x02)	Bulk-OUT Header (HOST_to_RDR_IccPowerOn)	Parameters	Checksum	ETX (0x03)
1 Byte	10 Bytes	0 Byte	1 Byte	1 Byte

Table 5: ACR122S Command Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	62h	
1	dDwLength <LSB .. MSB>	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command. Default=00h
6	bSeq	1	00-FFh	Sequence number for command
7	bPowerSelect	1	00h, 01h, 02h, or 03h	Voltage that is applied to the ICC 00h – Automatic Voltage Selection 01h – 5.0 volts 02h – 3.0 volts 03h – 1.8 volts
8	abRFU	2		Reserved for Future Use

Table 6: HOST_to_RDR_IccPowerOn Format

STX (0x02)	Bulk-IN Header (RDR_to_HOST_DataBlock)	abData	Checksum	ETX (0x03)
1 Byte	10 Bytes	N Bytes (ATR)	1 Byte	1 Byte

Table 7: ACR122S Response Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	80h	Indicates that a data block is being sent from the ACR122S
1	dwLength <LSB .. MSB>	4	N	Size of abData field. (N Bytes)
5	bSlot	1	Same as Bulk-OUT	Identifies the slot number for this command
6	bSeq	1	Same as Bulk-OUT	Sequence number for corresponding command
7	bStatus	1		
8	bError	1		
9	bChainParameter	1		

Table 8: RDR_to_HOST_DataBlock Format

Example. To activate the slot 0 (default), sequence number = 1, 5V card.

HOST-> 02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03
The ATR = 3B 2A 00 80 65 24 B0 00 02 00 82; SW1 SW2 = 90 00

4.2. To Deactivate the SAM Interface

STX (0x02)	Bulk-OUT Header (HOST_to_RDR_IccPowerOff)	Parameters	Checksum	ETX (0x03)
1 Byte	10 Bytes	0 Byte	1 Byte	1 Byte

Table 9: ACR122S Command Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	63h	
1	dDwLength <LSB .. MSB>	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command. Default=00h
6	bSeq	1	00-FFh	Sequence number for command
7	abRFU	3		Reserved for Future Use

Table 10: HOST_to_RDR_IccPowerOff Format

STX (0x02)	Bulk-IN Header (RDR_to_HOST_SlotStatus)	abData	Checksum	ETX (0x03)
1 Byte	10 Bytes	0 Byte	1 Byte	1 Byte

Table 11: ACR122S Response Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	81h	Indicates that a data block is being sent from the ACR122S
1	dwLength <LSB .. MSB>	4	0	Size of abData field. (0 Bytes)
5	bSlot	1	Same as Bulk-OUT	Identifies the slot number for this command
6	bSeq	1	Same as Bulk-OUT	Sequence number for corresponding command
7	bStatus	1		
8	bError	1		
9	bClockStatus	1		

Table 12: RDR_to_HOST_DataBlock Format

Example. To deactivate the slot 0 (default), sequence number = 2.

HOST -> 02 63 00 00 00 00 00 02 00 00 00 [Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 81 00 00 00 00 00 02 00 00 00 [Checksum] 03

4.3. To Do Data Exchange Through the SAM Interface

STX (0x02)	Bulk-OUT Header (HOST_to_RDR_XfrBlock)	Parameters	Checksum	ETX (0x03)
1 Byte	10 Bytes	M Byte	1 Byte	1 Byte

Table 13: ACR122S Command Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	6Fh	
1	dDwLength <LSB .. MSB>	4	M	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command. Default=00h
6	bSeq	1	00-FFh	Sequence number for command
7	bBWI	1	00-FFh	Used to extend the Block Waiting Timeout.
8	wLevelParameter	2	0000h	

Table 14: HOST_to_RDR_XfrBlock Format

STX (0x02)	Bulk-IN Header (RDR_to_HOST_DataBlock)	abData	Checksum	ETX (0x03)
1 Byte	10 Bytes	N Bytes (ATR)	1 Byte	1 Byte

Table 15: ACR122S Response Frame Format

Offset	Field	Size	Value	Description
0	bMessageType	1	80h	Indicates that a data block is being sent from the ACR122S
1	dwLength <LSB .. MSB>	4	N	Size of abData field. (N Bytes)
5	bSlot	1	Same as Bulk-OUT	Identifies the slot number for this command
6	bSeq	1	Same as Bulk-OUT	Sequence number for corresponding command
7	bStatus	1		
8	bError	1		
9	bChainParameter	1		

Table 16: RDR_to_HOST_DataBlock Format

Example. To send an APDU “80 84 00 00 08” to the slot 0 (default), sequence number = 3.

HOST -> 02 6F 05 00 00 00 00 03 00 00 00 80 84 00 00 08 [Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 80 0A 00 00 00 00 03 00 00 00 E3 51 B0 FC 88 AA 2D 18 90 00 [Checksum] 03

Response = E3 51 B0 FC 88 AA 2D 18; SW1 SW2 = 90 00

5.0. Pseudo-APDUs for Contactless Interface and Peripherals Control

ACR122S comes with two primitive commands for this purpose. <Class 0xFF>

5.1. Direct Transmit

To send a Pseudo-APDU (PN532 and TAG Commands), and the length of the Response Data will be returned.

Command	Class	INS	P1	P2	Lc	Data In
Direct Transmit	0xFF	0x00	0x00	0x00	Number of Bytes to send	PN532_TAG Command

Table 17: Direct Transmit Command Format (Length of the PN532_TAG Command + 5 Bytes)

Lc: Number of Bytes to Send (1 Byte)

Maximum 255 bytes

Data In: PN532_TAG Command

The data to be sent to the PN532 and Tag.

Response	Data Out	
Result	PN532_TAG Response	SW1 SW2

Table 18: Direct Transmit Response Format (Response Length + 2 Bytes)

Data Out: PN532_TAG Response

PN532_TAG Response returned by the reader.

Data Out: SW1 SW2

Status Code returned by the reader.

Results	SW1	SW2	Meaning
Success	61	LEN	The operation is completed successfully. The response data has a length of LEN bytes. The APDU "Get Response" should be used to retrieve the response data.
Error	63	00	The operation is failed.
Time Out Error	63	01	The PN532 does not response.
Checksum Error	63	27	The checksum of the Response is wrong.
Parameter Error	63	7F	The PN532_TAG Command is wrong.

Table 19: Status Code

5.2. Pseudo-APDU for Bi-Color LED and Buzzer Control

This APDU is used to control the states of the Bi-Color LED and Buzzer.

Command	Class	INS	P1	P2	Lc	Data In (4 Bytes)
Bi-Color LED and Buzzer Control	0xFF	0x00	0x40	LED State Control	0x04	Blinking Duration Control

Table 20: Bi-Color LED and Buzzer Control Command Format (9 Bytes)

P2: LED State Control

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

Table 21: Bi-Color LED and Buzzer Control Format (1 Byte)

Data In: Blinking Duration Control

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

Table 22: Bi-Color LED Blinking Duration Control Format (4 Bytes)

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

0x00: The buzzer will not turn on

0x01: The buzzer will turn on during the T1 Duration

0x02: The buzzer will turn on during the T2 Duration

0x03: The buzzer will turn on during the T1 and T2 Duration

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

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

Table 23: Status Code

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

Table 24: Current LED State (1 Byte)

Note:

1. The LED State operation will be performed after the LED Blinking operation is completed.
2. The LED will not be changed if the corresponding LED Mask is not enabled.
3. The LED will not be blinking if the corresponding LED Blinking Mask is not enabled. Also, the number of repetition must be greater than zero.
4. T1 and T2 duration parameters are used for controlling the duty cycle of LED blinking and Buzzer Turn-On duration. For example, if T1=1 and T2=1, the duty cycle = 50%. Duty Cycle = T1 / (T1 + T2).
5. To control the buzzer only, just set the P2 “LED State Control” to zero.
6. The make the buzzer operating, the “number of repetition” must be greater than zero.
7. To control the LED only, just set the parameter “Link to Buzzer” to zero.

Example 1: To read the existing LED State

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

// Not link to the buzzer //

APDU = “FF 00 40 00 04 00 00 00 00”

Response = “90 00”. RED and Green LEDs are OFF.

Example 2: To turn on RED and Green Color LEDs

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

// Not link to the buzzer //

APDU = “FF 00 40 0F 04 00 00 00 00”

Response = “90 03”. RED and Green LEDs are ON,

To turn off both RED and Green LEDs, APDU = “FF 00 40 0C 04 00 00 00 00”

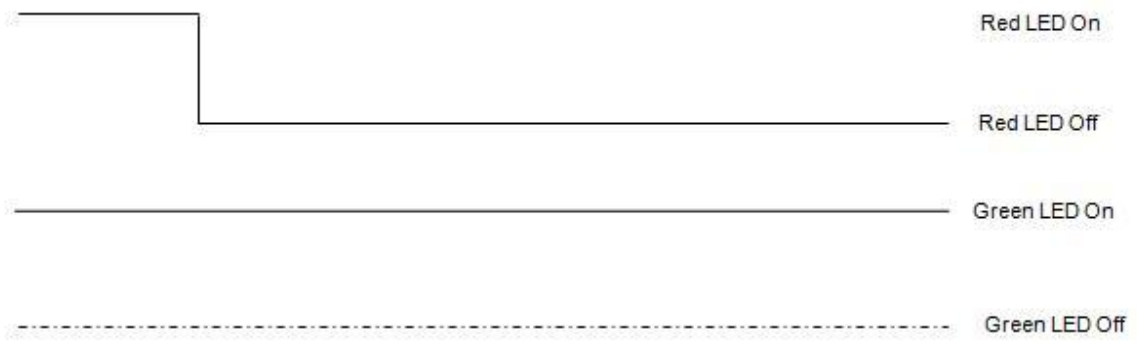
Example 3: To turn off the RED Color LED only, and leave the Green Color LED unchanged.

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

// Not link to the buzzer //

APDU = “FF 00 40 04 04 00 00 00 00”

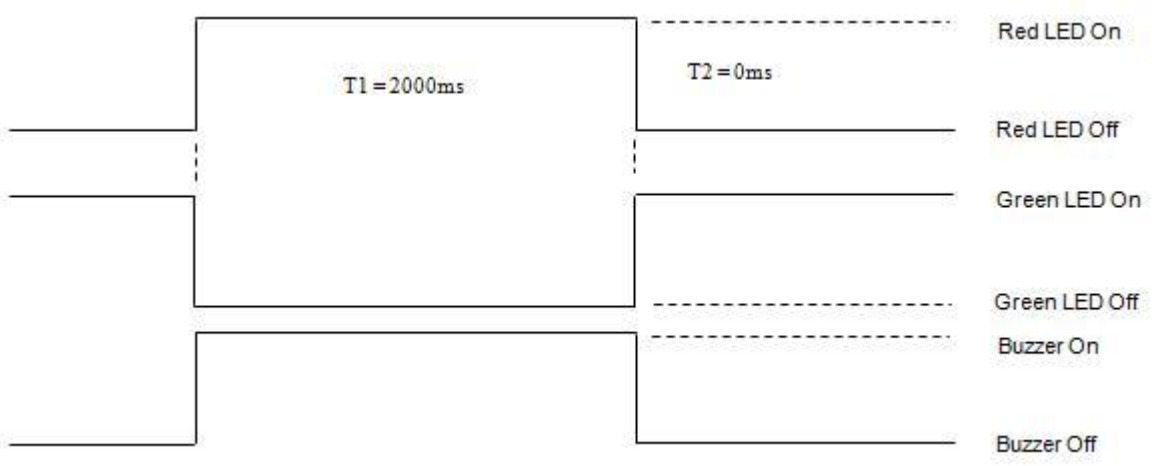
Response = “90 02”. Green LED is not changed (ON); Red LED is OFF,



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

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

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



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

T1 Duration = 2000ms = 0x14

T2 Duration = 0ms = 0x00

Number of repetition = 0x01

Link to Buzzer = 0x01

APDU = “FF 00 40 50 04 14 00 01 01”

Response = “90 02”

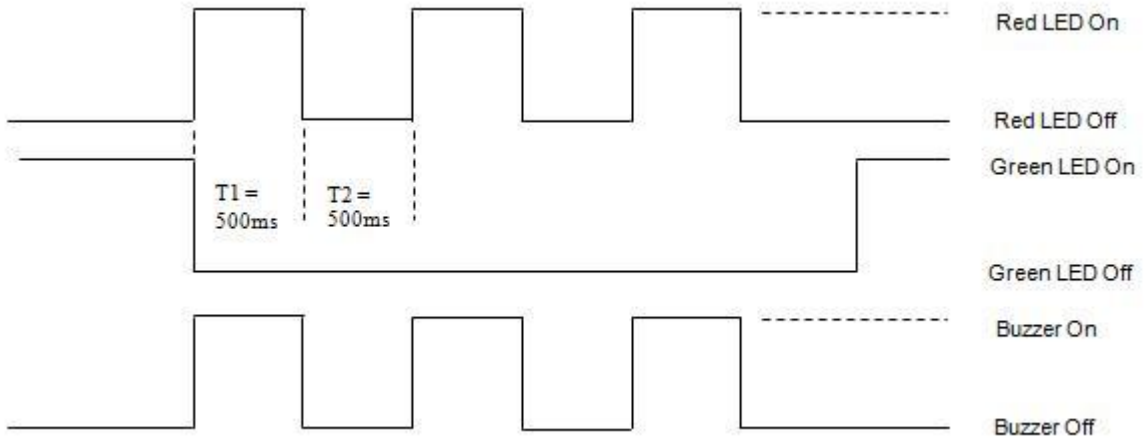
Example 5: To blink the Red LED of 1Hz for 3 times. After that, resume to initial state

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

// The Initial Red LED Blinking State is ON. Only the Red LED will be blinking.

// The buzzer will turn on during the T1 duration, while the Green LED will turn off during both the T1 and T2 duration.

// After the blinking, the Green LED will turn ON. The Red LED will resume to the initial state after the blinking //



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

T1 Duration = 500ms = 0x05

T2 Duration = 500ms = 0x05

Number of repetition = 0x03

Link to Buzzer = 0x01

APDU = “FF 00 40 50 04 05 05 03 01”

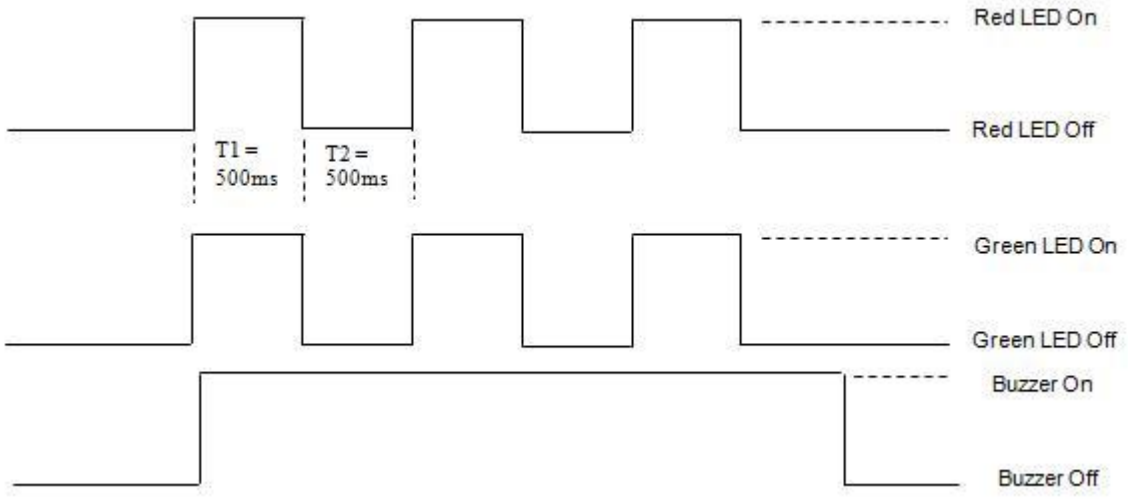
Response = “90 02”

Example 6: To blink the Red and Green LEDs of 1Hz for 3 times

// Assume both the Red and Green LEDs are initially OFF. //

// Both Initial Red and Green Blinking States are ON //

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



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

T1 Duration = 500ms = 0x05

T2 Duration = 500ms = 0x05

Number of repetition = 0x03

Link to Buzzer = 0x03

APDU = “FF 00 40 F0 04 05 05 03 03”

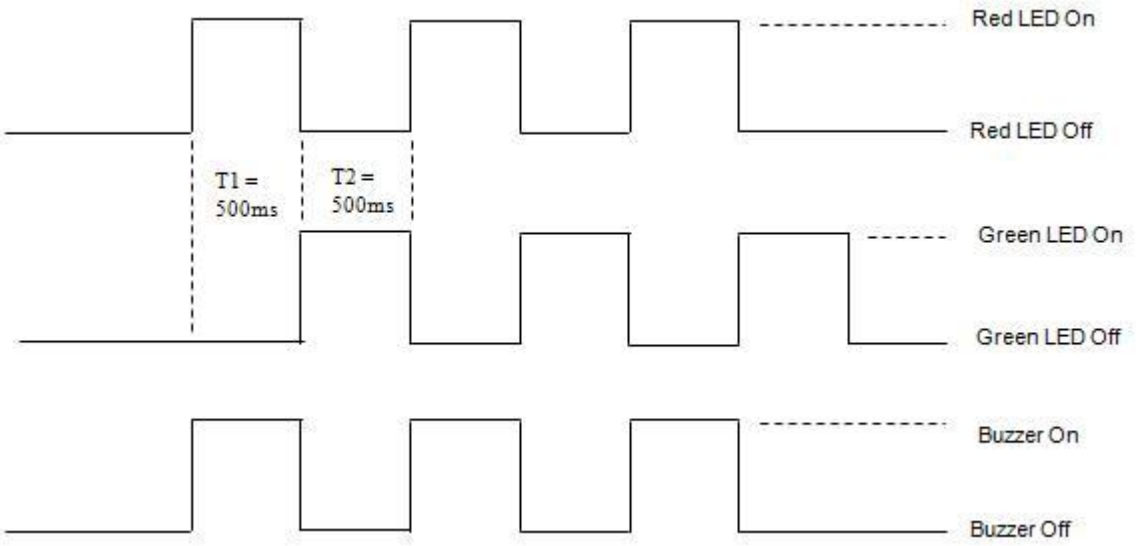
Response = “90 00”

Example 7: To blink the Red and Green LED in turn of 1Hz for 3 times

// Assume both Red and Green LEDs are initially OFF. //

// The Initial Red Blinking State is ON; The Initial Green Blinking States is OFF //

// The buzzer will turn on during the T1 duration//



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

T1 Duration = 500ms = 0x05

T2 Duration = 500ms = 0x05

Number of repetition = 0x03

Link to Buzzer = 0x01

APDU = “FF 00 40 D0 04 05 05 03 01”

Response = “90 00”

5.3. Pseudo-APDU for Changing the Communication Speed

This APDU is used to change the baud rate.

Command	Class	INS	P1	P2	Lc
Baud Rate Control	0xFF	0x00	0x44	New Baud Rate	0x00

Table 25: Baud Rate Control Command Format (9 Bytes)

P2: New Baud Rate

0x00: Set the new baud rate to 9600 bps.

0x01: Set the new baud rate to 115200 bps.

Data Out: SW1 SW2.

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

Table 26: Status Code

SW2: Current Baud Rate

0x00: The current baud rate is 9600 bps.

0x01: The current baud rate is 115200 bps.

Note:

After the communication speed is changed successfully, the program has to adjust its communication speed so as to continue the rest of the data exchanges.

The initial communication speed is determined by the existence of R12 (0 ohm).

- With R12 = 115200 bps
- Without R12 = 9600 bps (default)

Example 1: To initialize a FeliCa Tag (Tag Polling)

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 09 D4 4A 01 01 00 FF FF 01 00"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "09"
PN532 Command (InListPassiveTarget 212Kbps) = "D4 4A 01 01"
Tag Command (System Code Request) = "00 FF FF 01 00"

To send an APDU to the slot 0 (default), sequence number = 1.

HOST-> 02 6F 0E 00 00 00 00 01 00 00 00
FF 00 00 00 09 D4 4A 01 01 00 FF FF 01 00
[Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 81 1A 00 00 00 00 01 00 00 00
D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00
4B 02 4F 49 8A 8A 80 08 90 00
[Checksum] 03

The APDU Response is
"D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08 90 00"

In which,
Response returned by the PN532 =
"D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08"
NFCID2t of the FeliCa Tag = "01 01 05 01 86 04 02 02"
Status Code returned by the reader = "90 00"

Example 2: To write 16 bytes data to the FeliCa Tag (Tag Write)

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 23 D4 40 01 20 08 01 01 05 01 86 04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "23"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Write Data) = "20 08 01 01 05 01 86 04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA".

To send an APDU to the slot 0 (default), sequence number = 2.

HOST -> 02 6F 28 00 00 00 00 02 00 00 00
FF 00 00 00 00 23 D4 40 01 20 08 01 01 05 01 86
04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55
AA 55 AA 55 AA 55 AA
[Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 81 11 00 00 00 00 02 00 00 00
D5 41 00 0C 09 01 01 05 01 86 04 02 02 00 00 90 00
[Checksum] 03

The APDU Response would be
"D5 41 00 0C 09 01 01 05 01 86 04 02 02 00 00 90 00"

In which,
Response returned by the PN532 = "D5 41"
Response returned by the FeliCa Tag = "00 0C 09 01 01 05 01 86 04 02 02 00 00"
Status Code returned by the reader = "90 00"

Example 3: To read 16 bytes data from the FeliCa Tag (Tag Write)

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 13 D4 40 01 10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "13"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Read Data) = "10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00"

To send an APDU to the slot 0 (default), sequence number = 3.

HOST -> 02 6F 18 00 00 00 00 03 00 00 00
FF 00 00 00 13 D4 40 01 10 06 01 01 05 01 86 04
02 02 01 09 01 01 80 00 FF
[Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 81 22 00 00 00 00 03 00 00 00
D5 41 00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00
[Checksum] 03

The APDU Response would be
"D5 41 00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00"

In which,
Response returned by the PN532 = "D5 41"
Response returned by the FeliCa Tag =
"00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA"
Status Code returned by the reader = "90 00"

Example 4: To initialize an ISO 14443-4 Type B Tag (Tag Polling)

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 05 D4 4A 01 03 00"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "05"
PN532 Command (InListPassiveTarget Type B 106Kbps) = "D4 4A 01 03 00"

To send an APDU to the slot 0 (default), sequence number = 4.

HOST -> 02 6F 0A 00 00 00 00 04 00 00 00
FF 00 00 00 05 D4 4A 01 03 00
[Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 81 14 00 00 00 00 04 00 00 00
D5 41 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21
90 00 [Checksum] 03

The APDU Response is
"D5 4B 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21 90 00"

In which,
Response returned by the PN532 =
"D5 4B 01 01"
ATQB of the Type B Tag = "50 00 01 32 F4 00 00 00 00 33 81 81"
CRC-B = "01 21"
Status Code returned by the reader = "90 00"

Example 5: To send an APDU to an ISO 14443-4 Type B Tag (Data Exchange)

Step 1: Issue a "Direct Transmit" APDU.

The USER APDU Command should be "00 84 00 00 08"
The Composed APDU Command should be "FF 00 00 00 08 D4 40 01 00 84 00 00 08"

In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "08"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Get Challenge) = "00 84 00 00 08"

To send an APDU to the slot 0 (default), sequence number = 5.

HOST -> 02 6F 0D 00 00 00 00 05 00 00 00
FF 00 00 00 08 D4 40 01 00 84 00 00 08
[Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 81 0F 00 00 00 00 05 00 00 00
D5 41 00 01 02 03 04 05 06 07 08 90 00 90 00
[Checksum] 03

The APDU Response is
"D5 41 00 0B 01 02 03 04 05 06 07 08 90 00"

In which,
Response returned by the PN532 =
"D5 41 00"
Response from the Type B Tag = "01 02 03 04 05 06 07 08 90 00"
Status Code returned by the reader = "90 00"

5.4. Pseudo-APDU for Topaz512 and Jewel96

This APDU is used to Write-with-erase (8 Bytes), Write-no-erase (8 Bytes), Read (8 Bytes) and Read Segment.

Command	Class	INS	P1	P2	Lc	Data In
Direct Transmit	0xFF	0x00	0x00	0x00	Number of Bytes to send	PN532_TAG Command

Table 27: Topaz512 and Jewel96 Command Format

Lc: Number of Bytes to Send (1 Byte)

Maximum 255 bytes

Data In: PN532_TAG Command

The data to be sent to the PN532 and Tag.

Response	Data Out	
Result	PN532_TAG Response	SW1 SW2

Table 28: Direct Transmit Response Format (Response Length + 2 Bytes)

Data Out: PN532_TAG Response

PN532_TAG Response returned by the reader.

Data Out: SW1 SW2

Status Code returned by the reader.

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

Table 29: Status Code

Example 1: To Write-with-erase (8 Bytes) a Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 54 05 01 23 45 67 89 AB CD EF"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "0D"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Write-with-erase 8Bytes) = "54 "
Tag Address (00~3F (hex)) = "05 "
Tag Data = "01 23 45 67 89 AB CD EF "

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 12 00 00 00 00 01 00 00 00
          FF 00 00 00 0D D4 40 01 54 05 01 23 45 67 89 AB CD EF
          [Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 80 0D 00 00 00 00 01 01 00 00
          D5 09 05 01 23 45 67 89 AB CD EF 90 00
          [Checksum] 03
```

The APDU Response is
"D5 09 05 01 23 45 67 89 AB CD EF 90 00"

#In which,
Response returned by the PN532 =
"D5 09 05 01 23 45 67 89 AB CD EF 90 00"
Write Tag Address = "05 "
Write Tag 8Bytes Data = "01 23 45 67 89 AB CD EF "
Status Code returned by the reader = "90 00"

Example 2: To Write-no-erase (8 Bytes) a Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 1B 05 FF FF FF FF FF FF FF FF"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "0D"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Write-no-erase 8Bytes) = "1B "
Tag Address (00~3F (hex)) = "05 "
Tag Data = "FF FF FF FF FF FF FF FF "

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 12 00 00 00 00 01 00 00 00
          FF 00 00 00 0D D4 40 01 1B 05 FF FF FF FF FF FF FF FF
          [Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 80 0D 00 00 00 00 01 01 00 00
          D5 09 05 FF FF FF FF FF FF FF FF 90 00
          [Checksum] 03
```

The APDU Response is
"D5 09 05 FF FF FF FF FF FF FF FF 90 00"

#In which,
Response returned by the PN532 =
"D5 09 05 FF FF FF FF FF FF FF FF 90 00"
Write Tag Address = "05 "
Write Tag 8Bytes Data = "FF FF FF FF FF FF FF FF "
Status Code returned by the reader = "90 00"

Example 3: To Read 8 Bytes a Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 02 05 00 00 00 00 00 00 00 00"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "0D"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Read 8Bytes) = "02 "
Tag Address (00~3F (hex)) = "05 "
Tag Data = "00 00 00 00 00 00 00 00"

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 12 00 00 00 00 01 00 00 00
          FF 00 00 00 0D D4 40 01 02 05 00 00 00 00 00 00 00 00
          [Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 80 0D 00 00 00 00 01 01 00 00
          D5 09 05 01 23 45 67 89 AB CD EF 90 00
          [Checksum] 03
```

The APDU Response is
"D5 09 05 01 23 45 67 89 AB CD EF 90 00"

#In which,
Response returned by the PN532 =
"D5 09 05 01 23 45 67 89 AB CD EF 90 00"
Read Tag Address = "05 "
Read Tag 8Bytes Data = "01 23 45 67 89 AB CD EF "
Status Code returned by the reader = "90 00"

Example 4: To Read Segment a Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 10 00 00 00 00 00 00 00 00 00"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "0D"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Read Segment) = "10 "
Tag Address (00/10/20/30) = "00 "(Block 0)
Tag Data = "00 00 00 00 00 00 00 00"

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 12 00 00 00 00 01 00 00 00
          FF 00 00 00 0D D4 40 01 10 00 00 00 00 00 00 00 00 00
          [Checksum] 03

RDR -> 02 00 00 03

RDR  -> 02 80 0D 00 00 00 00 01 01 00 00
          D5 41 00 ... <128 bytes data> ... 90 00
          [Checksum] 03
```

The APDU Response is
"D5 41 00 … <128 bytes data> … 90 00"

#In which,
Response returned by the PN532 =
"D5 41 00 … <128 bytes data> … 90 00"
Read Tag Segment Data = "<128 bytes data> "
Status Code returned by the reader = "90 00"

Example 5: To Write Multi-Data at Topaz/Jewel Tag

Note that this function only can write at the segment 0.

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 36 D4 40 01 58 20 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "36"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Write Multi-Data) = "58 "
Tag Address = "20 (0 0100 000) "(Block 4, Byte-0) (refer to below Fig. 2)
*Tag Data = "*00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47*"*

Address operand 'ADD' Block = select one of blocks 0_h – F_h Byte = select one of Bytes 0 – 7							
msb							lsb
b8	b7	b6	b5	b4	b3	b2	b1
0_b	Block				Byte		

Figure 2: Tag Address

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 3B 00 00 00 00 01 00 00 00
          FF 00 00 00 36 D4 40 01 58 20 30 00 01 02 03 04 05 06 07 08 09 10
          11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
          33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
          [Checksum] 03

RDR  -> 02 00 00 03

RDR  -> 02 80 05 00 00 00 00 01 01 00 00
          D5 41 00 90 00
          [Checksum] 03
```

The APDU Response is
"D5 41 00 90 00"

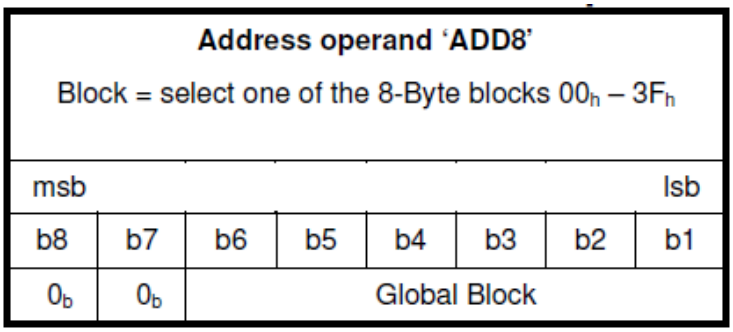
#In which,
Response returned by the PN532 = "D5 41 00 90 00"
*Write Tag Data = "*00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 *"*
Status Code returned by the reader = "90 00"
If Status Code returned by the reader = "63 00" that mean this operation is not complete.

Example 6: To Write Multi-8 bytes Data at Topaz512/Jewel96 Tag

Step 1: Issue a "Direct Transmit" APDU.

The APDU Command should be "FF 00 00 00 36 D4 40 01 5A 04 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47"

#In which,
Direct Transmit APDU = "FF 00 00 00"
Length of the PN532_Tag Command = "36"
PN532 Command (InDataExchange) = "D4 40 01"
Tag Command (Write Multi-Data) = "5A"
Tag Address (Block No.00-3F) = "04 "(Block No. 4) (refer to below Fig. 3)
*Tag Data = "*00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47*"*



Address operand 'ADD8'							
Block = select one of the 8-Byte blocks 00_h – $3F_h$							
msb							lsb
b8	b7	b6	b5	b4	b3	b2	b1
0_b	0_b	Global Block					

Figure 3: Tag Address

To send an APDU to the slot 0 (default), sequence number = 1.

```
HOST -> 02 6F 3B 00 00 00 00 01 00 00 00
          FF 00 00 00 36 D4 40 01 5A 04 30 00 01 02 03 04 05 06 07 08 09 10
          11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
          33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
          [Checksum] 03

RDR  -> 02 00 00 03

RDR  -> 02 80 04 00 00 00 00 01 01 00 00
          D5 09 90 00
          [Checksum] 03
```

The APDU Response is
"D5 09 90 00"

#In which,
Response returned by the PN532 = "D5 09 90 00"
*Write Tag Data = "*00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 *"*
Status Code returned by the reader = "90 00"
If Status Code returned by the reader = "63 00" that mean this operation is not complete.

5.5. Get the Firmware Version of the Reader

To derive the firmware version of the reader.

Command	Class	INS	P1	P2	Le
Get Response	0xFF	0x00	0x48	0x00	0x00

Table 30: Get Firmware Version Command Format (5 Bytes)

Le: Number of Bytes to Retrieve (1 Byte)

Maximum 255 bytes

Response	Data Out
Result	Firmware Version

Table 31: Get Firmware Version Response Format (10 bytes)

E.g. Response = 41 43 52 31 32 32 53 31 30 30 (Hex) = ACR122S100 (ASCII)

5.6. Basic Program Flow for FeliCa Applications

Step 0. Start the application. The first thing to do is to activate the “SAM Interface”. The ATR of the SAM (if a SAM is inserted) or a Pseudo-ATR “3B 00” (if no SAM is inserted) will be returned. In other words, the SAM always exists from the view of the application.

Step 1. The second thing to do is to change the operating parameters of the PN531. Set the Retry Time to one.

Step 2. Poll a FeliCa Tag by sending “Direct Transmit” and “Get Response” APDUs (Tag Polling).

Step 3. If no tag is found, go back to Step 2 until a FeliCa Tag is found.

Step 4. Access the FeliCa Tag by sending APDUs (Tag Read or Write).

Step 5. If there is no any operation with the FeliCa Tag, then go back to Step 2 to poll the other FeliCa Tag.

..

Step N. Deactivate the “SAM Interface”. Shut down the application.

Note:

1. The default Retry Time of the PN532 command “InListPassiveTarget” is infinity. Send the APDU “FF 00 00 00 06 D4 32 05 00 00 00” to change the Retry Time to one.
2. It is recommended to turn off the Antenna if there is no contactless access. APDU for turning on the Antenna Power = APDU “FF 00 00 00 04 D4 32 01 03”

   APDU for turning off the Antenna Power = APDU “FF 00 00 00 04 D4 32 01 02”

5.7. Basic Program Flow for NFC Forum Type 1 Tags Applications

E.g. Jewel and Topaz Tags

Typical sequence may be:
- Scanning the tags in the field (Polling)
- Read / Update the memory of the tag
- Deselect the tag

Step 1) **Polling** for the Jewel or Topaz Tag, 106 kbps
HOST-> 02 6F 09 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST-> FF 00 00 00 04 D4 4A 01 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0C 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 0C 00 B5 3E 21 00 90 00 [Checksum] 03
In which, Number of Tag found = [01]; Target number = 01
ATQA_RES = 0C 00; UID = B5 3E 21 00
Operation Finished = 90 00

Step 2) Read the memory address 08 (Block 1: Byte-0)
HOST-> 02 6F 0A 00 00 00 00 01 00 00 00 FF 00 00 00 05 D4 40 01 01 08 [Checksum] 03
RDR -> 02 00 00 03 02 80 06 00 00 00 00 01 01 00 00 D5 41 [00] 18 90 00 [Checksum] 03
In which, Response Data = 18

Tip: To read all the memory content of the tag starting from the memory address 00
HOST-> 02 6F 09 00 00 00 00 01 00 00 00 FF 00 00 00 04 D4 40 01 00 [Checksum] 03
RDR -> 02 00 00 03 02 80 7F 00 00 00 00 01 01 00 00 D5 41 00 11 48
RDR -> show all data … 90 00 [Checksum] 03

Step 3) Update the memory address 08(Block 1: Byte-0)with the data FF
HOST-> 2 6F 0B 00 00 00 00 01 00 00 00 FF 00 00 00 06 D4 40 01 53 08 FF [Checksum] 03
RDR -> 02 00 00 03 02 80 05 00 00 00 00 01 01 00 00 D5 41 [00] FF 90 00 [Checksum] 03
In which, Response Data = FF

Tip: To update more than one memory content of the tag starting form the memory address 08(Block 1: Byte-0)
HOST -> 02 6F 0D 00 00 00 00 01 00 00 00 FF 00 00 00 08 D4 40 01 58 08 02 AA BB [Checksum] 03
RDR -> 02 00 00 03 02 80 06 00 00 00 00 01 01 00 00 D5 41 [00] 90 00 [Checksum] 03
In which, Command = 58; Starting memory address = 08;
Number of write content = 02; Memory content = AA, BB;

Step 4) Deselect the Tag
HOST-> 02 6F 08 00 00 00 00 01 00 00 00 FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 02 80 05 00 00 00 00 01 01 00 00 D5 45 [00] 90 00 [Checksum] 03

5.8. Basic Program Flow for MIFARE Applications

Typical sequence may be:

- Scanning the tags in the field (Polling)
- Authentication
- Read / Write the memory of the tag
- Halt the tag (optional)

Step 1) **Polling** for the MIFARE 1K/4K Tags, 106 kbps
HOST-> 02 6F 09 00 00 00 00 01 00 00 00 FF 00 00 00 04 D4 4A 01 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0E 00 00 00 00 01 01 00 00

RDR -> D5 4B 01 01 00 02 18 04 F6 8E 2A 99 90 00 [Checksum] 03
In which, Number of Tag found = [01]; Target number = 01
SENS_RES = 00 02; SEL_RES = 18,
Length of the UID = 4; UID = F6 8E 2A 99
Operation Finished = 90 00

Tip: The tag type can be determined by recognizing the SEL_RES.
SEL_RES of some common tag types.

00 = MIFARE Ultralight
08 = MIFARE 1K
09 = MIFARE MINI
18 = MIFARE 4K
20 = MIFARE DESFIRE
28 = JCOP30
98 = Gemplus MPCOS

Step 2) **KEY A Authentication,** Block **04**, KEY = FF FF FF FF FF FF, UID = F6 8E 2A 99
HOST -> 02 6F 14 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 0F D4 40 01 **60** **04** FF FF FF FF FF FF F6 8E 2A 99 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Tip: If the authentication failed, the error code [XX] will be returned.
[00] = Valid, other = Error. Please refer to Error Codes Table for more details.

*Tip: For **KEY B Authentication***
HOST -> 02 6F 14 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 0F D4 40 01 **61** **04** FF FF FF FF FF FF F6 8E 2A 99
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Step 3) **Read** the content of Block **04**
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 FF 00 00 00 05 D4 40 01 **30** **04** [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [Checksum] 03
In which, Block Data = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Step 4) **Update** the content of Block **04**
HOST -> 02 6F 14 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 15 D4 40 01 **A0** **04** 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Step 5) **Halt the tag** (optional)
HOST -> 02 6F 08 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 45 [00] 90 00 [Checksum] 03

5.8.1. How to Handle Value Blocks of MIFARE 1K/4K Tag?

The value blocks are used for performing electronic purse functions, e.g. Increment, Decrement, Restore, Transfer, etc. The value blocks have a fixed data format which permits error detection and

correction and a backup management.

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Value				Value				Value				Adr	Adr	Adr	Adr

Value: A signed 4-Byte value. The lowest significant byte off a value is stored in the lowest address byte. Negative values are stored in standard 2's complement format.

Adr: 1-Byte address, which can be used to save the storage address of a block. (optional)

E.g. Value 100 (decimal) = 64 (Hex), assume Block = 0x05
The formatted value block = 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA

Step 1) **Update** the content of Block **05 with a value 100 (dec)**
HOST-> 02 6F 1A 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 15 D4 40 01 **A0** **05** 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Step 2) **Increment** the value of Block **05 by 1 (dec)**
HOST-> 02 6F 0E 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 09 D4 40 01 **C1** **05** 01 00 00 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

*Tip: **Decrement** the value of Block **05 by 1 (dec)***
HOST-> FF 00 00 00 09 D4 40 01 **C0** **05** 01 00 00 00

Step 3) **Transfer** the prior calculated value of Block **05 (dec)**
HOST-> 02 6F 0A 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 05 D4 40 01 **B0** **05** [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

*Tip: **Restore** the value of Block **05** (cancel the prior Increment or Decrement operation)*
HOST-> FF 00 00 00 05 D4 40 01 **C2** **05**

Step 4) **Read** the content of Block **05**
HOST-> 02 6F 0A 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 05 D4 40 01 **30** **05** [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 65 00 00 00 9A FF FF FF 65 00 00 00 05 FA 05 FA 90 00 [Checksum] 03
In which, the value = 101 (dec)

Step 5) **Copy** the value of Block **05** to Block **06 (dec)**
HOST-> 02 6F 0A 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 05 D4 40 01 **C2** **05** [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

HOST-> 02 6F 0A 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 05 D4 40 01 **B0** **06** [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Step 6) **Read** the content of Block **06**
HOST-> 02 6F 0A 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 05 D4 40 01 **30** **06** [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 65 00 00 00 9A FF FF FF 65 00 00 00 05 FA 05 FA 90 00 [Checksum] 03
In which, the value = 101 (dec). The Adr "05 FA 05 FA" tells us the value is copied from Block 05.

Please refer to the MIFARE specification for more detailed information.

MIFARE 1K Memory Map.

Sectors (Total 16 sectors. Each sector consists of 4 consecutive blocks)	**Data Blocks** (3 blocks, 16 bytes per block)	**Trailer Block** (1 block, 16 bytes)
Sector 0	0x00 ~ 0x02	0x03
Sector 1	0x04 ~ 0x06	0x07
..		
..		
Sector 14	0x38 ~ 0x0A	0x3B
Sector 15	0x3C ~ 0x3E	0x3F

1K Bytes

MIFARE 4K Memory Map.

Sectors (Total 32 sectors. Each sector consists of 4 consecutive blocks)	**Data Blocks** (3 blocks, 16 bytes per block)	**Trailer Block** (1 block, 16 bytes)
Sector 0	0x00 ~ 0x02	0x03
Sector 1	0x04 ~ 0x06	0x07
..		
..		
Sector 30	0x78 ~ 0x7A	0x7B
Sector 31	0x7C ~ 0x7E	0x7F

2K Bytes

Sectors (Total 8 sectors. Each sector consists of 16 consecutive blocks)	Data Blocks (15 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)
Sector 32	0x80 ~ 0x8E	0x8F
Sector 33	0x90 ~ 0x9E	0x9F
..		
..		
Sector 38	0xE0 ~ 0xEE	0xEF
Sector 39	0xF0 ~ 0xFE	0xFF

2K Bytes

Tip: Once the authentication is done, all the data blocks of the same sector are free to access. For example, once the data block 0x04 is successfully authenticated (Sector 1), the data blocks 0x04 ~ 0x07 are free to access.

5.8.2. How to Access MIFARE Ultralight Tags?

Typical sequence may be:

- Scanning the tags in the field (Polling)
- Read / Write the memory of the tag
- Halt the tag (optional)

Step 1) **Polling** for the MIFARE Ultralight Tags, 106 kbps
HOST-> 02 6F 09 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 04 D4 4A 01 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 11 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 44 00 07 04 6E 0C A1 BF 02 84 90 00 [Checksum] 03
In which, Number of Tag found = [01]; Target number = 01
SENS_RES = 00 44; SEL_RES = 00,
Length of the UID = 7; UID = 04 6E 0C A1 BF 02 84
Operation Finished = 90 00

Step 2) **Read** the content of Page **04**
HOST-> 02 6F 09 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 05 D4 40 01 **30 04** [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [Checksum] 03
In which, Block Data = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Tip: 4 consecutive Pages will be retrieved. Pages 4, 5, 6 and 7 will be retrieved. Each data page consists of 4 bytes.

Step 3) **Update** the content of Page **04 with** the data "AA BB CC DD"
HOST-> 02 6F 0E 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 09 D4 40 01 **A2 04** AA BB CC DD [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Or

Step 3) **Write (MIFARE compatible Write)** the content of Page **04 with** the data "AA BB CC DD"
HOST-> 02 6F 0E 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 15 D4 40 01 **A0 04** AA BB CC DD 00 00 00 00 00 00 00 00 00 00 00 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Tip: This command is implemented to accommodate the established MIFARE 1K/4K infrastructure. We have to assemble the data into a 16 bytes frame. The first 4 bytes are for data, the rest of the bytes (12 ZEROS) are for padding. Only the page 4 (4 bytes) is updated even through 16 byte are sent to the reader.

Step 4) **Read** the content of Page **04 again**
HOST-> 02 6F 0E 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 05 D4 40 01 **30 04** [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [Checksum] 03
In which, Block Data = AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16

Tip: Only the page 4 is updated. Pages 5, 6 and 7 remain the same.

Step 5) **Halt the tag** (optional)
HOST-> 02 6F 0E 00 00 00 00 01 00 00 00
HOST-> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 45 [00] 90 00 [Checksum] 03

MIFARE Ultralight Memory Map.

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

512 bits
Or
64 Bytes

Please refer to the MIFARE Ultralight specification for more detailed information.

Appendix A. Topaz

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

Appendix B. Topaz512

EEPROM Memory Map (Segment0)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	00	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	25_h		Locked
Data	01	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	02	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	03	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	04	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	05	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	06	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	07	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	08	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	09	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes
Data	0A	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	0B	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	0C	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	0D									N/A
Lock/OTP	0E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	N/A
Lock/OTP	0F	OTP-6	OTP-7	LOCK-2	LOCK-3	LOCK-4	LOCK-5	LOCK-6	LOCK-7	N/A

EEPROM Memory Map (Segment1)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	10	Data96	Data97	Data98	Data99	Data100	Data101	Data102	Data103	Yes
Data	11	Data104	Data105	Data106	Data107	Data108	Data109	Data110	Data111	Yes
Data	12	Data112	Data113	Data114	Data115	Data116	Data117	Data118	Data119	Yes
Data	13	Data120	Data121	Data122	Data123	Data124	Data125	Data126	Data127	Yes
Data	14	Data128	Data129	Data130	Data131	Data132	Data133	Data134	Data135	Yes
Data	15	Data136	Data137	Data138	Data139	Data140	Data141	Data142	Data143	Yes
Data	16	Data144	Data145	Data146	Data147	Data148	Data149	Data150	Data151	Yes
Data	17	Data152	Data153	Data154	Data155	Data156	Data157	Data158	Data159	Yes
Data	18	Data160	Data161	Data162	Data163	Data164	Data165	Data166	Data167	Yes
Data	19	Data168	Data169	Data170	Data171	Data172	Data173	Data174	Data175	Yes
Data	1A	Data176	Data177	Data178	Data179	Data180	Data181	Data182	Data183	Yes
Data	1B	Data184	Data185	Data186	Data187	Data188	Data189	Data190	Data191	Yes
Data	1C	Data192	Data193	Data194	Data195	Data196	Data197	Data198	Data199	Yes
Data	1D	Data200	Data201	Data202	Data203	Data204	Data205	Data206	Data207	Yes
Data	1E	Data208	Data209	Data210	Data211	Data212	Data213	Data214	Data215	Yes
Data	1F	Data216	Data217	Data218	Data219	Data220	Data221	Data222	Data223	Yes

EEPROM Memory Map (Segment2)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	20	Data224	Data225	Data226	Data227	Data228	Data229	Data230	Data231	Yes
Data	21	Data232	Data233	Data234	Data235	Data236	Data237	Data238	Data239	Yes
Data	22	Data240	Data241	Data242	Data243	Data244	Data245	Data246	Data247	Yes
Data	23	Data248	Data249	Data250	Data251	Data252	Data253	Data254	Data255	Yes
Data	24	Data256	Data257	Data258	Data259	Data260	Data261	Data262	Data263	Yes
Data	25	Data264	Data265	Data266	Data267	Data268	Data269	Data270	Data271	Yes
Data	26	Data272	Data273	Data274	Data275	Data276	Data277	Data278	Data279	Yes
Data	27	Data280	Data281	Data282	Data283	Data284	Data285	Data286	Data287	Yes
Data	28	Data288	Data289	Data290	Data291	Data292	Data293	Data294	Data295	Yes
Data	29	Data296	Data297	Data298	Data299	Data300	Data301	Data302	Data303	Yes
Data	2A	Data304	Data305	Data306	Data307	Data308	Data309	Data310	Data311	Yes
Data	2B	Data312	Data313	Data314	Data315	Data316	Data317	Data318	Data319	Yes
Data	2C	Data320	Data321	Data322	Data323	Data324	Data325	Data326	Data327	Yes
Data	2D	Data328	Data329	Data330	Data331	Data332	Data333	Data334	Data335	Yes
Data	2E	Data336	Data337	Data338	Data339	Data340	Data341	Data342	Data343	Yes
Data	2F	Data344	Data345	Data346	Data347	Data348	Data349	Data350	Data351	Yes

EEPROM Memory Map (Segment3)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	30	Data352	Data353	Data354	Data355	Data356	Data357	Data358	Data359	Yes
Data	31	Data360	Data361	Data362	Data363	Data364	Data365	Data366	Data367	Yes
Data	32	Data368	Data369	Data370	Data371	Data372	Data373	Data374	Data375	Yes
Data	33	Data376	Data377	Data378	Data379	Data380	Data381	Data382	Data383	Yes
Data	34	Data384	Data385	Data386	Data387	Data388	Data389	Data390	Data391	Yes
Data	35	Data392	Data393	Data394	Data395	Data396	Data397	Data398	Data399	Yes
Data	36	Data400	Data401	Data402	Data403	Data404	Data405	Data406	Data407	Yes
Data	37	Data408	Data409	Data410	Data411	Data412	Data413	Data414	Data415	Yes
Data	38	Data416	Data417	Data418	Data419	Data420	Data421	Data422	Data423	Yes
Data	39	Data424	Data425	Data426	Data427	Data428	Data429	Data430	Data431	Yes
Data	3A	Data432	Data433	Data434	Data435	Data436	Data437	Data438	Data439	Yes
Data	3B	Data440	Data441	Data442	Data443	Data444	Data445	Data446	Data447	Yes
Data	3C	Data448	Data449	Data450	Data451	Data452	Data453	Data454	Data455	Yes
Data	3D	Data456	Data457	Data458	Data459	Data460	Data461	Data462	Data463	Yes
Data	3E	Data464	Data465	Data466	Data467	Data468	Data469	Data470	Data471	Yes
Data	3F	Data472	Data473	Data474	Data475	Data476	Data477	Data478	Data479	Yes

Reserved for internal use
User Block Lock & Status
OTP bits

Appendix C. Jewel64

EEPROM Memory Map (Segment0)										
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	25_h		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
Reserved	8	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	9	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	A	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	B	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Data	C	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Reserved	D	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Lock/OTP	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	N/A

Reserved for internal use
User Block Lock & Status
OTP bits

Appendix D. Jewel96

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