RD220 Serial USB RFID Reader

Demonstration Software User Manual

Document Version

1.0

Revision History

Revision	Date	Software Version	Description/ Change / Updated / Comment
1.0	March 2013	1.0	1st Release

Contents

List of Figures

List of Table

1. Introduction

The RD-220 is a series of 13.56MHz RFID reader. The RD-220 supports all major global secured baseband ISO standards namely ISO14443-A, -B, and ISO15693 as well as the MIFARE classic/plus cards.

This manual describes how to use demonstration software for evaluating and exploring various features of RD-220 through providing protocol. The RD-220 protocol contains basic commands as stated in the ISO protocol and combo commands for one-stop operation as well as arbitrary commands for higher layer protocol communication. Performing RFID operation through the providing protocol is suitable for system integrators or hardware/software developers who have to be involved in the 13.56MHz RFID applications. Demonstration software is designed to guide user/developer to quickly understand the RD-220 protocol and be capable to implement your own RFID software applications. Moreover, the open source C# of the demonstration software is also provided in CD for reference.

The cards or cards that RD-220 can support are summarized in Table 1-1.

Table 1-1 Card or card that RD-220 supports

Standard	Cards or cards that RD-220 can supports
ISO 14443A	MIFARE Family, ISO14443A memory card, ISO14443A smart card
ISO 14443B	Type-B SRI Series (ST), ISO14443B memory card, ISO14443B smart card
ISO 15693	SIC5600 (SIC), I-Code Family (NXP), Card-it (TI), LRI Series (ST), ISO15693 Label

2. Getting Started

Before user can operate the demonstration software, proper operational environment and the following requirements must be prepared.

2.1 System and Hardware Requirements

- Computer : PC with USB Port
- Operating System : Windows 7, Windows VISTA, Windows XP SP2
- RFID Reader : RD-220 Reader
- Software Requirement : .NET framework version higher than 4.0 Installed
- Others : Card/Tag

2.2 Content in CD

Content in CD, shown in Figure 1, consists of

- Demonstration Software in folder *"Demo_Software"*
- Documents namely Module Datasheets and Software Manuals in folder "*Documents*"
- .Net Framework in folder "*Dot_Net_Framework*"
- RD220 Driver for emulating RS232 on USB in folder *"RD220_Driver"*
- Example software source code in *"SW_Source_Codes"*

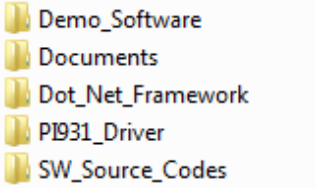


Figure 1 Content in CD

2.3 Software Installation

2.3.1 Driver Installation (For Virtual RS232 on USB)

1. Connect USB cable to the RD-220 reader Hardware. Then, there will be a “Found New Hardware” pop up indication.

Figure 2 A “Found New Hardware” pops up

2. Windows will ask for the way for driver installation. Select “No, not this time”.

Figure 3 Select “No, not this time”.

3. Windows will request for driver installation. Select Option “Install from a list or specific location (Advanced)”.

Figure 4 Select Option “Install from a list or specific location (Advanced)”

4. Select “Search for the best driver in these locations” and check “Include this location in the search” box. Then, Browse to folder “D:\RD220_Driver” in CD or where the driver is located. Click “Next”.

Figure 5 Browse to folder "RD220_Driver" in CD or where the driver is located

5. Windows ask to ensure installation the driver from SIC as shown in Figure 6; please click “Continue Anyway”.

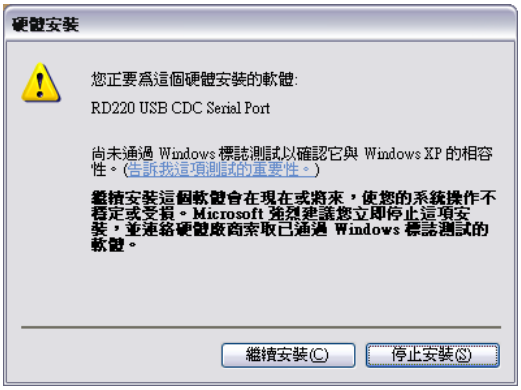


Figure 6 Click “Continue Anyway”

6. Wait for driver installation.

Figure 7 Windows is installing the driver.

7. If installation completes, there will a window indicating installation finish as shown below displayed. Please click Finish.

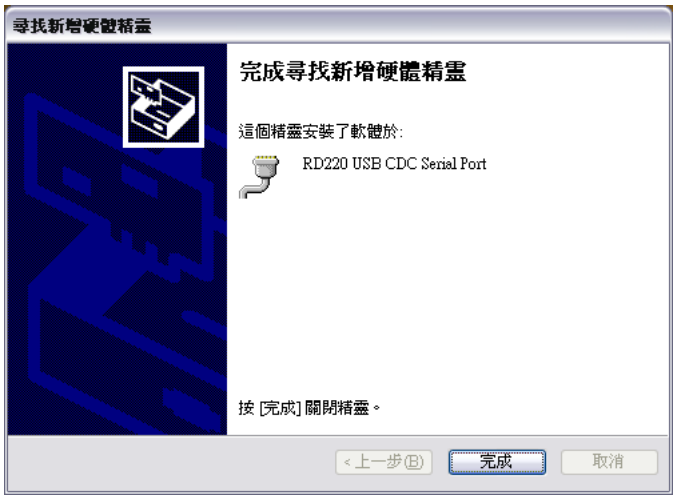


Figure 8 Installation completes.

8. There is the pop up shows hardware is ready to use. At this time, the device is successfully connected to PC and the LED on the module is turn on.

Figure 9 pop up shows hardware is ready to use

9. User can check the number of comport from Device Manager. As shown in Figure 10, the number of comport for this RD-220 hardware in this example is “COM4”. Note that the assigned number of comport is different for each computers.

Figure 10 COM Port for RD-220 Hardware displays in Device Manager.

3. Quick Start with Demonstration Software

The demonstration software is “SIC HF SDK.exe” provided in the folder *“Demo_Software”*. There is no software setup required; just double click the “SIC HF SDK.exe”. The demonstration software can run either from CD or a copy on hard drive. The GUI of software is shown in Figure 11 and ready to use.

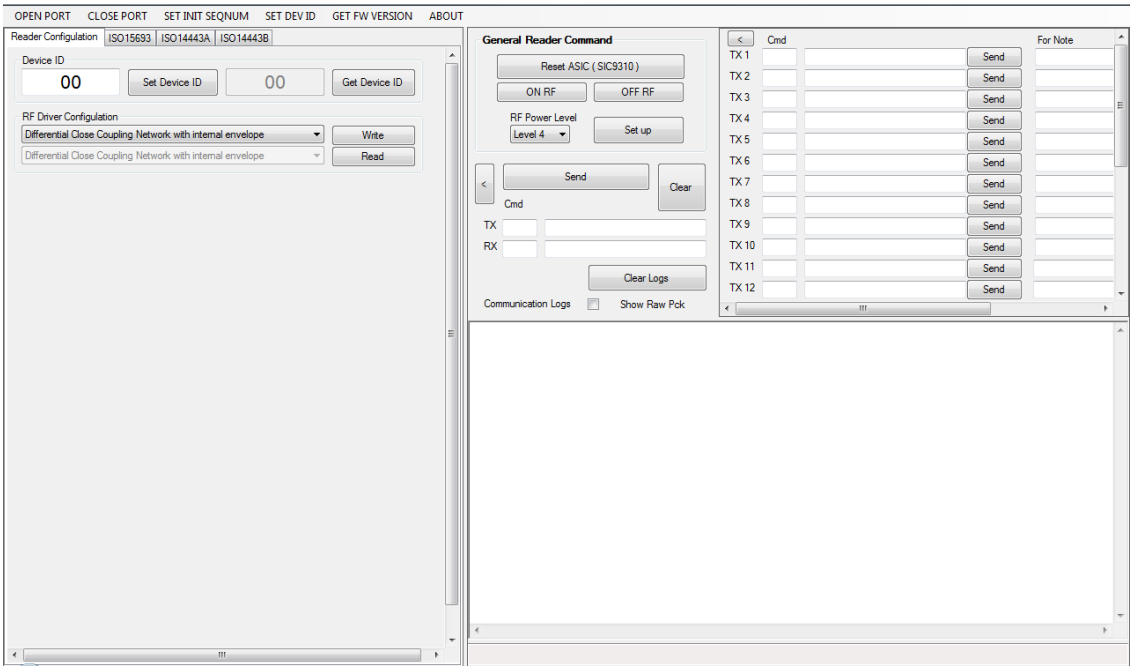


Figure 11 Demonstration software

Following steps, as shown in Figure 12, demonstrate a simple usage in reading UID of ISO15693 card for quick understanding.

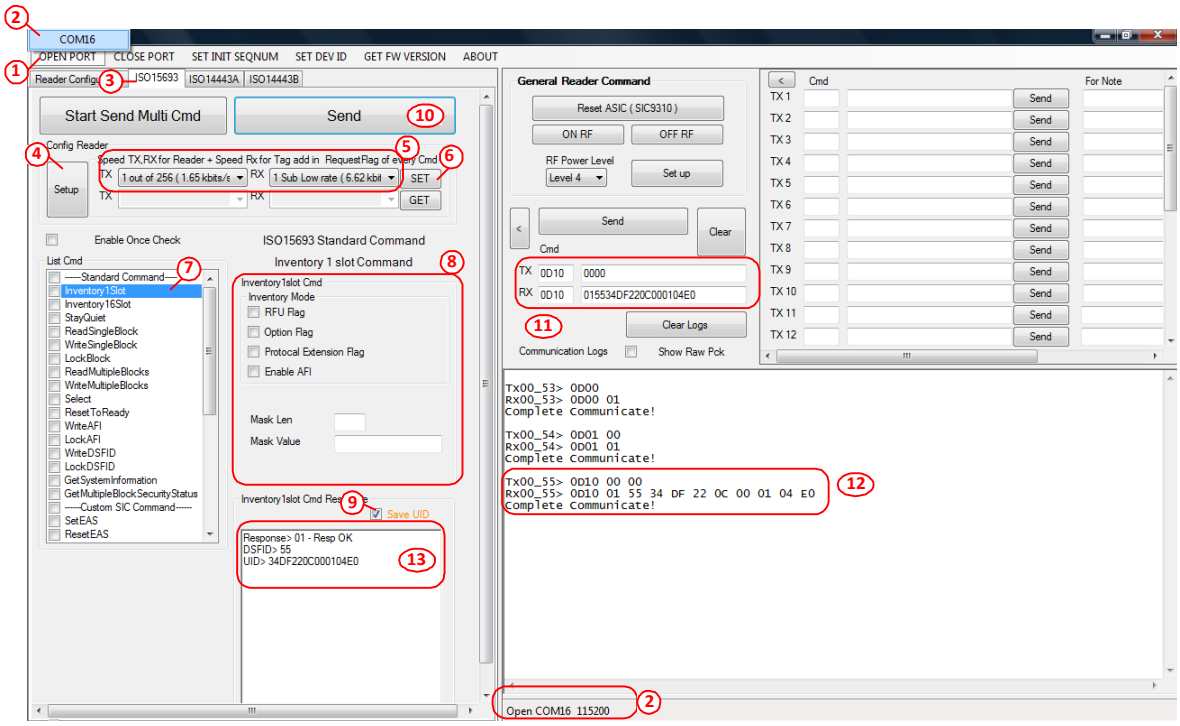


Figure 12 Demonstration software

1) Connect a Pi-931 device to computer and wait until computer recognize COMPORT as shown in Figure 10. Then click “OPEN PORT” Menu tap to query available COMPORT in computer.

2) Available COMPORT in computer is shown. Click the COMPORT number belonging to reader hardware. If connection is successful, there will be the connected comport with communication speed displayed at bottom of the GUI.

3) Click “ISO15693” tap to selection operation related to ISO15693

4) Click “Setup” button to configure reader IC to be ready to transmit and receive ISO15693 frame format

5) Select preferred speed using in RF transaction. In this example, select “1 out of 256 (1.65 Kbits/s)” for TX (downlink) and “1 Sub Low Rate (6.62 Kbits/s)” (Uplink).

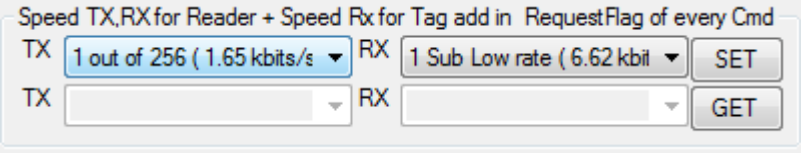


Figure 13 TX and RX Speed for reader CODEC in ISO15693 protocol

6) Click “SET” button, shown in Figure 13, to configure codec speed of reader

7) Select Command “Inventory 1 Slot” by clicking on the name of that command in order to read the card UID. The selected command will be highlighted.

8) Input necessary parameters. The parameters required in this command are Inventory mode, Mark Len and Mark Value. User can left these inputs blank in this case. Then, value of “0” is used.

9) Check “Save UID” check box to save readable UID in software buffer. This UID might be used in further operation.

10) Ensure that ISO15693 card is placed in operation range of the hardware reader. Click “Send” to transmit the command to reader hardware to read UID of ISO15693 card.

11) Transaction between reader hardware and PC is display in communication logs. For more information about response frame format, please refer to “Pi-931 Protocol” Document.

12) For standard commands in command list, the response is translated and displayed in command response section for easy understand. In this case, response is successful by response code “01” while DFSID and UID are shown.

Note that user can watch transactions between reader hardware and PC to be ideas in developing application from each step.

4. Demonstration Software Component

This software mainly consists of four sections as shown in Figure 14 namely

1) Port-and-Constant Setup
2) RFID standard taps and Reader Configuration tap
3) Raw data input and output for developer
4) Transaction logs.

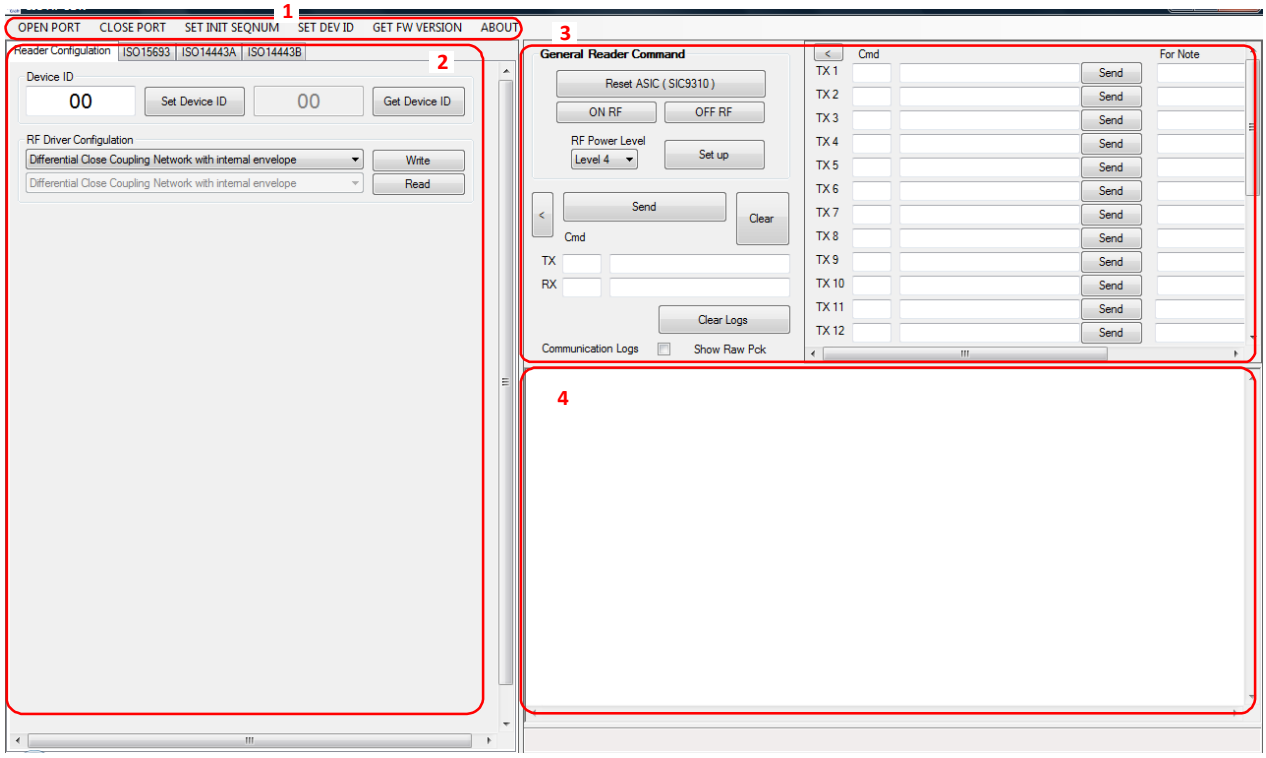


Figure 14 Sections in Demonstration software

4.1 Port-and-Constant Setup

This section consists of Menus related to hardware setup namely OPEN PORT, CLOSE PORT, SET INIT SEQNUM, SET DEV ID, GET FW VERSION and ABOUT.

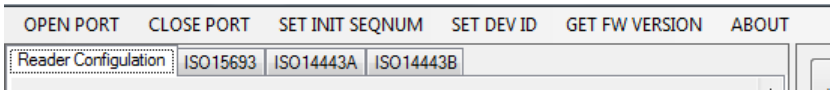


Figure 15 Port-and-Constant Setup

4.1.1 OPEN PORT

OPEN PORT is used to query and open communication port to the reader device.

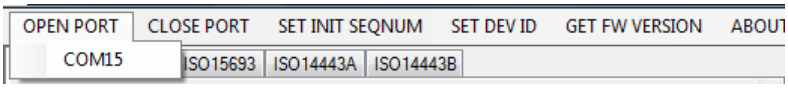


Figure 16 **OPEN PORT** Menu

Following steps describe opening the communication port with Pi-931.

- Click **OPEN PORT** menu to search available com port present in computer.
- Available COM ports are shown in menu content under **OPEN PORT** menu.
- Click on the COM port number belonging to reader hardware being operated to open communication.
- If connection is successful, there will be a connected comport with communication speed displayed at bottom of the GUI, as shown in Figure 17.

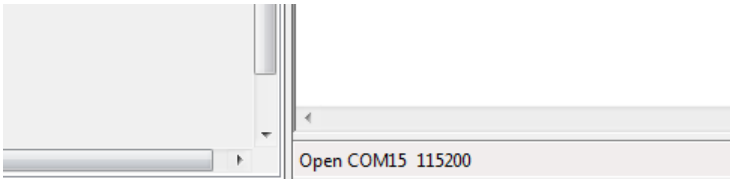


Figure 17 connected com port and communication speed (kbps)

4.1.2 CLOSE PORT

CLOSE PORT is used to close current communication port.

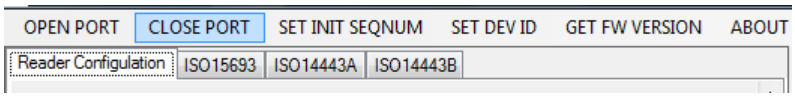


Figure 18 **CLOSE PORT** Menu

- Click **CLOSE PORT** to close current operating COM port.
- The connection shown at bottom of the GUI disappears.

4.1.3 SET INIT SEQNUM

SET INIT SEQNUM is used to set initial sequence number **SEQNUM** in transmission command packet. The **SEQNUM**, 1-byte information, is automatically increased after transmission of each packet by this software. As shown in Figure 19, user can input two-hexadecimal number to set the initial sequence number.

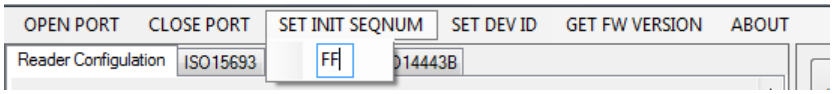


Figure 19 **SET INIT SEQNUM** Menu

4.1.4 SET DEV ID

SET DEV ID is used to define target Device ID in transmission command packet. The Device ID (Dev ID) indicates specific device or all in a network to operate. The Dev ID is consisted of an ID and a silent control bit. The ID can be between 0x00 and 0x7F. The most significant bit in the Dev ID is the silent bit in which the operating devices will not response back to host. The device that receives matched ID from incoming packet will operate and respond back to host. The Dev ID in this setting can be from 0x00 and 0xFF. Then, there will be no response from packet containing the Dev ID between 0x80 and 0xFF. For Device ID of 0x00, all devices will operate and response back to host. The detail of Dev ID is summarized in Table 4-1. User can define Dev ID in packet by entering two- hexadecimal number as shown in Figure 20.

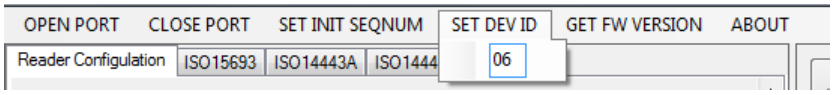


Figure 20 **SET DEV ID** Menu

Table 4-1 Dev ID		
Silence Bit [1] Dev_ID[7]	**Device ID** Dev_ID [6:0]	**Meaning**
0	0x00 [2]	All devices that receive command operate and respond back to host with Dev_ID of 0x00.
0	0x01 – 0x7F	The ID-matched device operates and responds back to host.
1	0x00	The operating device will not respond back to host

(1) The silence bit is an option for preventing data collision in reader network from simultaneous response.

(2) Device ID 0x80 can use for broadcasting if there are multiple readers connected in a network.

4.1.5 GET FW VERSION

GET FW VERSION is used to get firmware version from the connected device. Firmware version is reported as shown in Figure 22

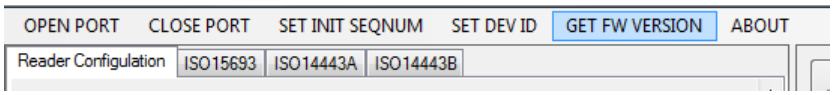


Figure 21 **GET FW VERSION** Menu

4.2 RFID standard taps and Reader Configuration tap

This section consists of three RFID standard taps and Reader Configuration tap. GUI in ISO15693 standard tap is shown in Figure 23. In ISO14443A and ISO14443B tap contains similar component with different commands. For Reader configuration as shown in Figure 26, the provided GUI is for setting up and getting Device ID and RF Driver configuration.

4.2.1 RFID standard taps

Each RFID standard tap contains data transmission button, RF communication speed selection, parameters setup of each associated command as well as response reports from RF transaction of each command.

4.2.1.1 "Standard Setup" Button

"Setup" Button is used to setup parameters in the reader to be ready to transmit and receive following standard in tap name. User must activate this setup before performing any RF-related operations in associated RFID protocol.

4.2.1.1 "Send" Button

"Send" Button is used to transmit command that is currently highlighted in command list

4.2.1.2 "Start Send Multi Cmd" Button

"Start Send Multi Cmd" button is used to transmission multiple commands consecutively that are checked in command list. From example in Figure 23, providing that no any command fail during each operation, ***Inventory 1 Slot***, ***Read Single Block***, ***Write Single Block*** and ***Select*** are operated in chronological order respectively. Note that command in the list is run from top to button.

4.2.1.3 Reader TX and Rx Speed selection

This section is used to setup transmission and reception speed of CODEC in reader IC. User must setup the transaction speed before performing any RF-related operations in associated RFID protocol. In addition, current setup speed in the reader can be retrieved by "Get" button.

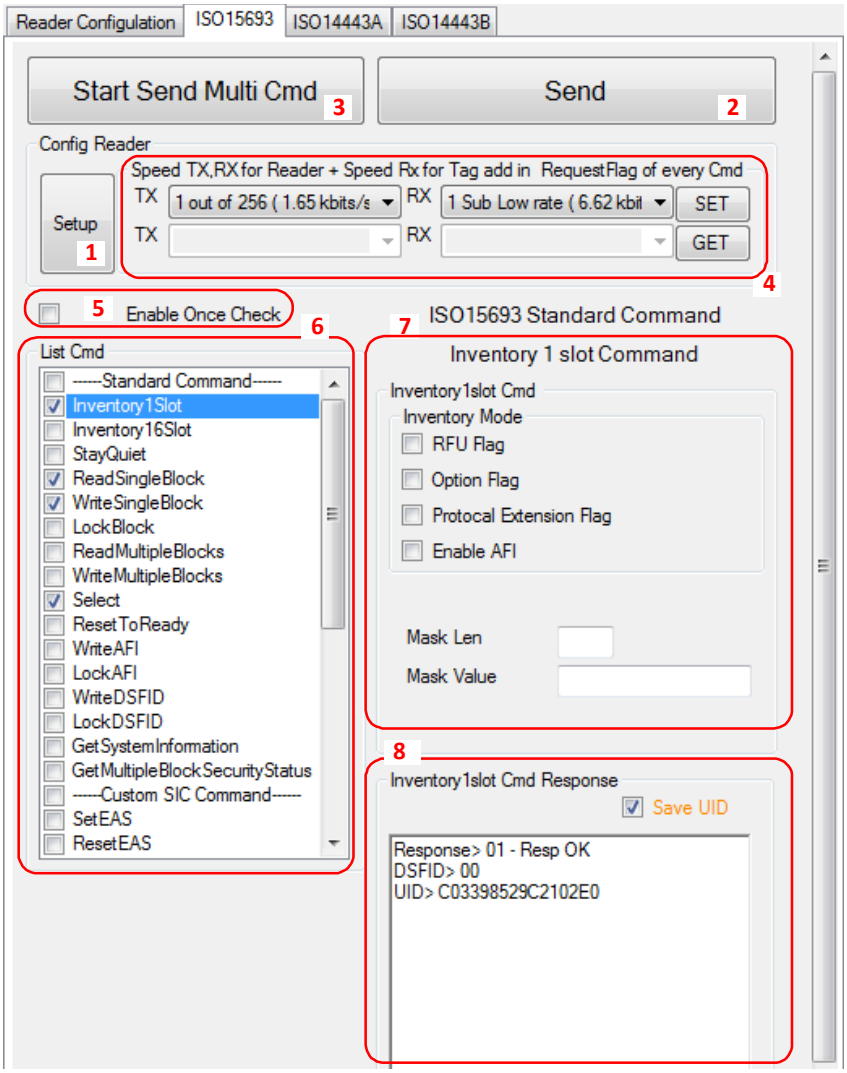


Figure 23 Example of button and setup in ISO 15693 tap

4.2.1.4 "Enable once check" Check box

The "Enable Once check" is an option in selecting command on command list. If "Enable Once check" box is checked, a pointed command in command list is checked on one click.

4.2.1.5 Command List

The command list provides air-interface commands in each RFID standard. The commands include primitive command based on operating RFID standard and combo command for one stop operation as well as some proprietary commands for SIC RFID.

4.2.1.6 Parameter setup

The section is for setting up parameters required by each command. The section is different from command to command. User may have to refer to ISO standards for meaning and function of each parameter. If the inputs are left blank, value of "0" is used.

In some command, there is a "Load Saved UID" checked box to load UID from previous operation to use in current operation. For the command required UID in operation, the "Load Saved UID" checked box is provided.

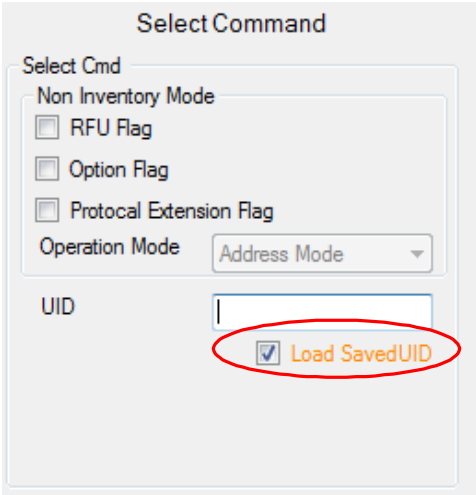


Figure 24 "Load Saved UID" checked box

4.2.1.7 Command Response

The Command Response section displays and translates operation result and available response of each command. As shown in Figure 25, "01" indicates response is successful without any error. Also, received UID and DSFID in ISO15693 are displayed. The UID in response packet starts from least significant byte to most significant byte which reflects to what transmit from card in chronological order. For more information about meaning of response, please refer to Pi-931 Protocol.

In some command, there is a "Save UID" check box for storing UID form current transaction for further operations. To store UID in software buffer, this checked box must be set prior operating command.

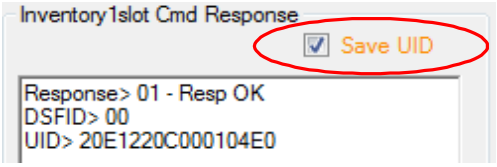


Figure 25 "Save UID" checked box

4.2.2 Reader Configuration tap

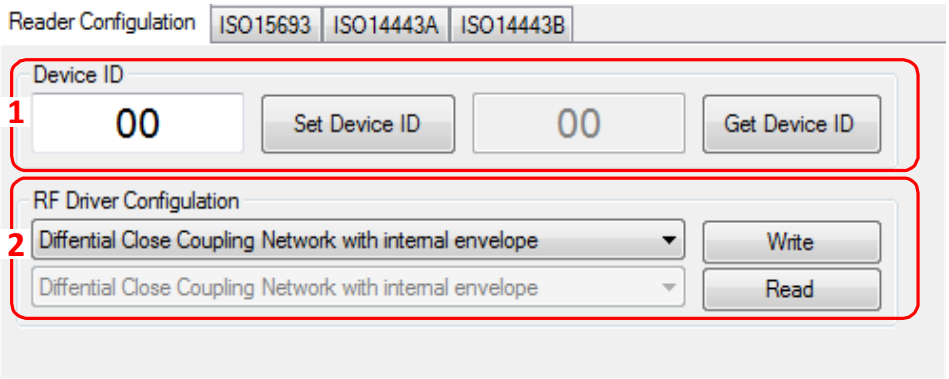


Figure 26 Command and setup in Reader Configuration tap

4.2.2.1 "Device ID" Setup

"Set Device ID" is for programming Device ID for individual reader. Only device ID between 0x00 and 0x7F is allowed. Also, device ID of connected device can be read from "Get Device ID" button.

4.2.2.2 "RF Driver Configuration" Setup

This section is for setup driver configuration of the hardware. The driver configuration is information used by microcontroller to set up driver characteristic to serve specific RF topology in Pi931 family. This information is stored in EEPROM of SIC9310. To make the reader properly operable and achieve the highest performance, the driver configuration must be specifically set to match to RF topology of hardware. The RF topology can be one of these configurations namely

1. Differential driver with internal envelop detector,
2. Differential driver with external envelop detector,
3. Single ended driver with external envelop detector,
4. 50-ohm-output Class-E driver with external envelop detector.

For every Pi-931 reader manufactured from silicon craft, the "RF Driver Configuration" is pre-programmed to match RF driver topology of the reader hardware. So, the "RF Driver Configuration" is provided for users who designs their own hardware based on Pi931-MD module and requires to reconfigure hardware constant to different RF driver topology.

4.3 Raw data input and output for developer

This section is provided for developers to communicate with the hardware by hexadecimal code. Example of usage this section are transmitting commands not provided in command list. This is a common situation in 13.56-MHz RFID application development. Moreover, this section can be used for debugging developing applications. To use this section, users have to input command in Hexadecimal format in TX and decode responses from hardware shown in RX by user defined programming or manually. For more information about module communication protocol and hexadecimal code, please refer to Pi-931 protocol document. The section is consisted of six components, which will be described below.

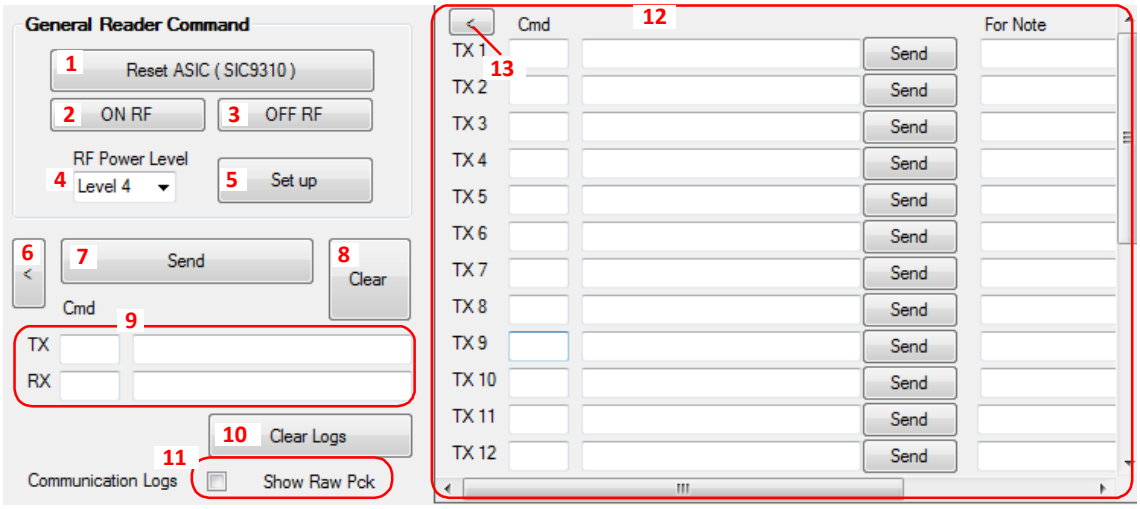


Figure 27 Raw data input and output for developer

4.3.1 "Reset ASIC" button

This button is used to reset reader IC (SIC9310)

4.3.2 "ON RF" button

This "ON RF" button starts 13.56-MHz carrier emission.

4.3.3 "OFF RF" button

This "OFF RF" button stops 13.56-MHz carrier emission.

4.3.4 "RF Power Level" Selection

This pull down menu is for selecting transmitting strength of 13.56-MHz carrier.

4.3.5 "Set up" button (Power Level)

The Set up button is for configuring strength of transmitting carrier following defined RF Power Level.

4.3.6 "Width Restore (<)" button

This button is used to restore width and show all part of sub panel "RFID standard taps".

4.3.7 "Send" button

The "Send" button is used to transmit command and data in TX box in section 4.3.3.

4.3.8 "Clear" button

The "Clear" button is used to clear content in section 4.3.3.

4.3.9 Direct TX Input and RX Output

TX Input consists of command box and its associated data to be send to RFID reader while RX output consists of feedback command mode and response data from RFID reader. As shown in Figure 28, "0x0A13", which performs select command in ISO14443A, is the transmitted command and "0x001CBB4FCD" is the transmitted data to RFID reader. In RX box, "0x0A13" is feedback command to confirm operation and "0x0108" is response data from RFID reader.

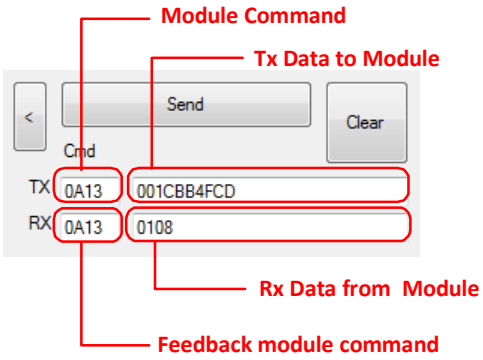


Figure 28 Example of Direct TX Input and RX Output

4.3.10 "Clear Logs" button

The "Clear Logs" button is used to clear all logs event in Transaction Log display.

4.3.11 "Show Raw Pck" Check box

If "Show Raw Pck" is checked, all every transaction between reader and host is displayed in full format. This will help user understand frame format structure of Pi-931 Protocol from header to checksum. The effect of "Show Raw Pck" is shown in Figure 29.

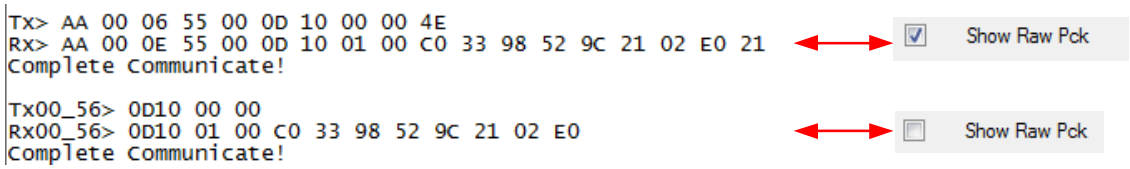


Figure 29 Effect of "Show Raw Pck" in transaction display

4.3.12 Additional TX Input and RX Output section

The function of this section is similar as described in 4.3.4. The purpose of this section is for testing consecutive arbitrary command unavailable in command list. User can review each result in the Transaction Log. In addition, the box on the most right is for recording and comment. This help developer recognize meaning of each hexadecimal command. Usage of this section is shown in Figure 30 Usage of additional TX input and RX output sectionFigure 30.

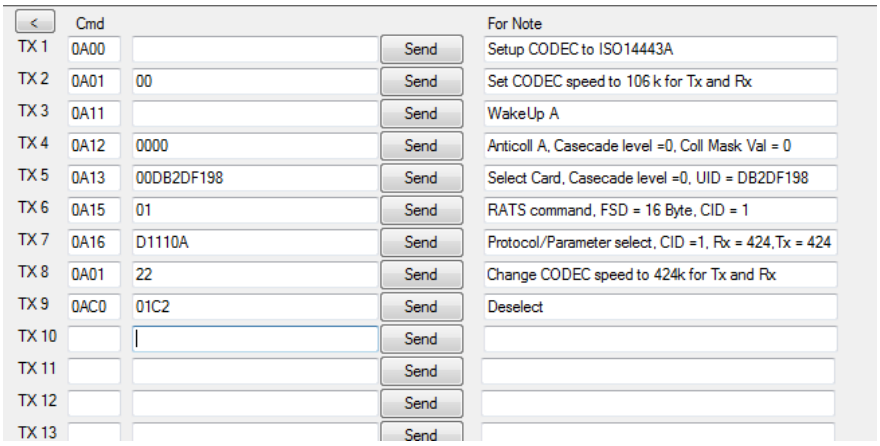


Figure 30 Usage of additional TX input and RX output section

4.3.13 "<" button

The "<" button in "Additional TX Input and RX Output section" is used to extend or shrink this section for easy accessibility. When the symbol on the button is "<", click the button extend the section. Once, the symbol on the button become ">", click the button shrink this section.

4.4 Transaction Logs

If "Communication Logs" check box is checked, transaction between host and the device is logged and displayed in this section as shown in Figure 31.

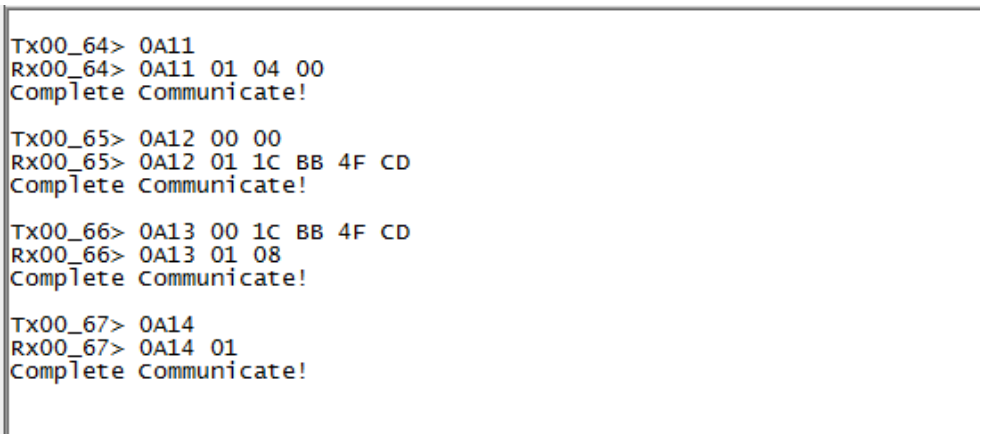


Figure 31 Transaction Logs text box.

5. Pi-931 Protocol

To operate with Pi-931 module effectively, user should prior be familiar with technical terms, transaction speeds and parameters required in ISO14443 and ISO15693 as well as MIFARE. The command available in this demonstration software is directly mapped from existing command in the protocol. So, user can understand how to interact with card step by step by manually inputting and activating commands at appropriate state of card and can learn how to use our protocol and compose correct structure from this demonstration software.

To create application based on Pi-931 module, user should be able to transmit command in hexadecimal code and review response from the reader device. The Pi-931 protocol itself is basically a series of hexadecimal code. The command frame format is shown Figure 32 while Table 5-1 summarize meaning of bytes in frame. For the response frame, frame format detail is shown Figure 33 and Table 5-2.

Name	SOP	LENG-H	LENG-L	Seq Num	Dev_ID	CMD Category	CMD	Data[0]… Data[n-1]	LRC
Values	0xAA	0x00	0x00						
No. Byte	1-byte	1-byte	1-byte	1-byte	1-byte	1-byte	1-byte	n-byte	1-byte

Figure 32 Command Frame Format

Table 5-1 Meaning of byte in Command Frame Format	
Name	**Meaning**
SOP	Start-of-Package byte (0xAA)
LENG-H	High byte of packet length counting from sequence number to Data[n-1]
LENG-L	Low byte of packet length counting from sequence number to Data[n-1]
Seq Number	Sequence number of Package
Dev_ID	Device ID byte : Silence bit (1Bit MSB) + Device_ID (7 Bit)
CMD Category	Command Category byte to specify operating standard or reader setup mode
CMD	Command byte in specified Command Category standard
Data[0] … Data[n-1]	Data bytes
LRC	Check sum of the packet which is XORing result from LENGTH to Data[n-1]

Name	SOP	LENG-H	LENG-L	Seq Num	Dev_ID	CMD Category	FBP CMD	Resp	Data[0]… Data[n-1]	LRC
Values	0xAA	0x00	0x00							
No. Byte	1-byte	1-byte	1-byte	1-byte	1-byte	1-byte	1-byte	1-byte	n-byte	1-byte

Figure 33 Command Frame Format

Table 5-2 Meaning of bytes in Response Frame	
Name	**Meaning**
SOP	Start-of-Package byte (0xAA)
LENG-H	High byte of packet length counting from sequence number to Data[n-1]
LENG-L	Low byte of packet length counting from sequence number to Data[n-1]
Seq Number	Sequence number of operated command packet
Dev_ID	Response device ID byte : 0 (1 Bit) + ID of operating Device (7 Bit)
CMD Category	Operated Command Category byte
FBP CMD	Operated command byte
Resp	Response flag of operated command
Data[0] … Data[n-1]	Response Data Bytes
LRC	Check sum of the packet which is XORing result from LENGTH to Data[n-1]

The relation between hexadecimal of command and response displayed in transaction logs window and frame format as stated above is shown in Figure 34. This example shows performing inventory1slot in ISO15693. To display only necessary numbers in command and response packet, user can uncheck “Show Raw Pck” as explain in 4.3.9. Figure 35 depicted decomposition of abridged frame format.

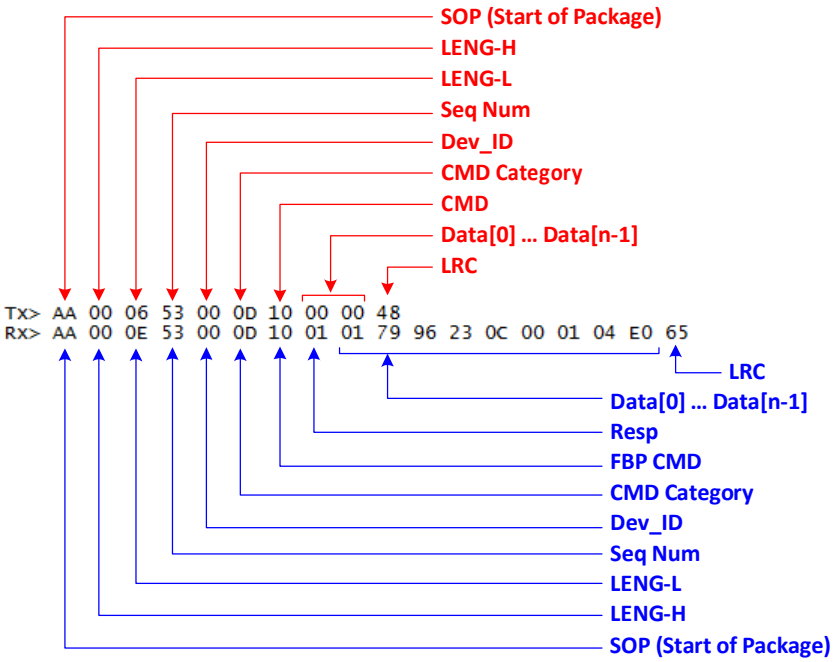


Figure 34 Decomposition of raw frame format of Pi-931

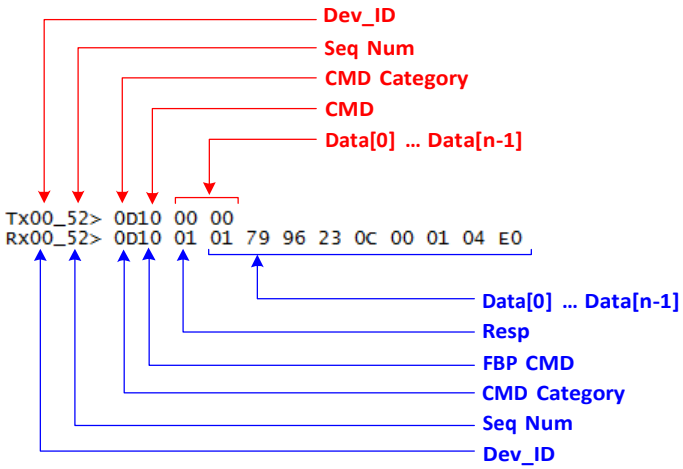


Figure 35 Decomposition of abridged frame format displayed in demonstration software

6. Using Demonstration Software

6.1 ISO14443A

Figure 36 shows GUI in ISO14443 tap. In the tap, user can transmit single or multiple commands, setup transaction speed as well as review card response. Command list consists of standard commands, MIFARE commands, Combo commands and transparent commands.

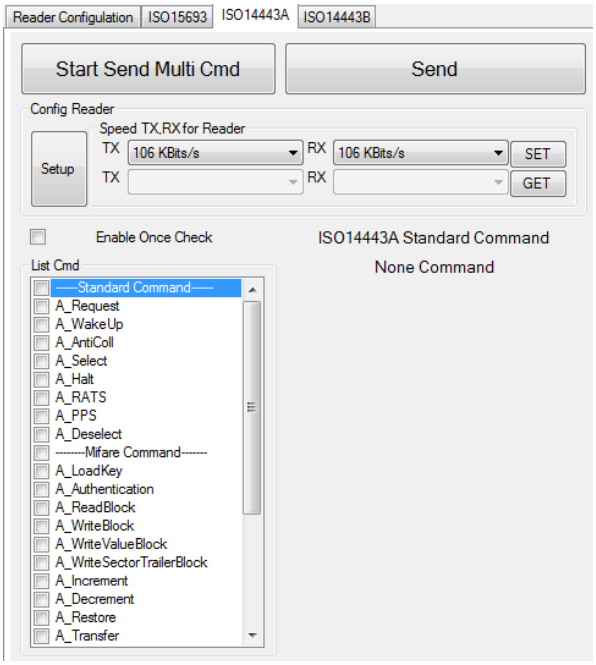


Figure 36 ISO14443A command list

6.1.1 ISO14443A standard commands

The standard commands used in ISO14443A-3 and -4 are illustrated in Figure 38, showing relation of each command in card-state transition diagram. Hence, each command shall be applied at appropriate state. The commands in this section are **A_Request**, **A_WakeUp**, **A_Anticoll**, **A_Select**, **A_RATS**, **A_PPS**, **A_Halt** and **A_Deselect**. For more information about the commands, please refer to ISO14443-3 and -4 standards and Pi-931 Protocol.

6.1.1.1 A_Request

The **A_Request** command GUI is shown Figure 37. This command is used to probe if ISO14443A cards are in there field. This command requires no input.

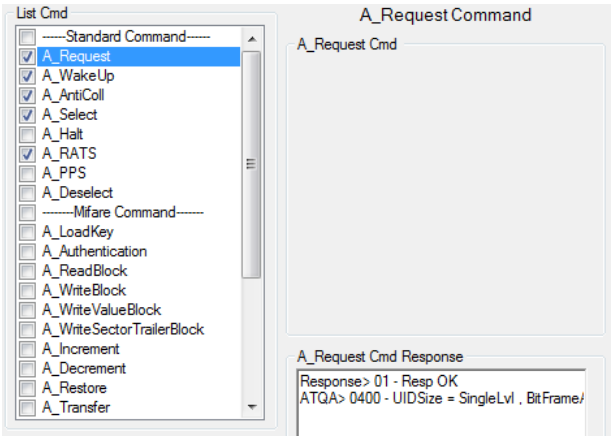


Figure 37 **A_Request** Command

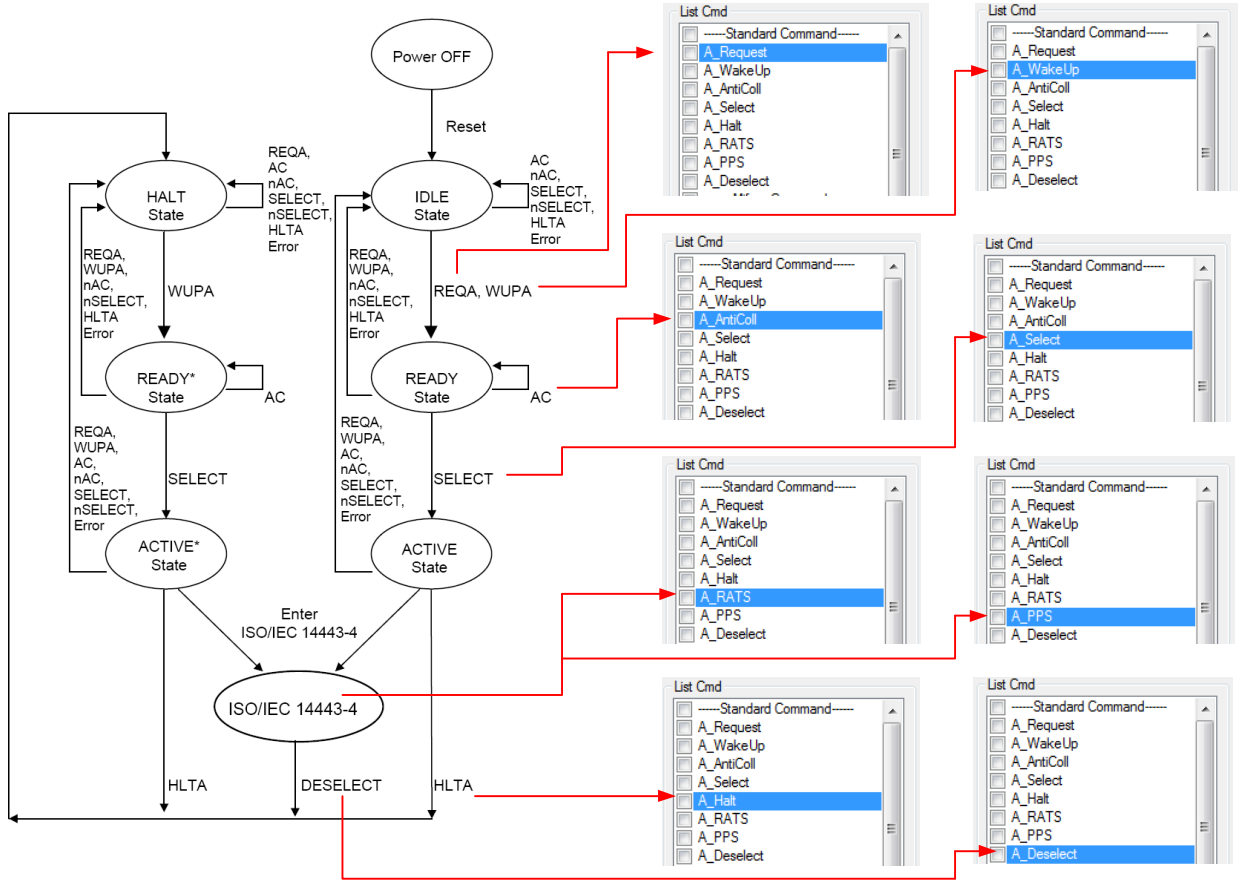


Figure 38 ISO14443A standard commands available in Pi-931

6.1.1.2 A_WakeUp

The **A_WakeUp** command GUI is shown Figure 39. This command is used to probe if ISO14443A cards are in there field. This command requires no input.

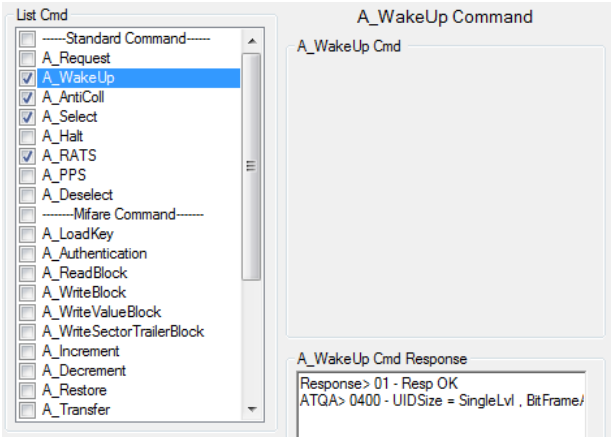


Figure 39 **A_Wakeup** Command

6.1.1.3 A_Anticoll

The **A_Anticoll** command GUI is shown Figure 40. The **A_Anticoll** not only transmits command as in the standard but also automatically seek out a complete UID from cards in the RF field. Providing that the cards performed anti-collision are in the same range, the parameter CollMaskVal, as shown in Figure 40, in **A_Anticoll** can be used to select a card that condition matches. If the CollMaskVal is 0, the reader selects the UID from the card

where the first collided bit is 0, and vice versa. The example of CollMaskVal usage is shown in Figure 41. If CollMaskVal is 1, UID of "3A 2A 10 92"is selected. If CollMaskVal is 0, UID of "0A BD 0F 92" is selected. The collision occurs at the fifth bit where first byte response from one card is 0x3A (00111010) and other is 0x0A (00001010). Note that shown UID in response packet starts from least significant byte to most significant byte which reflects to what transmits from card in chronological order. However, other factors such as transmission energy, size of antenna and coupling factor from card and others in RF environment introduce performance of this feature CollMaskVal. This feature can work well in reader equipped with proper antenna. Moreover, the **A_Anticoll** supports UID cascade level from level-1 to level-3.

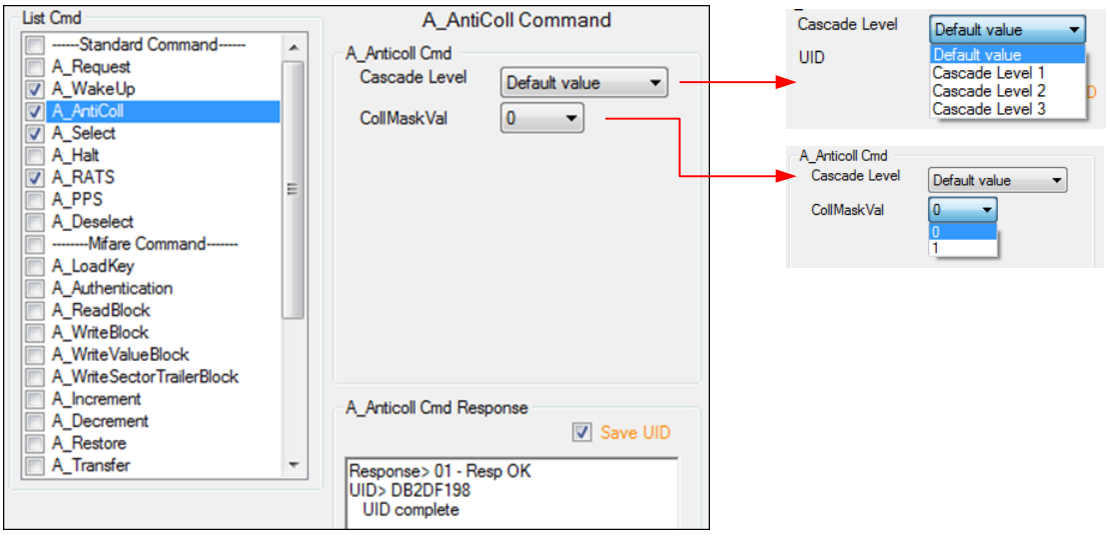


Figure 40 Parameters in **A_AntiColl** command

```
Tx00_90> 0A11
Rx00_90> 0A11 01 04 00 <-- ATQA
Complete Communicate!

Tx00_91> 0A12 00 01
Rx00_91> 0A12 01 3A 2A 10 92 <-- UID ( CollMaskVal = 1 )
Complete Communicate!

Tx00_92> 0131 <-- OFF Field
Rx00_92> 0131 01
Complete Communicate!

Tx00_93> 0130 <-- ON Field
Rx00_93> 0130 01
Complete Communicate!

Tx00_94> 0A11
Rx00_94> 0A11 01 04 00 <-- ATQA
Complete Communicate!

Tx00_95> 0A12 00 00
Rx00_95> 0A12 01 0A BD 0F 92 <-- UID ( CollMaskVal = 0 )
Complete Communicate!
```

Figure 41 Result ID from **CollMaskVal**

6.1.1.4 A_Select

The **A_Select** command GUI is shown Figure 42. This command requires complete UID from the **A_Anticoll** command before operating. The **A_Select** supports UID cascade level from level-1 to level-3.

6.1.1.5 A_RATS (Request for answer to select)

The **A_RATS** command GUI is shown Figure 43. This command requires FSD (Maximum PCD frame size) and CID (Card Identifier). This command shall be activated after card is selected the active state by **A_Select** command. The response from this command indicates card capabilities namely transaction speed, Frame wait time, Start-up Frame guard time, NAD support, CID support and History bytes. For more information about meaning of the parameters, please refer to ISO14443-4. This command is in ISO14443A-4 which is the beginning point in entering smartcard protocol ISO7816.

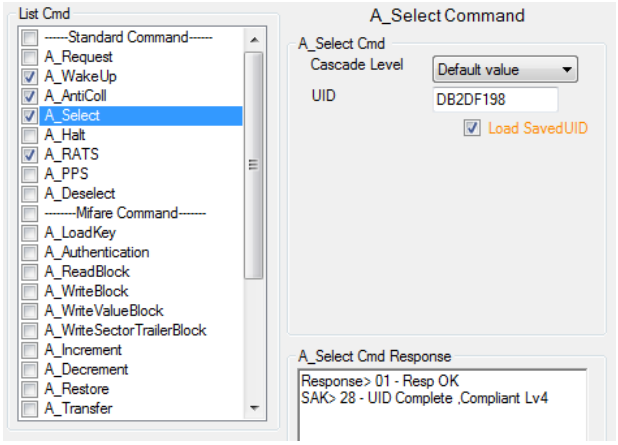


Figure 42 Parameters in **A_Select** command

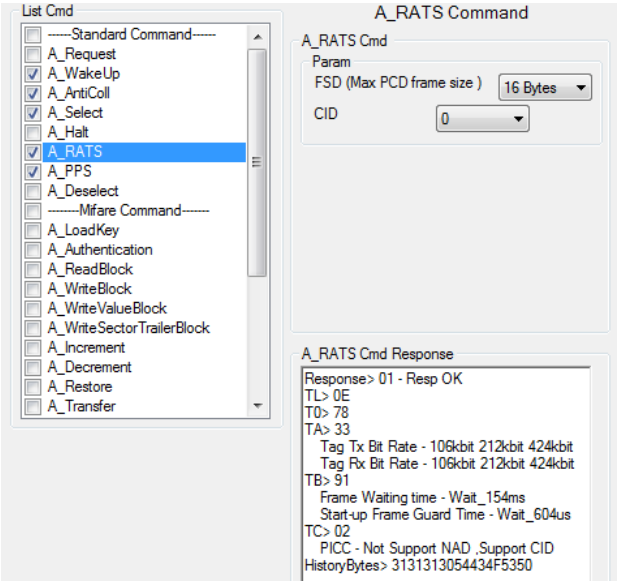


Figure 43 Parameters in **A_RATS** command

6.1.1.6 A_PPS (Protocol and parameter selection request)

The **A_PPS** command GUI is shown Figure 44. This command is used to select operating protocol and parameter to use in further operation. This command requires CID, PPS0 and PPS1. This command shall be activated next to **A_RATS** command. After executing this command, user shall set the transaction speed match to what card response. For more information about meaning of the parameters, please refer to ISO14443-4.

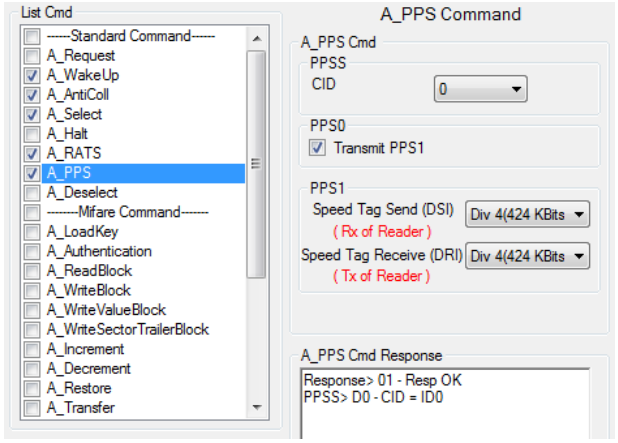


Figure 44 Parameters in **A_PPS** command

6.1.1.7 A_Deselect

The **A_Deselect** command GUI is shown Figure 45. This command is used to quit operation of ISO14443-4 or higher layer to idle state of ISO14443-3. This command requires no input and, actually, this command is made up of **A_TransparentWithCRC.** For more information about meaning of the parameters, please refer to ISO14443-4.

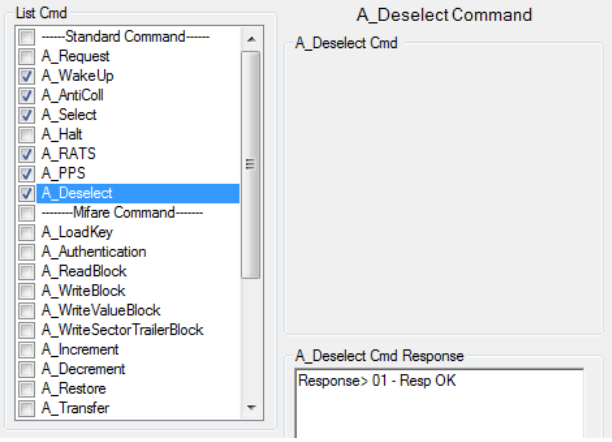


Figure 45 **A_Deselect** command

6.1.2 Performing Basic operation in ISO14443A standard

Figure 46 shows steps in performing basic command in ISO14443A from wake up to Halt through "Start Send multiple cmd" button. The steps are described as follows.

1. Select ISO14443A Tap
2. Click "Setup" to initialize reader to received ISO14443A frame
3. Define transaction speed. For ISO14443A, speed must be start from 106kbps for both TX and RX.
4. Click "Set" to configure transaction speed
5. Select command to be run by checking box in front of the command. In this example, **A_WakeUp, A_Anticoll, A_Select** and **A_Halt** are selected.
   - 5.1 Select Cascade Level in **A_Anticoll** command to Default value and ensure that "Save UID" box is checked.
   - 5.2 Select Cascade Level in **A_Select command** to Default value and ensure that "Load Saved ID" box is checked
6. Click "Start Send Multi Cmd" button to run all selected command in defined order. Ensure that ISO14443A card is placed in vicinity of reader.
7. If no error occurs, transaction results of each step are shown.
8. Results from running each command are shown in associated response box.

   Note that if user requires running command not in provided order; please use "Raw data input and output" section as described in 4.3.10.

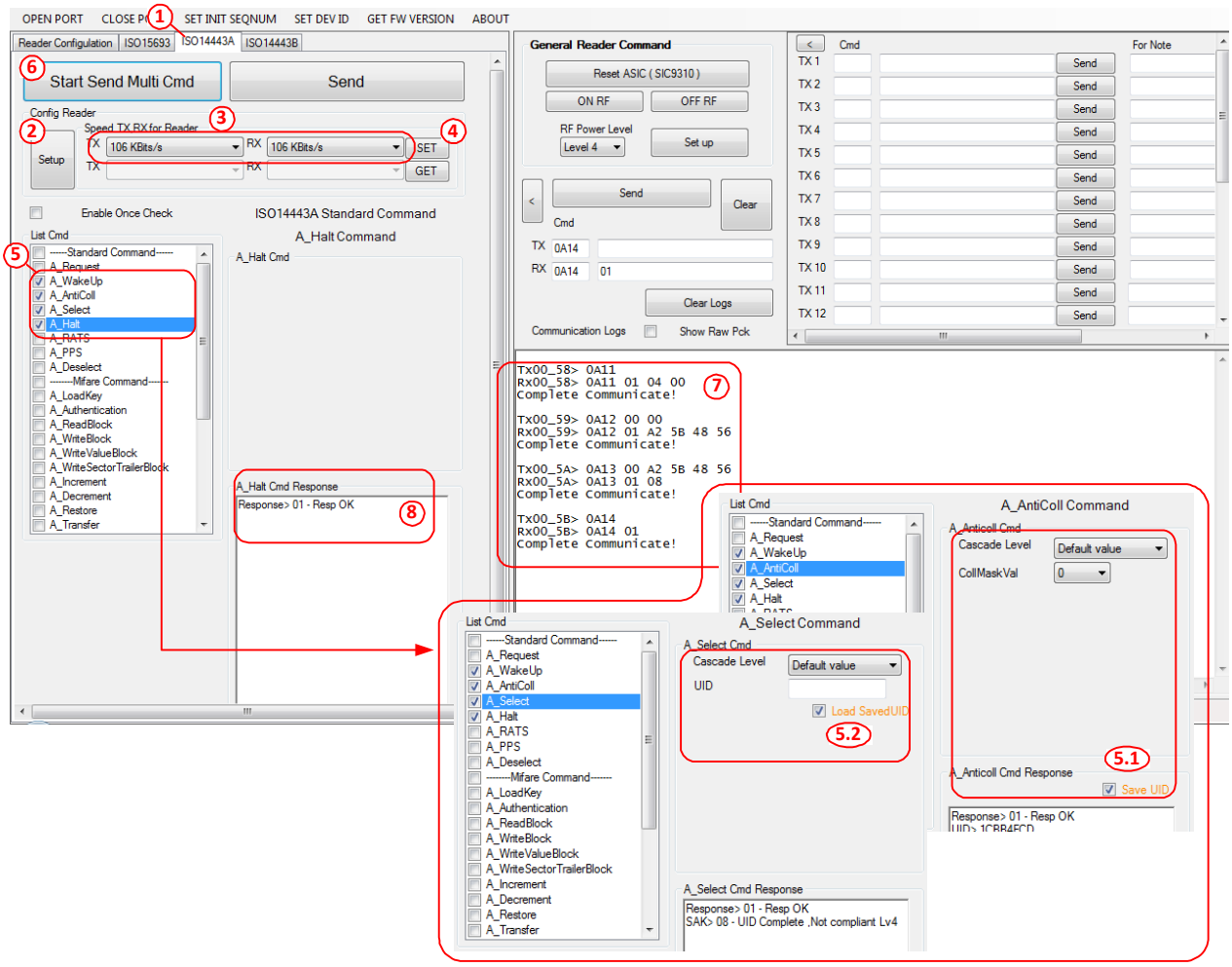


Figure 46 Running multiple commands in ISO14443A

If commands to be transmitted are not provided in command list such as command for smart card, user can use **A_TransparentWithCRC** and **A_TransparentWithoutCRC** to directly transmit hexadecimal code to air. User may have to select preferred timeout period for no response, if some smartcard take long time in processing before response. The **A_TransparentWithCRC** and **A_TransparentWithoutCRC** GUI is shown in Figure 47.

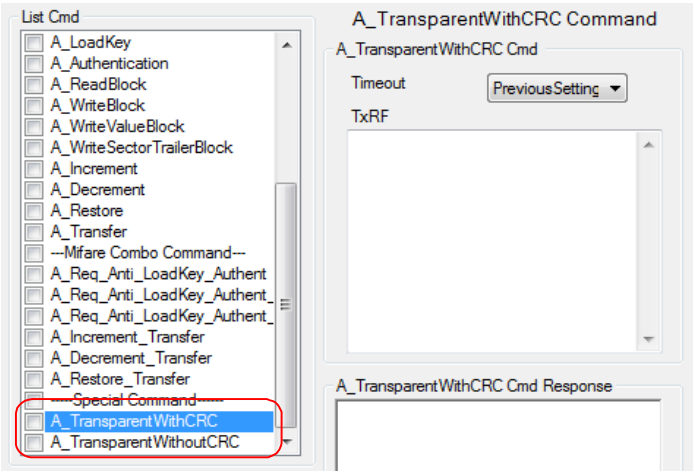


Figure 47 **A_TransparentWithCRC** and **A_TransparentWithoutCRC** for ISO14443A

6.1.3 MIFARE

MIFARE is encryption applications based on ISO14443A and operate on transaction speed of 106kbps. Before performing MIFARE command, a card must be prior selected in active state. The MIFARE commands Pi-931 can support are shown in Figure 48. There are basic MIFARE commands and MIFARE combo commands. The basic command are single operation commands as stated in MIFARE datasheet while combo commands are consecutive specific-operation commands from REQA to specific command in MIFARE such as read and write. The combo commands ease user to shorten reader operating time, development time and complexity of coding in user application.

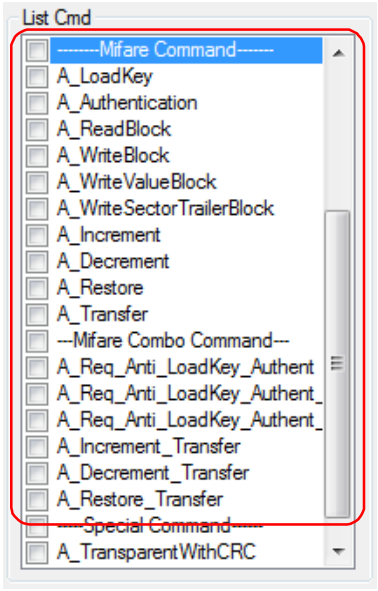


Figure 48 Available commands related to MIFARE in Pi-931

Relation between Pi-931 commands in MIFARE state transition diagram are shown in Figure 49 and Figure 50.

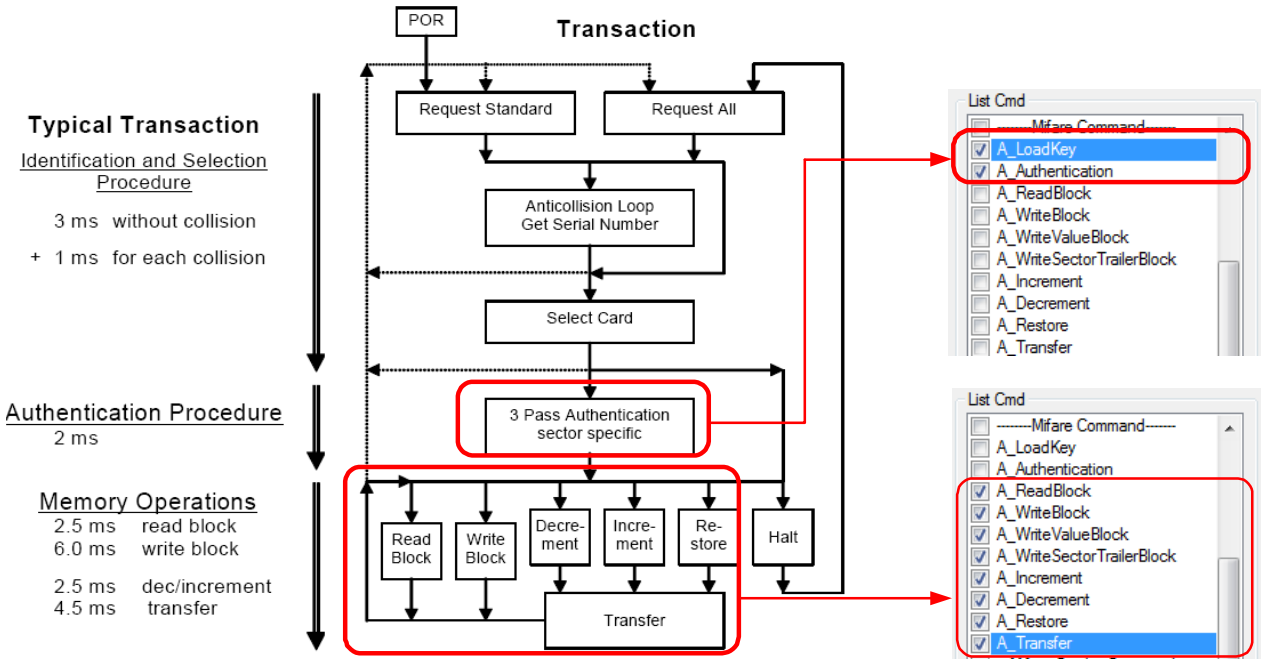


Figure 49 MIFARE Standard commands

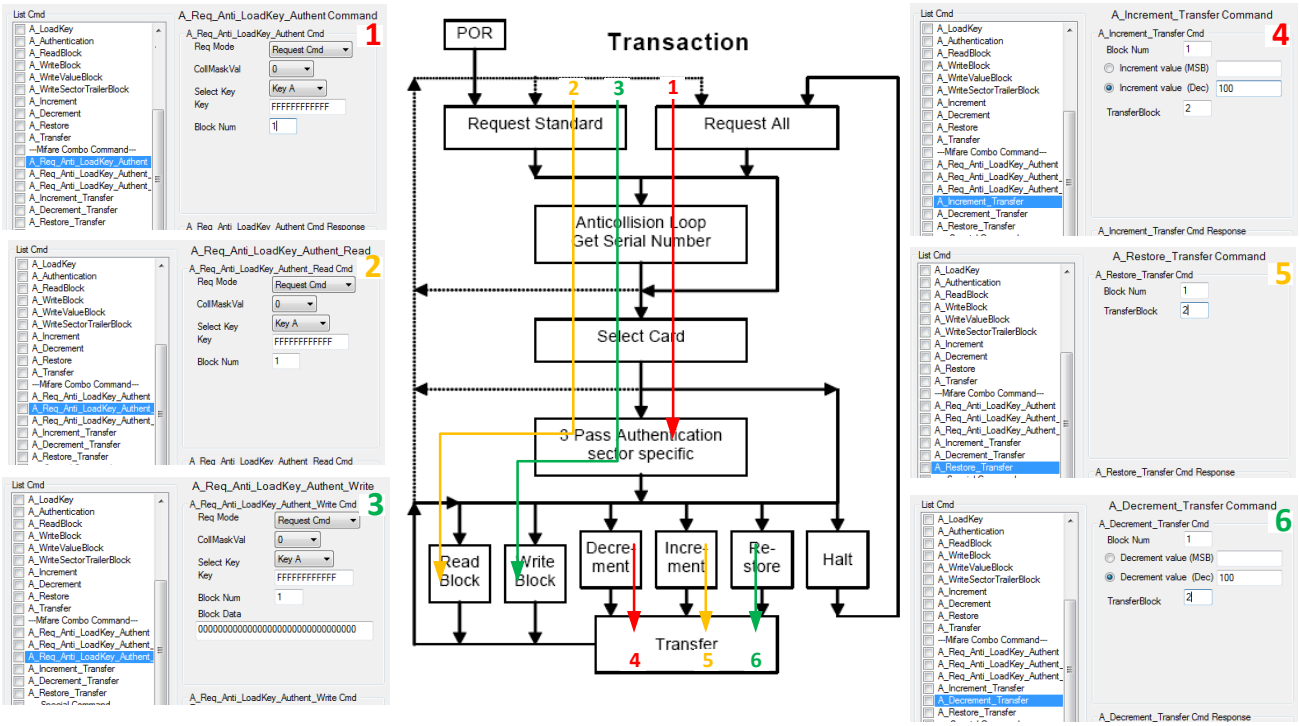


Figure 50 MIFARE Combo commands

6.1.3.1 A_LoadKey

The **A_LoadKey** command GUI is shown Figure 51. This command requires a 48-bit hexadecimal key matched the key in memory sector on card being authenticated. The supplied key is loaded into key buffer in reader IC to be used authentication.

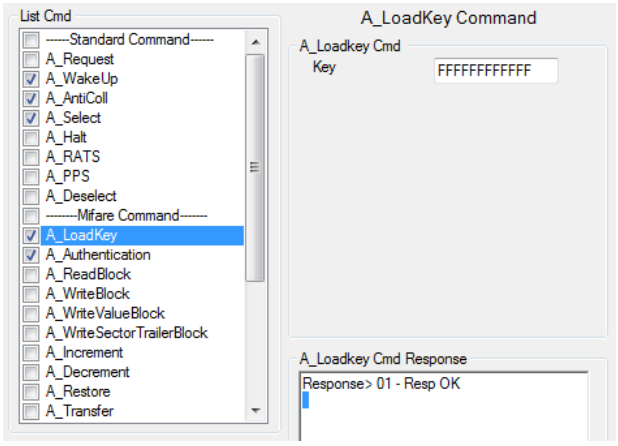


Figure 51 **A_LoadKey** commands

6.1.3.2 A_Authentication

The **A_Authentication** command GUI is shown Figure 52. Parameter as follows shall be specified.

Select Key : Key A or Key B on card used in authentication

Block Num : The block address to be accessed

UID : Unique ID of card to be accessed

This operation shall be performed after executing **A_LoadKey**. For typical 1k-byte MIFARE card, every four blocks is governed by keys (A and B) of each sector. For example, block 4, 5, 6 and 7 rely on the same key stored in block. If block 4 was authenticated, block 5, 6 and 7 can also be accessed with re-authentication. If authentication is successful, reader will return successful code “0x01”as shown in Figure 52.

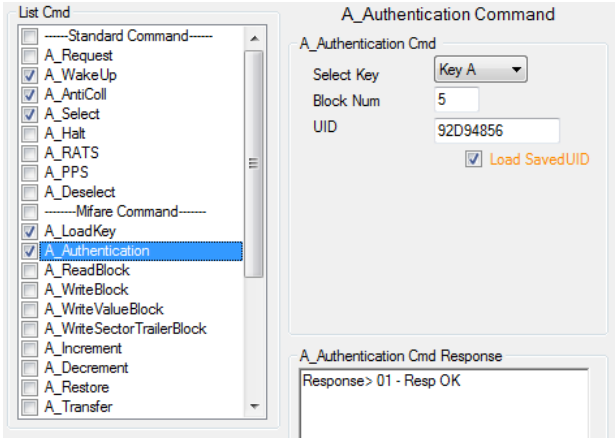
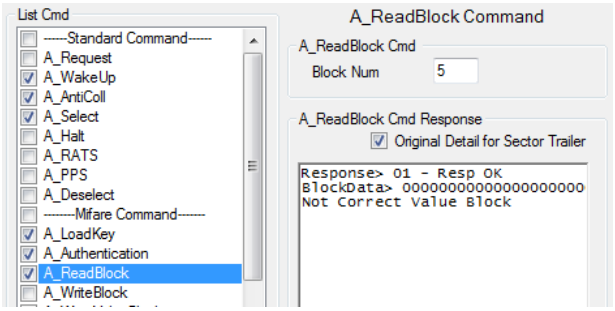


Figure 52 **A_Authentication** commands

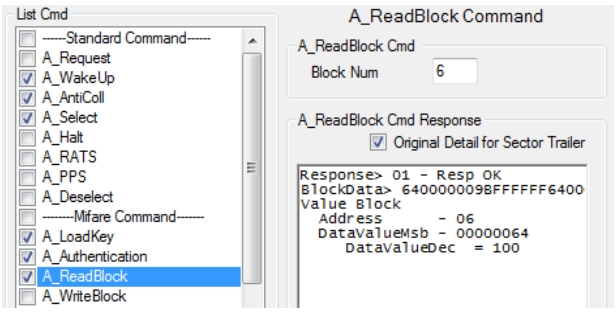
6.1.3.3 A_ReadBlock

The **A_ReadBlock** command GUI, shown Figure 53, is to read data from target block address. Therefore, the target block address to be read is a required parameter. User should read data with in authenticated sector. Else, the card will response error and accessing process must be restart from **A_Request** command. This operation shall be performed after successful authentication.

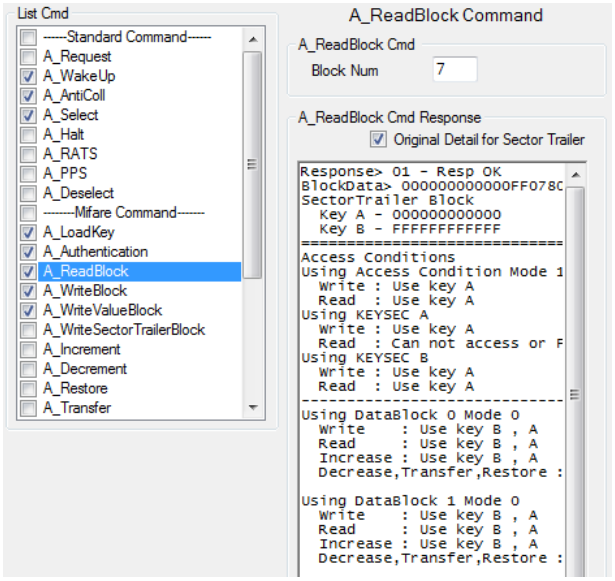
Software automatically reports characteristic of received data whether it is data block, value block or Sector Trailer Block. Reading result from plain data block, value block and sector trailer block are shown in Figure 53A, Figure 53B and Figure 53C respectively.



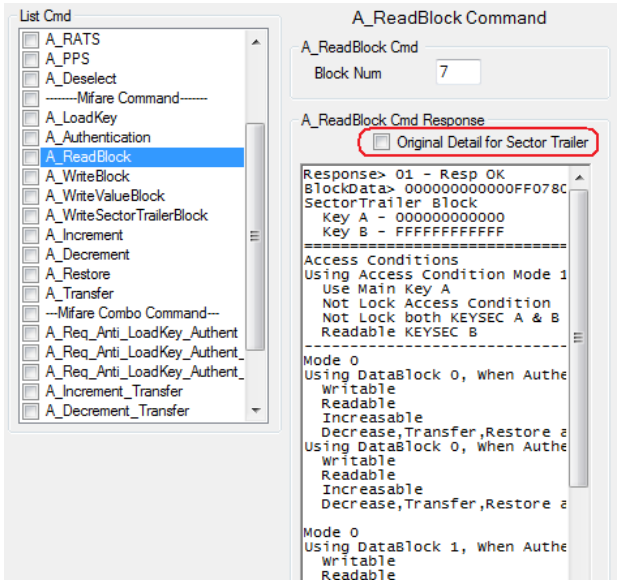
A) Data Block



B) Value Block



C) Sector Trailer block



D) Sector Trailer Block with control block unchecked

Figure 53 **A_ReadBlock** commands

6.1.3.4 A_WriteBlock

The **A_WriteBlock** command GUI, shown Figure 54, is to write data to target block address. Parameters as follows shall be specified.

Block Num : The target block address to be written

Block Data : 16-byte data coded in hexadecimal format

This operation shall be performed after successful authentication.

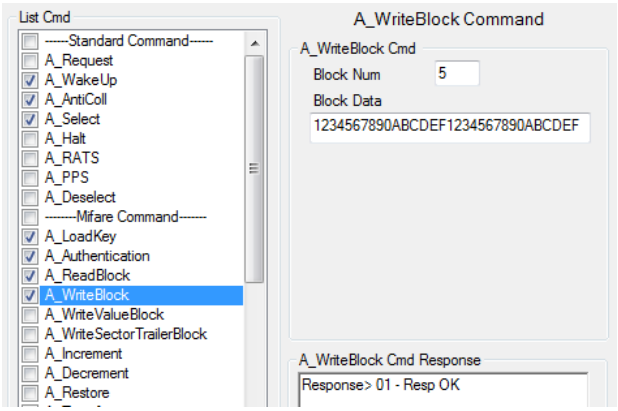


Figure 54 **A_WriteBlock** commands

6.1.3.5 A_WriteValueBlock

The **A_WriteValueBlock** command GUI, shown Figure 55, is to write value to target block address. Parameters as follows shall be specified.

Address value : a 1-byte data storing in value block for indicating address of back up block

Value : a 4-byte data coded in 2's complement format

In this demonstration software, user can either input amount of value in hexadecimal or decimal number. To input value in hexadecimal, use box "Data Value (MSB)". To input value in decimal, input data in box "Data Value (Dec)". For hexadecimal, value of 0x00000000 to 0x7FFFFFFF is positive number from 0 to 2,147,483,647 while value of 0x80000000 to 0xFFFFFFFF is negative number from -2,147,483,648 to -1. For decimal number, data shall be within -2,147,483,648 to 2,147,483,647.

Block Num : The target block address to be written.

This operation shall be performed after successful authentication.

Note that value and address value is redundantly stored in 16-byte blocks as depicted in Figure 56.

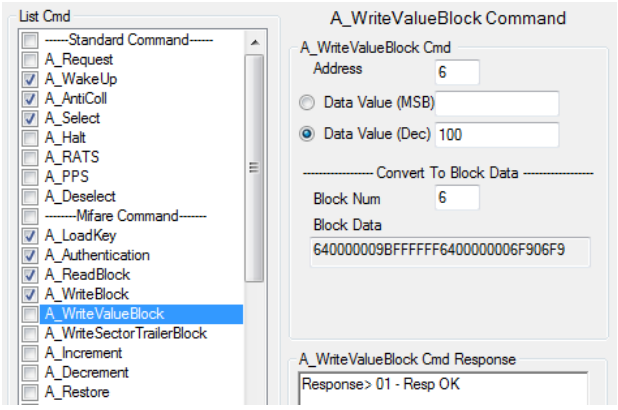


Figure 55 **A_WriteValueBlock** commands

Value Blocks																
Byte #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	**Value**				**_Value**				**Value**				**Adr**	**_Adr**	**Adr**	**_Adr**

Note **_Value** is inversion of **Value** Note **_Adr** is inversion of **Adr**

Figure 56 Structure of value block

6.1.3.6 A_WriteSectorTrailerBlock

The sector trailer located in last block of each sector contains secret key A and key B, which return logical "0" when read, and the access condition for the four blocks of each sector. The structure of the sector trailer block and access bit organization are shown in Figure 57. The accessibility of each block is controlled by three control bits namely C1x, C2x and C3x. The access condition is redundantly stored with its inversion to secure data protection. There are different meanings about accessibility for the sector trailer block and data blocks as shown in Figure 58 and Figure 62. Note that byte 9 in the sector trailer block is unused area.

Sector Trailer Blocks																
Byte #	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Key A						**Access Bits**				**Key B**					

Access Bits								
Bit #	7	6	5	4	3	2	1	0
Byte 6	**_C23**	**_C22**	**_C21**	**_C20**	**_C13**	**_C12**	**_C11**	**_C10**
Byte 7	**C13**	**C12**	**C11**	**C10**	**_C33**	**_C32**	**_C31**	**_C30**
Byte 8	**C33**	**C32**	**C31**	**C30**	**C23**	**C22**	**C21**	**C20**
Byte 9								

Note **_Cxx** is inversion of **Cxx**

	Access bit			Description
	C1x	**C2x**	**C3x**	
Block0	**C10**	**C20**	**C30**	Data block 0 in the sector
Block1	**C11**	**C21**	**C31**	Data block 1 in the sector
Block2	**C12**	**C22**	**C32**	Data block 2 in the sector
Block3	**C13**	**C23**	**C33**	Data block 3 in the sector

Figure 57 Structure of the sector trailer block

Access bits			Access condition for				Application
C1	C2	C3	read	write	increment	decrement, transfer, restore	
0	0	0	key A|B[1]	key A|B1	key A|B1	key A|B1	transport configuration
0	1	0	key A|B[1]	never	never	never	read/write block
1	0	0	key A|B[1]	key B[1]	never	never	read/write block
1	1	0	key A|B[1]	key B[1]	key B[1]	key A|B[1]	value block
0	0	1	key A|B[1]	never	never	key A|B[1]	value block
0	1	1	key B[1]	key B[1]	never	never	read/write block
1	0	1	key B[1]	never	never	never	read/write block
1	1	1	never	never	never	never	read/write block
Access bits			Access condition for				Application

[1] if Key B may be read in the corresponding Sector Trailer it cannot serve for authentication (all grey marked lines in previous table). Consequences: If the reader tries to authenticate any block of a sector with key B using grey marked access conditions, the card will refuse any subsequent memory access after authentication.

Figure 58 Access conditions for data blocks

Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read
0	1	0	never	never	key A	never	key A	never	Key B may be read
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

Remark: the grey marked lines are access conditions where key B is readable and may be used for data.

Figure 59 Access conditions for the sector trailer

The GUI for **A_WriteSectorTrailerBlock** is shown in Figure 60. Parameters as follows shall are required.

Key A : A 6-bytes secret key

Key B : A 6-bytes secret key

Access Condition : The access condition for Sector Trailer, Block0, Block1 and Block2 the number of 0 to 7 is selectable which a decimal result of bit “C1x-C2x-C3x”

Block Num : Target block to be written

To ease beginner to understand the meaning of each access condition mode, the demonstration software summarizes accessibility of read and write of each block. Also, the check block “Original” can be optionally unchecked to display compatibility of each command.

Before write the sector trailer, user must be aware the effect of written configuration because some configuration may be irreversible. Hence, the card is permanently locked in such mode.

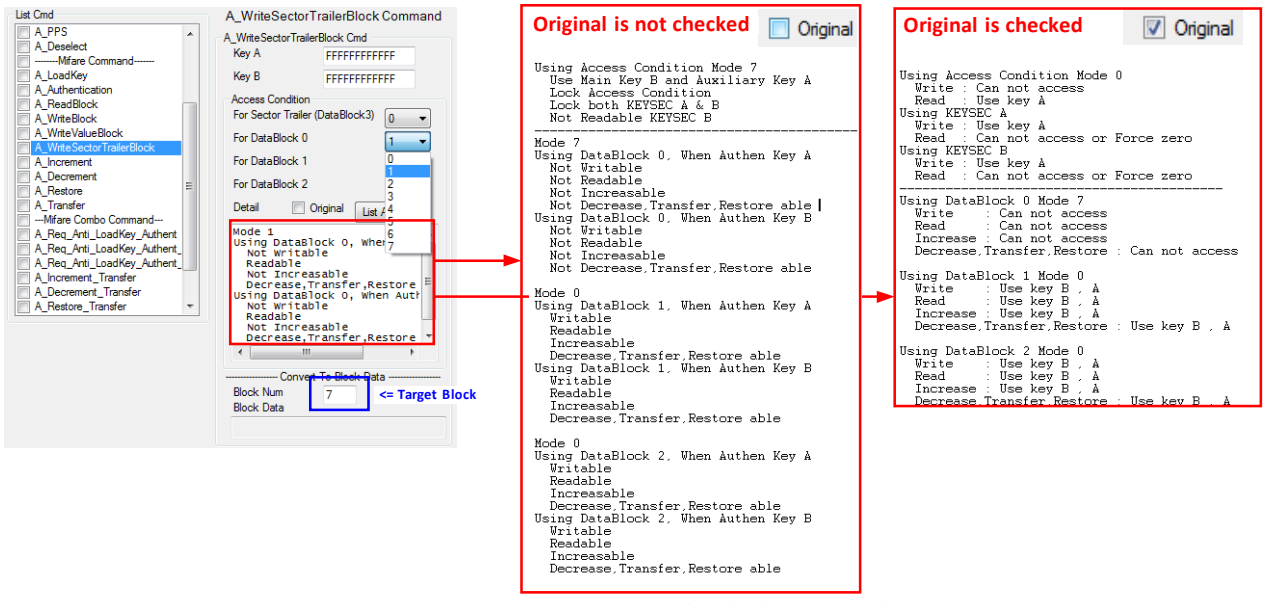
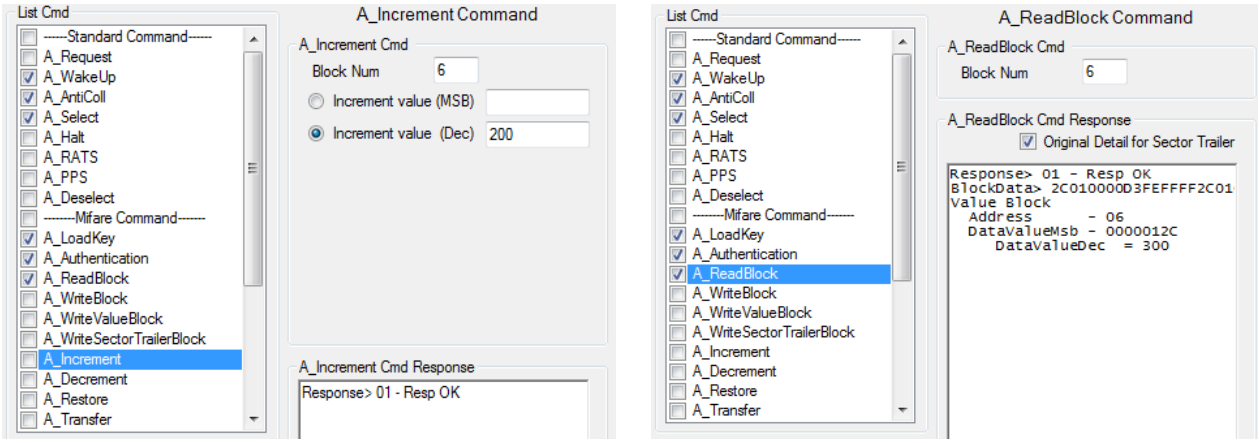


Figure 60 **A_WriteSectorTrailerBlock** commands

Note that the demonstration software formats sector trailer from data in GUI and sent to reader by normal write command.

6.1.3.7 A_Increment

The **A_Increment** command GUI is shown Figure 61A. Incremental value shall be specified either in hexadecimal or decimal value. Figure 61B shows example of incremental result of 200 from 100 as in Figure 55. Increment which will result in amount of final value beyond 0x7FFFFFFF (2,147,483,647) is inhibited and cause error in operation. Note that result from increment is stored in card buffer. User must store to target address by using command **A_Transfer**. This operation shall be performed after successful authentication.

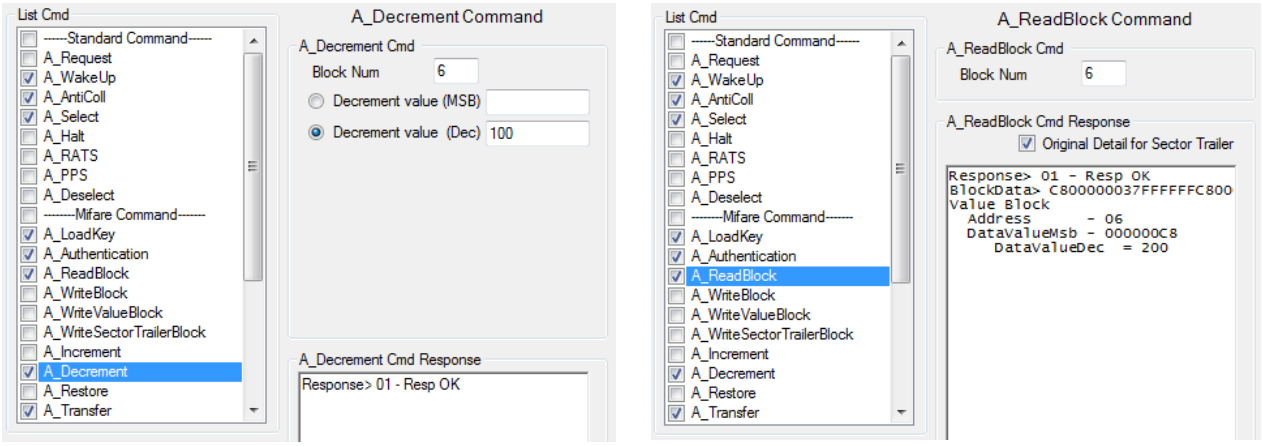


A) **A_Increment** command GUI B) Incremental result

Figure 61 **A_Increment** commands

6.1.3.8 A Decrement

The **A Decrement** command GUI is shown Figure 62A. Decreasing value shall be specified either in hexadecimal or decimal value. Figure 62B shows example of decrement of 100 from 300 as in Figure 61. Decrement which will result in amount of final value below 0x80000000 (-2,147,483,648) is inhibited and cause error in operation. Note that result from increment is stored in card buffer. User must store to target address by using command **A_Transfer**. This operation shall be performed after successful authentication.



A) **A Decrement** command GUI B) Decrement result

Figure 62 **A Decrement** commands

6.1.3.9 A_Restore

The **A_Restore** command GUI is shown Figure 63. Parameter for this command is block address in which content will be loaded into card buffer. This operation shall be performed after successful authentication.

6.1.3.10 A_Transfer

The **A_Transfer** command GUI is shown Figure 64. Parameter for this command is target block address where active content in card buffer will be written. Once content in card buffer is used, the content can not used again. Transferring inactive content causes error response. Figure 64 shows data transfer to block 5. This operation shall be performed after successful authentication.

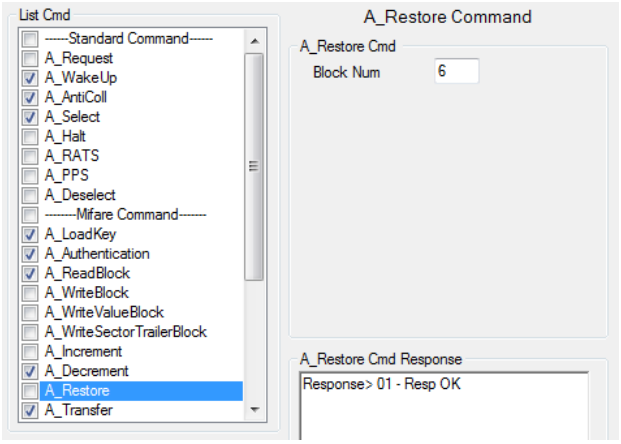
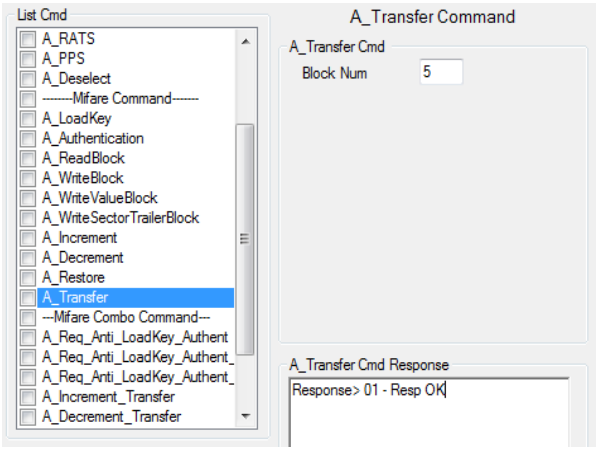
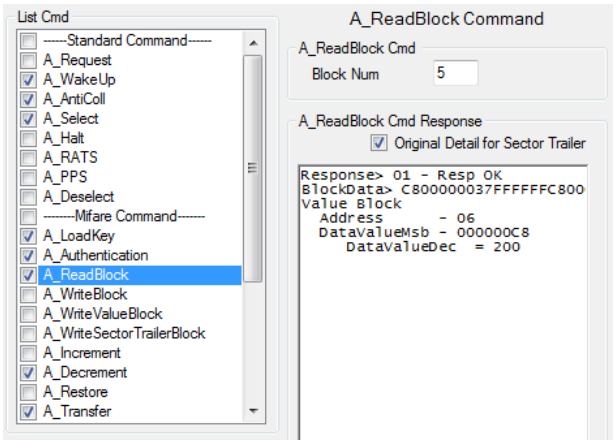


Figure 63 **A_Restore** commands



A) **A_Transfer** command GUI



B) Transfer result

Figure 64 **A_Transfer** commands

6.1.3.11 A_Req_Anti_LoadKey_Authent

The **A_Req_Anti_LoadKey_Authent** is a combo command performing ISO14443A from card power-on to authentication as listed below.

1. **A_Request** or **A_Wake_up**
2. **A_Anticoll**
3. **A_Select**
4. **A_Loadkey**
5. **A_Authent**

This command requires parameters as follows before operating.

Req Mode : Select Request or Wakeup command used to probe card in field

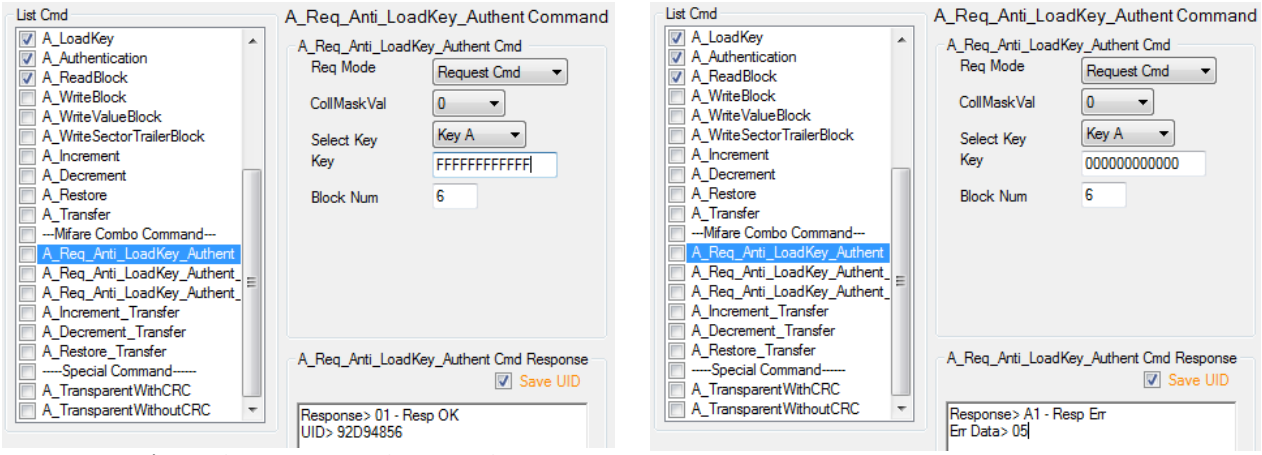
CollMaskVal : Select CollMaskVal to select card as explain in **A_Anticoll** command

Select Key : Select Key A or Key B on card used in authentication

Key : 48-bit key

Block Num : The block address to be accessed

If operation is successful, reader will return UID and card will be ready to use MIFARE commands such as write, read, restore and transfer. Figure 65A shows successful authentication while Figure 65B shows error from authentication. Notice that error code in Figure 65B is “05”. This reflects to state that error occurs. The error state is authentication.



A) Authentication with correct key B) Authentication with wrong key

Figure 65 **A_Req_Anti_LoadKey_Authent** commands

6.1.3.12 A_Req_Anti_LoadKey_Authent_Read

The **A_Req_Anti_LoadKey_Authent_Read** is a combo command performing ISO14443A command from card power-on to read block as listed below.

1. **A_Request** or **A_Wake_up**
2. **A_Anticoll**
3. **A_Select**
4. **A_Loadkey**
5. **A_Authent**
6. **A_ReadBlock**

This command requires parameters as follows before operating.

Req Mode : Select Request or Wakeup command used to probe card in field

CollMaskVal : Select CollMaskVal to select card as explain in **A_Anticoll** command

Select Key : Select Key A or Key B on card used in authentication

Key : 48-bit key

Block Num : The block address to be read

Figure 66 shows reading result from block 6 from this combo command.

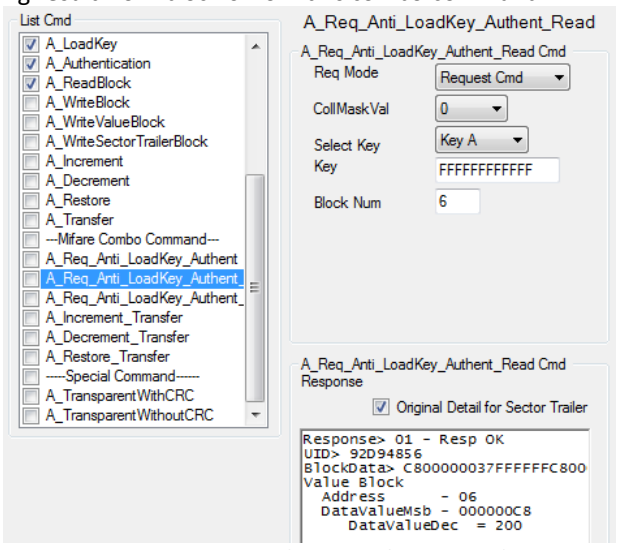


Figure 66 **A_Req_Anti_LoadKey_Authent_Read** commands

6.1.3.13 A_Req_Anti_LoadKey_Authent_Write

The **A_Req_Anti_LoadKey_Authent_Write** is a combo command performing ISO14443A command from card power-on to write block as listed below.

1. **A_Request** or **A_Wake_up**
2. **A_Anticoll**
3. **A_Select**
4. **A_Loadkey**
5. **A_Authent**
6. **A_WriteBlock**

This command requires parameters as follows before operating.

Req Mode : Select Request or Wakeup command used to probe card in field

CollMaskVal : Select CollMaskVal to select card as explain in **A_Anticoll** command

Select Key : Select Key A or Key B on card used in authentication

Key : 48-bit key

Block Num : The block address to be written

Figure 67 shows writing result into block 6 from this combo command.

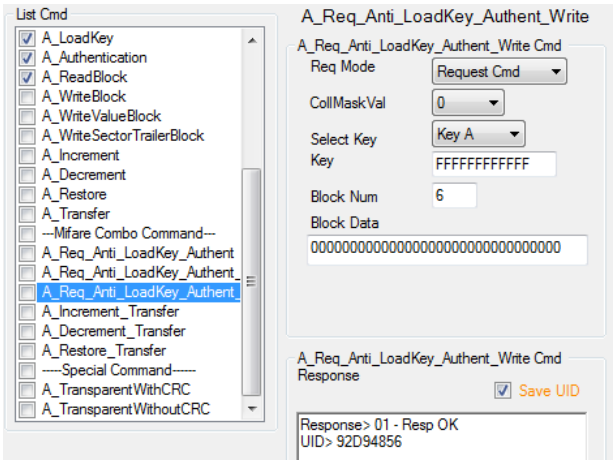


Figure 67 **A_Req_Anti_LoadKey_Authent_Write** commands

6.1.3.14 A_Increment_Transfer

The **A_Increment_Transfer** is a combo command performing MIFARE command as listed below.

1. **A_Increment**
2. **A_Transfer**

This command requires parameters as follows before operating.

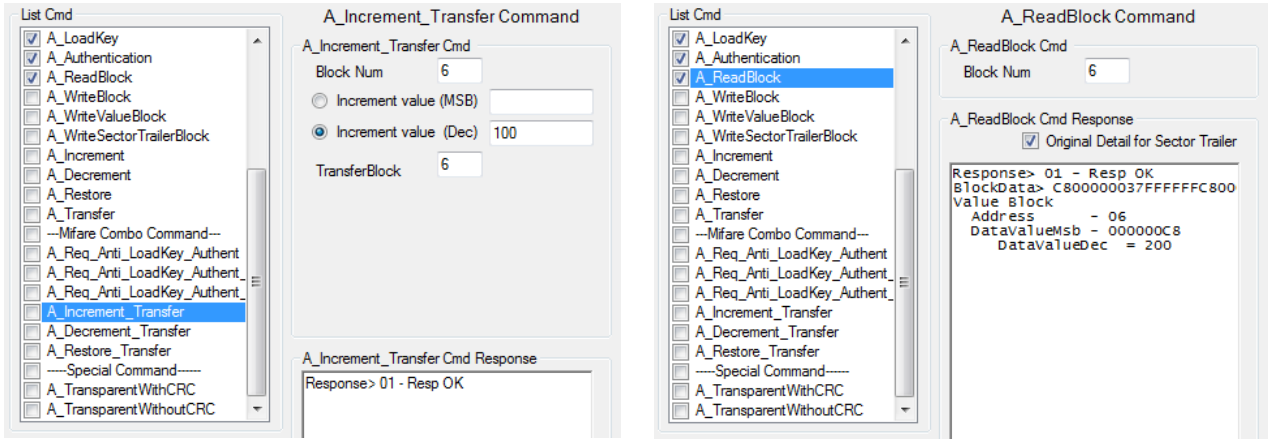
Address value : A 1-byte data storing in value block for indicating address of back up block

Block Num : The block address to perform increment

Value : A 4-byte data coded in 2's complement format or decimal number between -2,147,483,648 and 2,147,483,647.

Transfer Block : Target block address where increasing value will be written

This operation shall be performed after successful authentication. Figure 68 shows incremental result in block 6 from 100 by 100. Note that increment which will result in amount of final value beyond 0x7FFFFFFF (2,147,483,647) is inhibited and cause error in operation.



A) Input in **A_Increment_Transfer** B) Incremental result

Figure 68 **A_Increment_Transfer** commands

6.1.3.15 A_Decrement_Transfer

The A_Decrement_Transfer is a combo command performing MIFARE command as listed below.

1. **A_Increment**
2. **A_Transfer**

This command requires parameters as follows before operating.

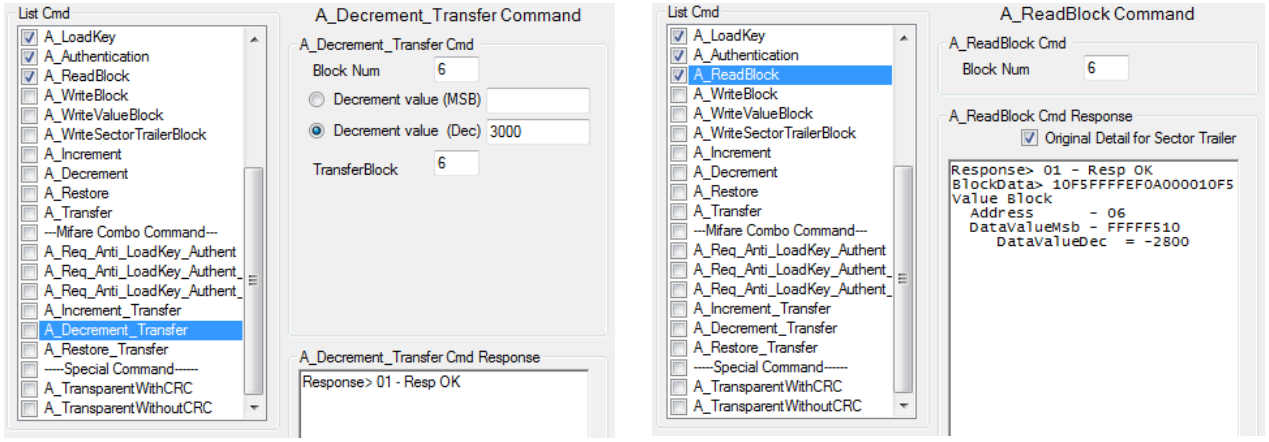
Address value : A 1-byte data storing in value block for indicating address of back up block

Block Num : The block address to perform decrement

Value : A 4-byte data coded in 2's complement format or decimal number between -2,147,483,648 and 2,147,483,647.

Transfer Block : Target block address where decreasing value will be written

This operation shall be performed after successful authentication. Figure 69 shows decrement result in block 6 from 200 by 3000. Note that decrement which will result in amount of final value below 0x80000000 (-2,147,483,648) is inhibited and cause error in operation.



A) Input in **A_Decrement_Transfer** B) Decrement result

Figure 69 **A_Decrement_Transfer** commands

6.1.3.16 A_Restore_Transfer

The A_Decrement_Transfer is a combo command performing MIFARE command as listed below.

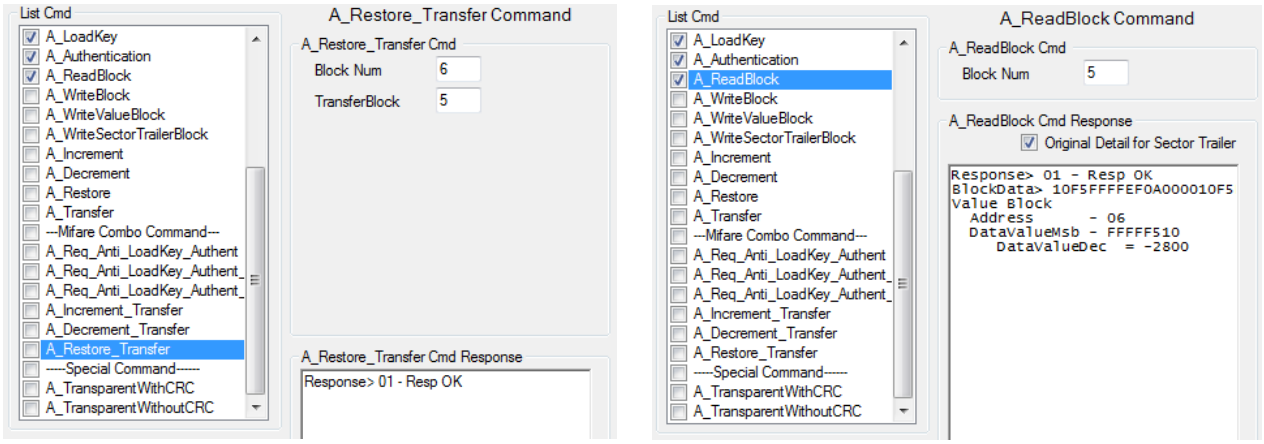
1. **A_Restore**
2. **A_Transfer**

This command requires parameters as follows before operating.

Block Num : The block address in which content will be loaded into card buffer

Transfer Block : Target block address where active content in card buffer will be written

This operation shall be performed after successful authentication. Figure 70 shows decrement result in transferring data from block 6 to block 5.



A) Input in **A_Restore_Transfer** B) Transfer result

Figure 70 **A_Restore_Transfer** commands

6.1.3.17 A_Req_Anti_Select

The A_Req_Anti_Select is a combo command used in getting UID and selecting ISO14443A card from card power-on to authentication as listed below.

1. **A_Request** or **A_Wake_up**
2. **A_Anticoll**
3. **A_Select**

This command requires parameters as follows before operating.

Req Mode : Select Request or Wakeup command used to probe card in field

CollMaskVal : Select CollMaskVal to select card as explain in **A_Anticoll** command

Figure 71 **A_Restore_Transfer** commands

6.2 ISO14443B

The ISO14443B commands list of Pi-931 is shown Figure 72.

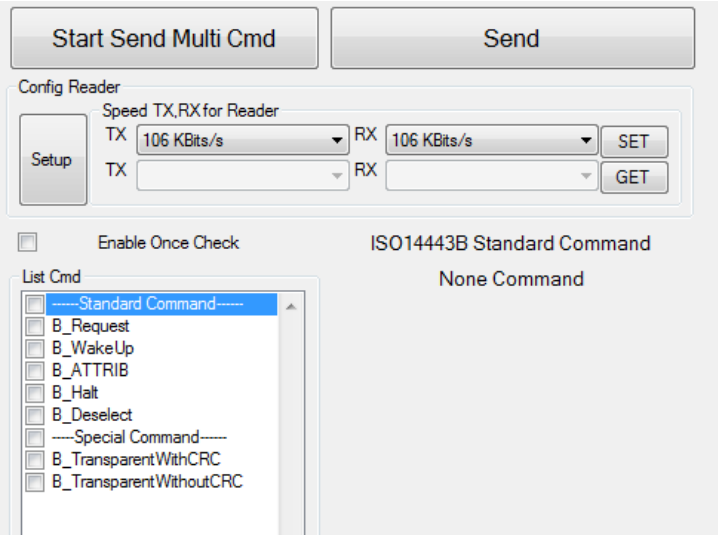


Figure 72 ISO14443B command list

6.2.1 ISO14443B standard commands

Figure 73 depicts the relation of the commands with card state transition diagram. Each command shall be applied at an appropriate state. The commands in this section are **B_Request**, **B_WakeUp**, **B_ATTRIB**, **B_Halt** and **B_Deselect, B_TransparentWithCRC** and **B_TransparentWithoutCRC**. For more information about the commands, please refer to ISO14443-3 standards and Pi-931 Protocol.

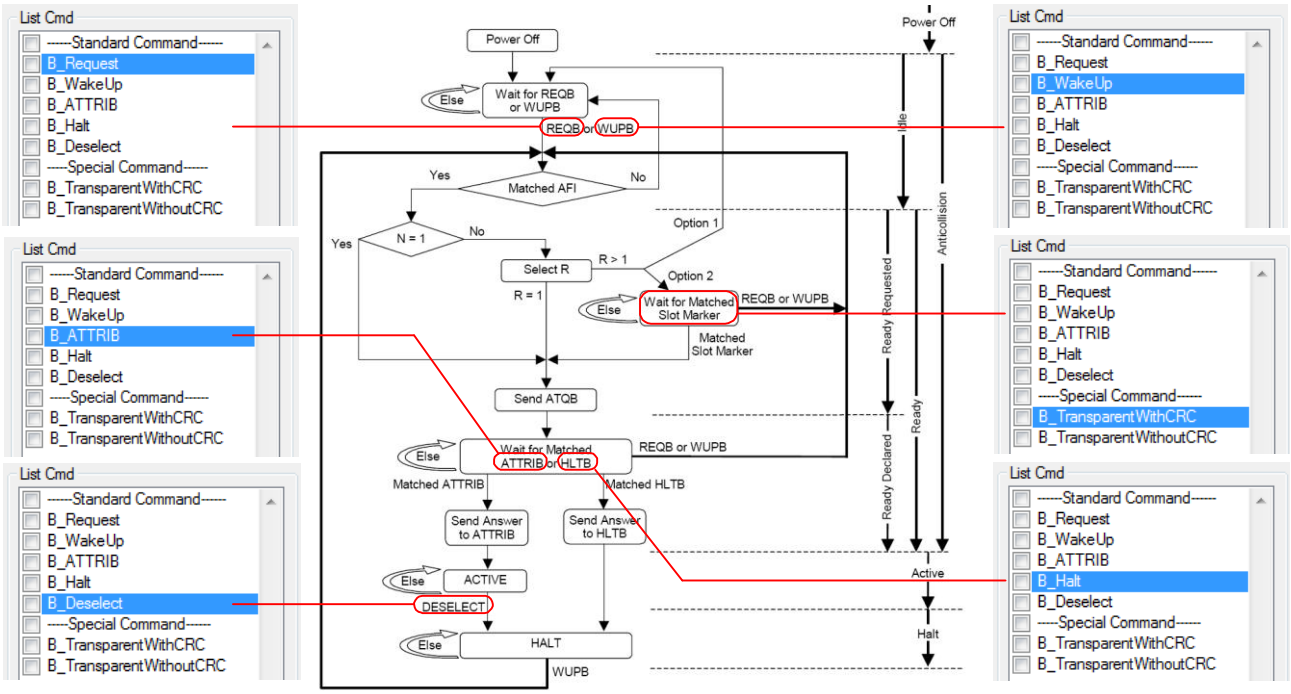


Figure 73 ISO14443B standard commands available in Pi-931

6.2.1.1 B_Request

The GUI of the **B_Request** is shown Figure 74. The AFI value and the number of slot are required parameters. The **B_Request** command of Pi-931 not only transmits the REQB command following ISO14443B but also performs anti-collision if the number of slot more than 1 is specify. The first card of which PUPI is completely received in anti-collision slot is reported. Hence, the slot Marker is already embedded; that correspond command

for Slot Marker is not provided. However, user can transmits by the slot market via transparent command. The responses from this command, called ATQB, are PUPI, application data and protocol information. The demonstration software decomposes the responses to show card properties. A Pseudo-Unique PICC Identifier (PUPI) is used to differentiate cards during anti-collision. User can check "Save PUPI" check block to collect in software buffer to use in further operation. The Application data field is used to inform the PCD which applications are currently installed in the PICC. The Protocol Info field indicates the parameters supported by the card. For more information, please refer to ISO14443-3.

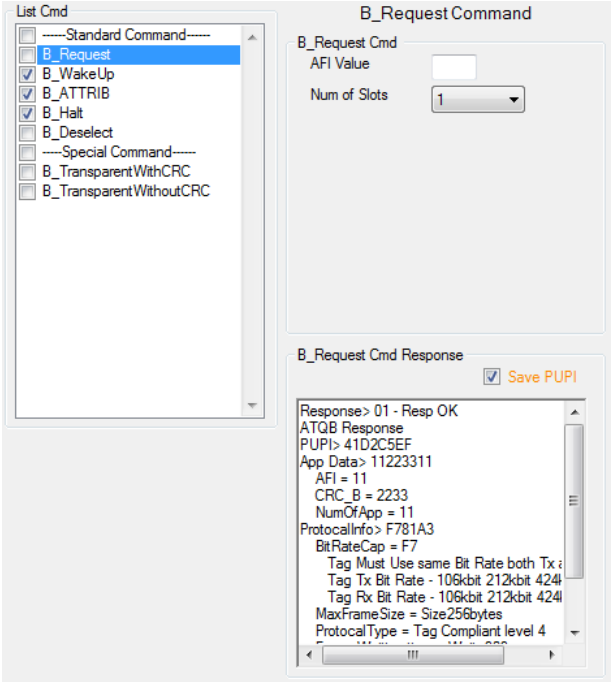


Figure 74 **B_Request** Command

6.2.1.2 B_WakeUp

The operation of **B_WakeUp** is similar to **B_Request**. Different from **B_Request**, **B_WakeUp** can also wake up card in Halt state. The **B_WakeUp** GUI is shown in Figure 75.

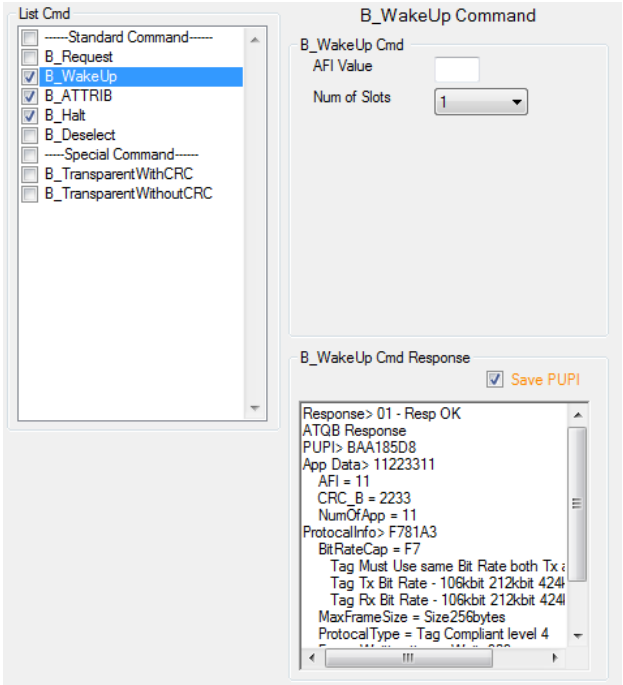


Figure 75 **B_WakeUp** Command

6.2.1.3 B_ATTRIB

This command is used to select operating protocol and parameter to use in further operation of smartcard. This command requires Param1, Param2, Param3 and Param4 as stated in ISO14443B. This command shall be activated next to **B_Request** or **B_WakeUp** command. After executing this command, user shall set the transaction speed match to what card response. For more information about meaning of the parameters, please refer to ISO14443-3 and -4. The GUI of **B_ATTRIB** as shown in Figure 76 aids user in composing Param1, Param2, Param3 and Param4.

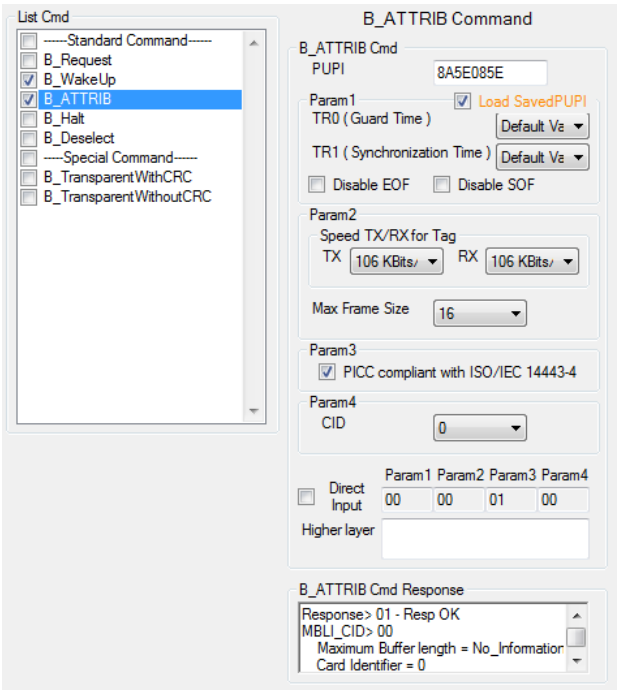


Figure 76 **B_ATTRIB** Command

6.2.1.4 B_Halt

The **B_Halt** command is used to put card in Halt state. The GUI of **B_Halt** is shown in Figure 77. The PUPI is a required parameter. This command shall be applied after receiving ATQB from **B_Request** or **B_wakeup**.

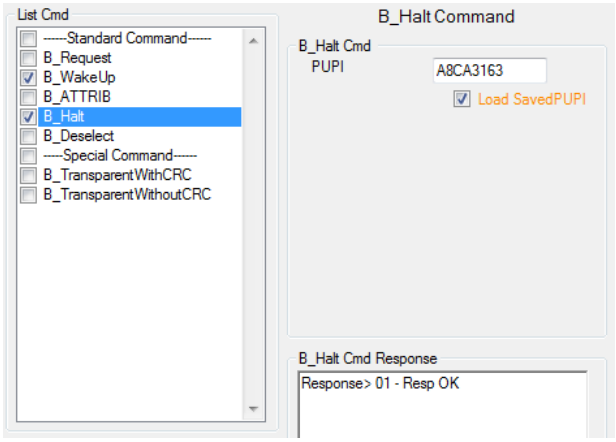


Figure 77 **B_Halt** Command

6.2.1.5 B_Deselect

The **B_Deselect** is used to exit active state. This command requires no input and, actually, this command is made up of **B_TransparentWithCRC.** The GUI of **B_Deselect** is shown in Figure 77

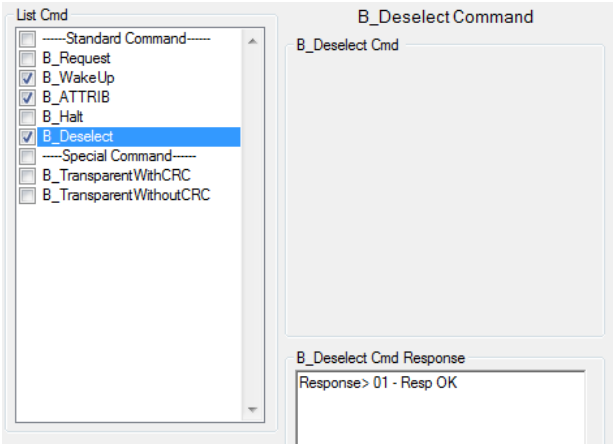


Figure 78 **B_Deselect** Command

6.2.1.6 B_TransparentWithCRC

If commands to be transmitted are not provided in command list such as command for smart card or slot marker, user can input raw hexadecimal code in **B_TransparentWithCRC** to invoke operation as stated in the RF protocol. Before executing, user might need to adjust timeout for card response to be suitable for user applications. The timeout setting option in the GUI is shown in Figure 80.

The example of "initiate" command in SRI4K card is shown in Figure 79. User shall review results in transaction logs windows. For example, subsequent operations from Figure 79 to read data are shown in Figure 81. For more information about SRI4K, please refer to the datasheet from ST.

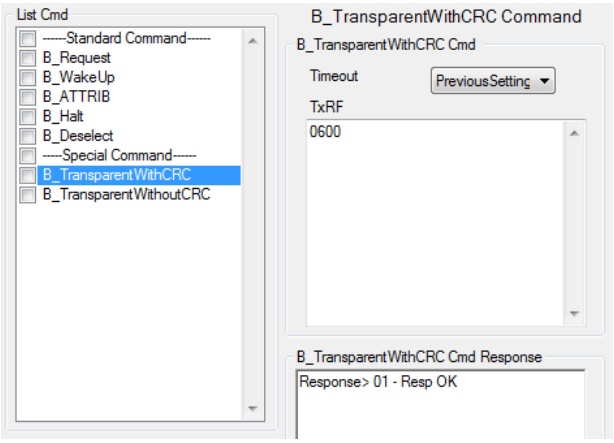


Figure 79 Example of using **B_ TransparentWithCRC** with SRI4K card

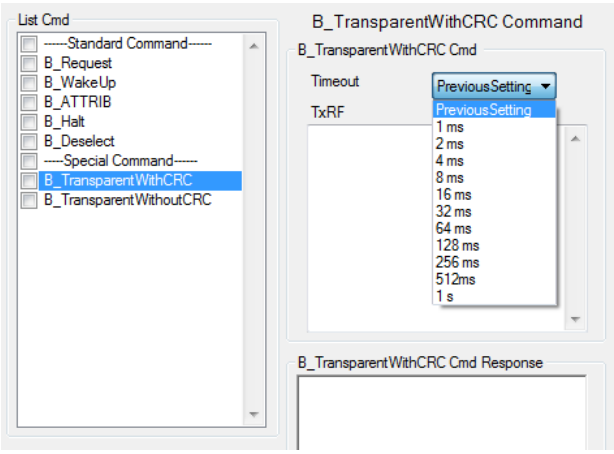


Figure 80 Timeout setting for TransparentWithCRC and TransparentWithoutCRC command

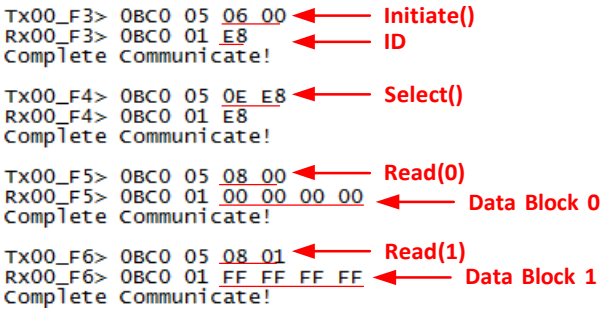


Figure 81 Example of running multiple ISO14443B command

6.2.1.7 B_TransparentWithoutCRC

The operation of this command is similar **to B_ TransparentWithCRC**. The different is there is no calculated CRC appending at the end of transmitted RF frame.

6.3 ISO15693

The GUI in ISO15693 tap is shown in Figure 82 in which user can select RF transaction speed and operating command. The command list consists of ISO15693 standard commands, Custom command for SIC RFID card IC and special command as is shown in Figure 83. The RF transaction speeds for downlink and uplink are summarized in Table 6-1.

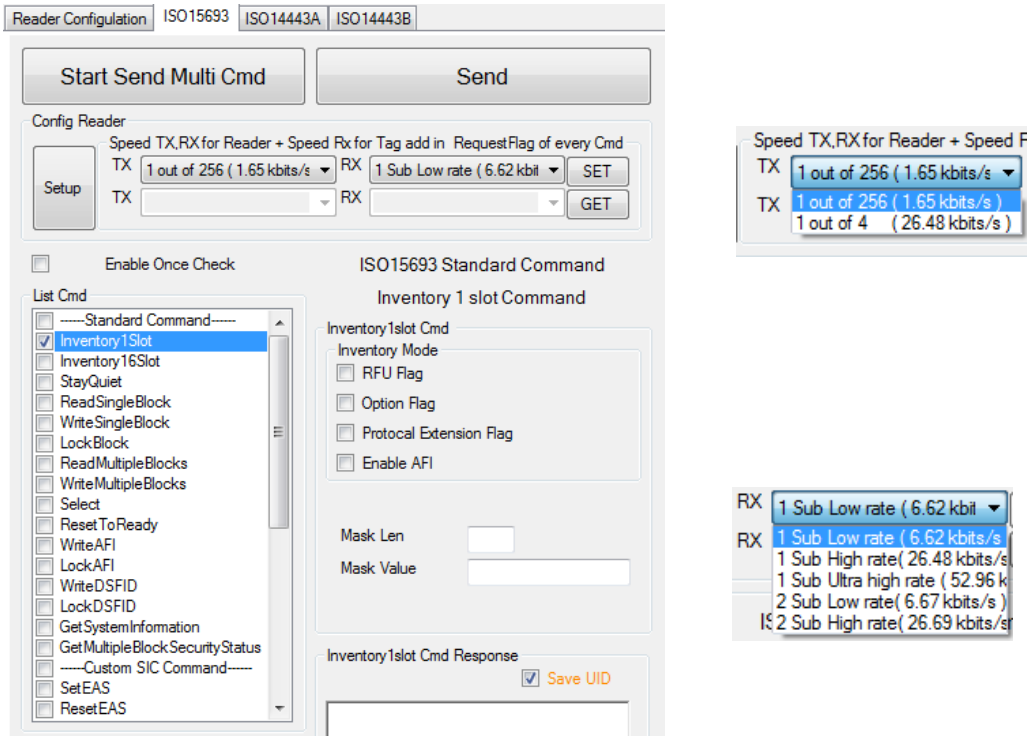


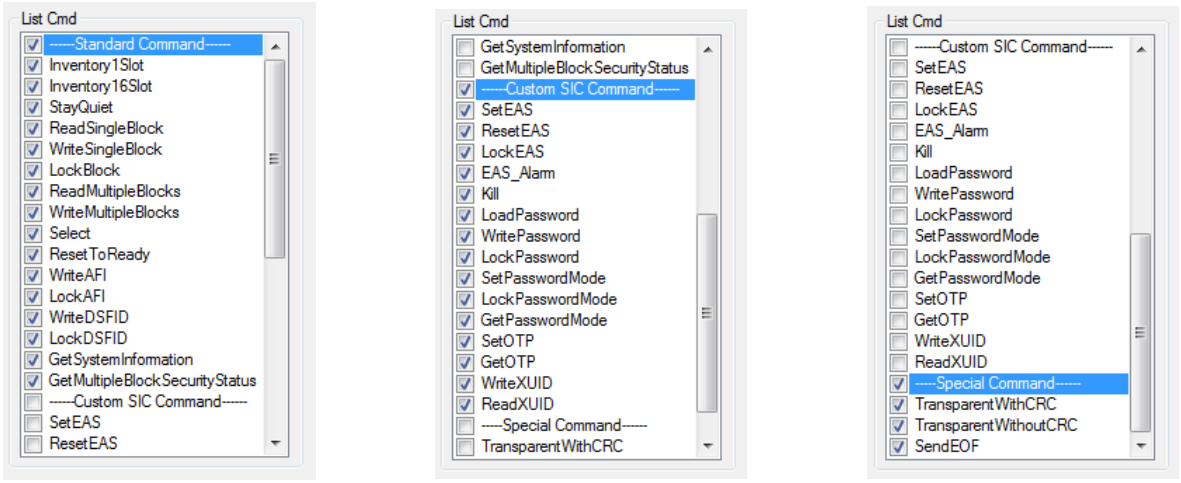
Figure 82 GUI in ISO15693 Tap

Table 6-1 Supported RF transmission speed

Transaction	Supported RF Coding	RF Speed	Description
Downlink	1 out of 4	26.48 Kbits/s	1 out of 4 coding
	1 out of 256	1.65 Kbits/s	1 out of 256 coding
Uplink	1 Sub Low rate	6.62 Kbits/s	One subcarrier at Low data rate
	1 Sub High rate	26.48 Kbits/s	One subcarrier at High data rate
	1 Sub Ultrahigh rate	52.96 Kbits/s	One subcarrier at Ultrahigh data rate
	2 Sub Low Rate	6.67 Kbits/s	Two subcarrier at Low data rate
	2 Sub High Rate	26.69 Kbits/s	Two subcarrier at High data rate

The ISO15693 command in Pi-931 requires operating mode supplied in command packet before executing. Command mode in operation can be primarily divided into two groups namely inventory mode and non-inventory mode. Inventory mode is for accessing the UID while non-inventory is for any normal card operations. The non-inventory mode can be subdivided into three modes namely non-address mode, address mode and select mode.

Figure 84 illustrates command modes and related operable command. Similar to other RFID standards, card itself do occupy a specific state at certain time. Transition state diagram in ISO15693 is shown in Figure 85. In each state, card reacts to transmitted command differently. Table 6-2 summarizes operable mode between command mode and card state.



Standard Command Custom SIC Command Special Command

Figure 83 ISO15693 standard command

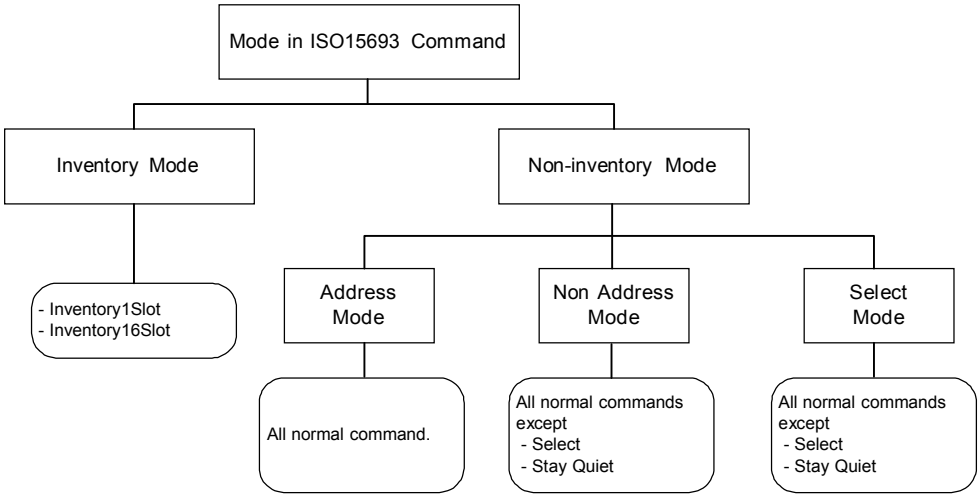


Figure 84 command mode in ISO15693

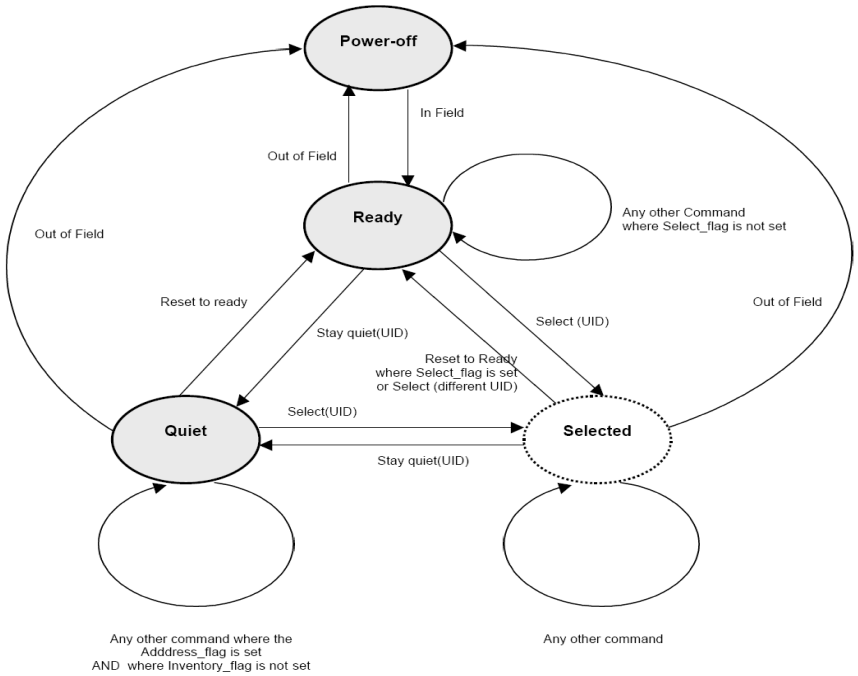


Figure 85 State transition diagram of the ISO15693 card

Table 6-2 operable mode between command mode and card state				
Command Mode / Card State	Inventory Mode	Non-Inventory Mode		
		Address Mode (with UID)	Non-Address Mode (without UID)	Selected Mode
Ready State	Yes	Yes	Yes	No
Selected State	Yes	Yes	Yes	Yes
Quiet State	No	Yes	No	No

In inventory mode, there are four control flags as shown in Figure 86. The RFU Flag, Option Flag and Protocol Extension Flag can be optionally checked to perform special functions. The functions of these flags are depended on purpose of card IC manufacturer. For Enable AFI flag, if this flag is checked, AFI value must be supplied. The benefit of the AFI (Application family identifier) is to pre-discriminate type of card and reduces the number of card in anti-collision process.

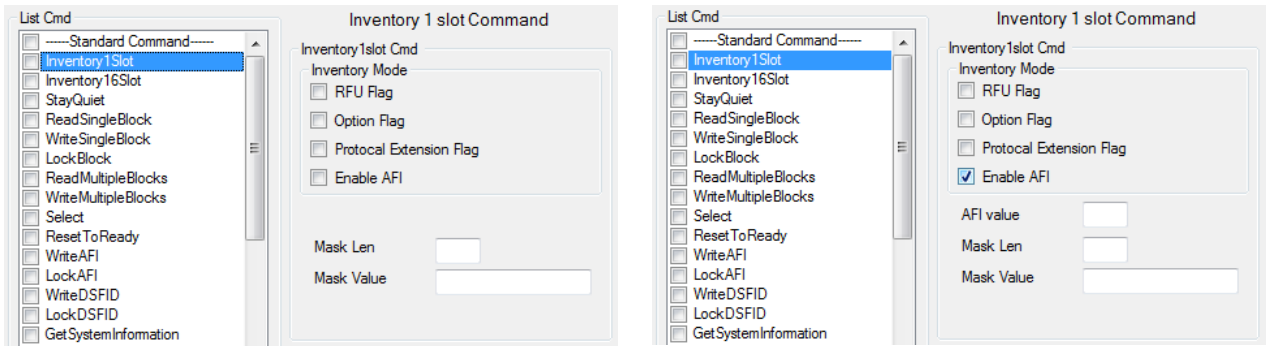


Figure 86 Flag in inventory mode

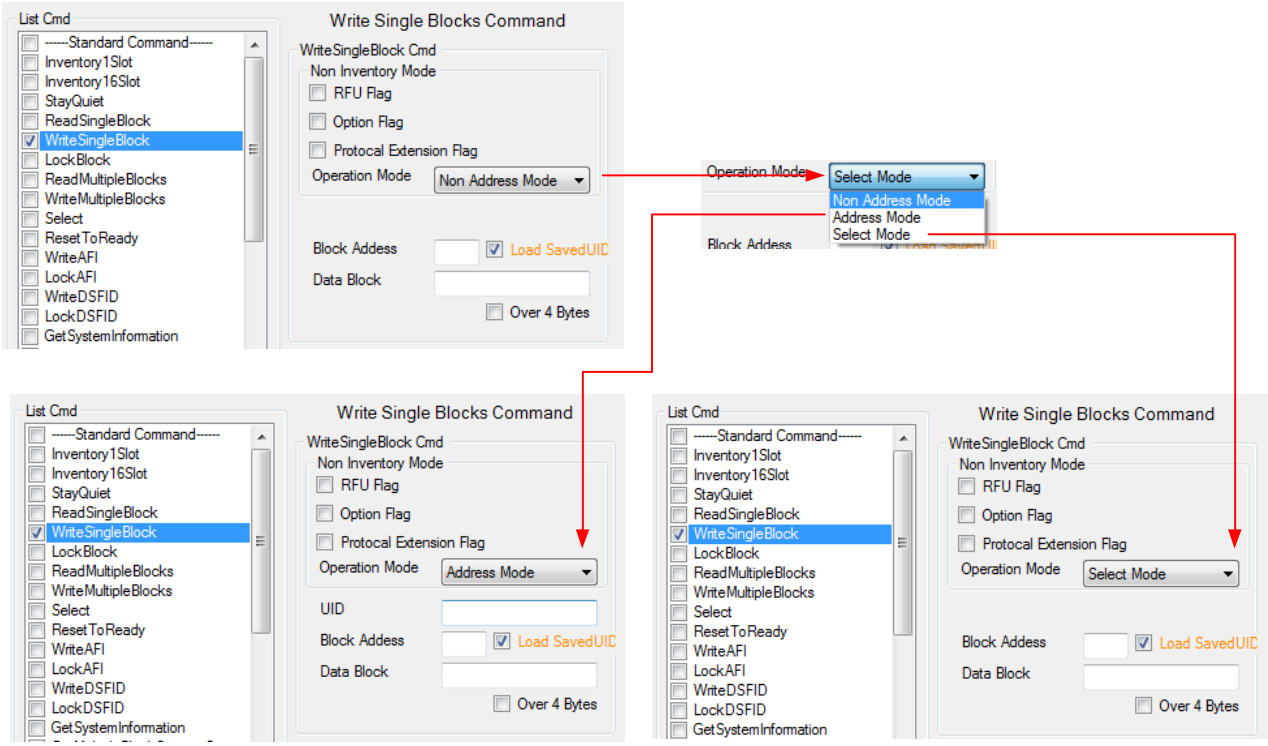


Figure 87 Sub mode in Non-inventory mode

The non-inventory mode is for general operations. In this mode, UID is not required. Except quiet state, all cards in field will be affected. The address mode is for specific-card operation. In this mode, UID is required and the card UID match will only be affected. The select mode is for performing group operation. All cards in selected state

will be affected in this mode. Selection of Non-inventory in the demonstration software is shown in Figure 87. Meaning and functionality of the flags in non-inventory mode is depended on card IC manufacturer.

Note that the inventory mode is mandatory for command "Inventory1Slot" and "Inventory16Slot". Command "Select" and "Stay Quiet" must only be used in address mode. Then, UID is mandatory for these commands. The GUI in demonstration software guides applicable modes for each command.

6.3.1 ISO15693 Standard Command

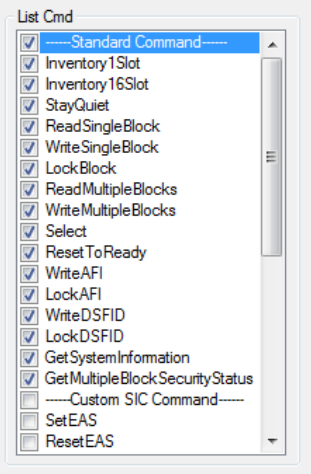


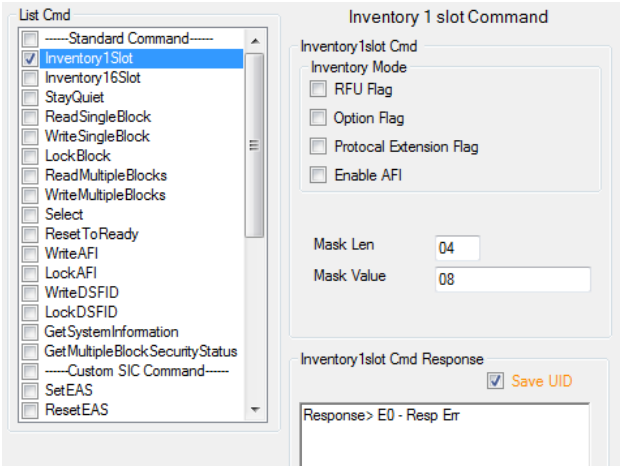
Figure 88 Standard Commands in ISO15693

Table 6-3 Standard Command for ISO15693

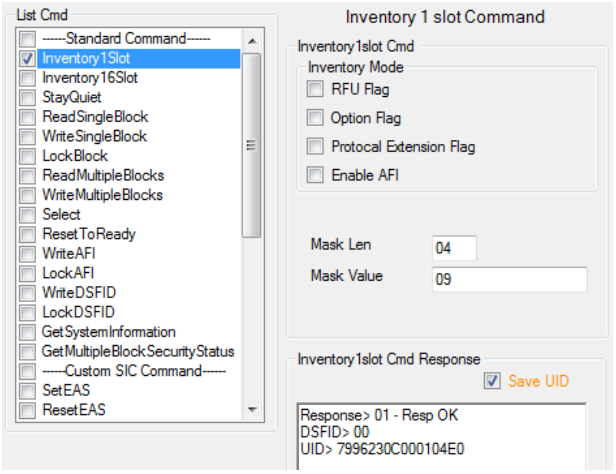
Command Name	Description	Mandatory Input*
Inventory 1 slot	Perform Inventory 1-slot command	MaskLen, MaskValue
Inventory 16 slot (1)	Perform Inventory 16-slot command	MaskLen, MaskValue
Stay Quiet	Perform Stay-Quiet Command	UID
Read Single Blocks	Read Block	Block Address
Write Single Blocks	Write Block	Block Address, Data Block
Lock Block	Lock block	Block Address
Read Multiple Blocks	Read Multiple Block	Block Address, Num of Block
Write Multiple Blocks	Write Multiple Blocks	Block Address, Num of Block, Block size, Data Multi Block
Select	Perform Select Command	UID
Reset to Ready	Perform Reset-to-Ready Command	-
Write AFI	Write AFI value	AFI Value
Lock AFI	Lock AFI value	-
Write DSFID	Write DSFID value	DSFID Value
Lock DSFID	Lock DSFID value	-
Get System Information	Get system information	-
Get Multiple Block Security status	Get multiple block security status	Block Address, Num of Block

6.3.1.1 Inventory 1slot

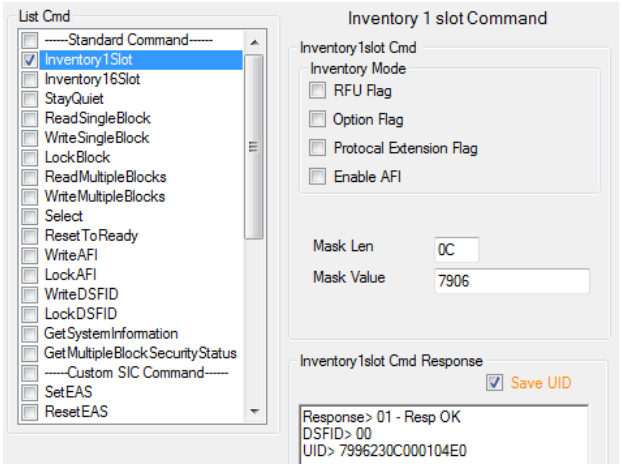
Command **Inventory1slot** is used in determining UID. Except from four flags in inventory mode, parameter Mask Len and Mask Value must be supplied. Mask Value is part of UID used in selecting cards in anti-collision process. Only cards in which part of UID matches to Mask Value will response back. Mask Len is a one -byte data specifying length in bit of Mask Value to be used. Mask Value and Mask Len shall be input in hexadecimal number. If these parameters are input with "00" or left blank, all cards will response. UID in response packet starts from least significant byte to most significant byte, which reflects to what transmit from card in chronological order.



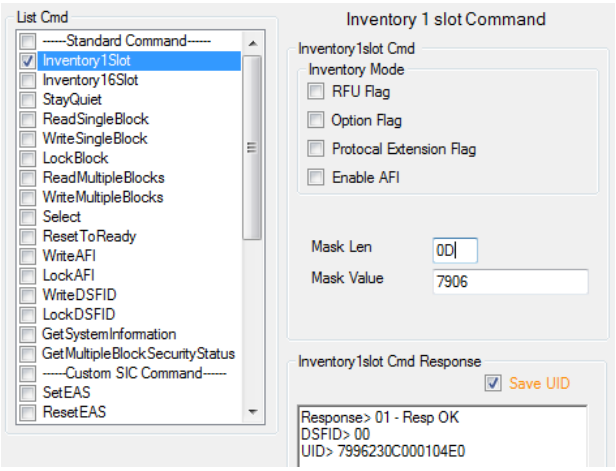
A) Mask Len = 04, Mask Value = 08



B) Mask Len = 04, Mask Value = 09



C) Mask Len = 0C, Mask Value = 7906



D) Mask Len = 0D, Mask Value = 7906

Figure 89 Example of Inventory 1 slot

Figure 89 shows example in executing **Inventory1slot** to a single card with various value of Mask Len and Mask value. The UID of card is "79 96 23 0C 00 01 04 E0". Then, UID can be written in chronological order from left to right hand side as follows.

UID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Hex	79	96	23	0C	00	01	04	E0
Transmitted Binary	1001 1110	0110 1001	1100 0100	0011 0000	0000 0000	1000 0000	0010 0000	0000 0111

A) Mask Len = 04, Mask Value = 08
Four bits of "0001" representing "8" is transmitted with this command. This part of UID is not matched the LSB part the card UID. Therefore, there is no response from card.

UID Hex	79
UID in Transmitted binary	1001 1110
Transmitted mask value	0001

B) Mask Len = 04, Mask Value = 09
Four bits of "0001" representing "8" is transmitted with this command. This part of UID is matched the LSB part the card UID. Therefore, card response a complete UID.

UID Hex	79
UID in Transmitted binary	0001 1110
Transmitted mask value	0001

C) Mask Len = 0C, Mask Value = 7906
12 bits of "0001 1110 0110" is transmitted with this command. This part of UID is not matched the LSB part the card UID. Therefore, card response a complete UID.

UID Hex	79	96
UID in Transmitted binary	0001 1110	0110 1001
Transmitted mask value	0001 1110	0110

D) Mask Len = 0D, Mask Value = 7906
13 bits of "0001 1110 0110 0" is transmitted with this command. This part of UID is not matched the LSB part the card UID. Therefore, there is no response from card.

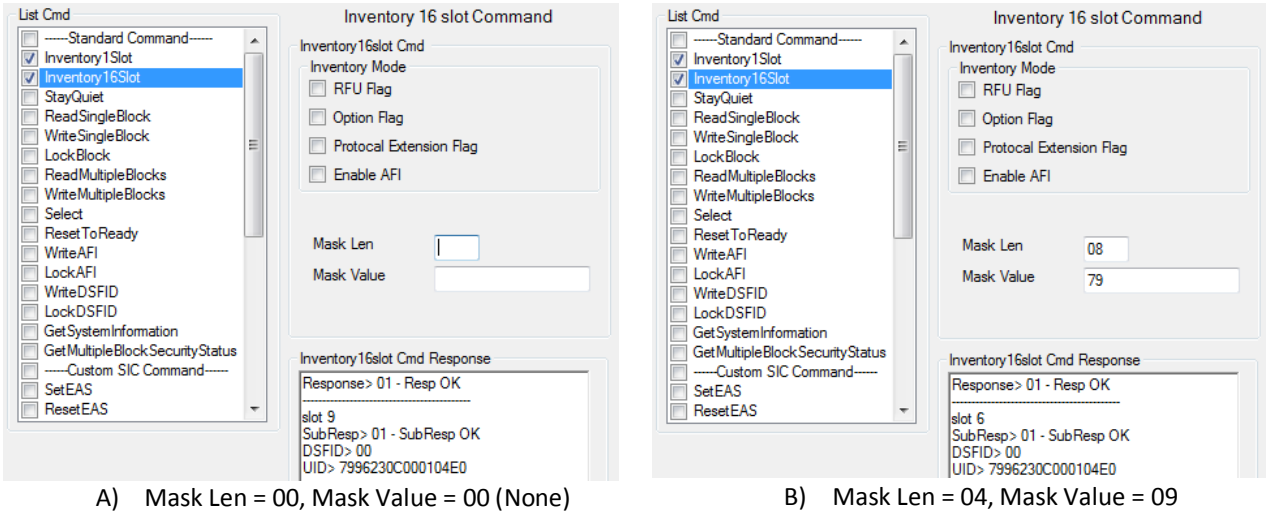
UID Hex	79	96
UID in Transmitted binary	0001 1110	0110 1001
Transmitted mask value	0001 1110	0110 0

6.3.1.2 Inventory 16 slot

Command **Inventory16slot** is used in anti-collision process and determining UID. This command reduces data collision by spreading response from multiple cards into 16 slots systematically. Card will response if Mask Value with in specific Mask Len matches. This is similar to Mask Len and Mask Value in **Inventory 1slot**. In addition, card answers in slot number coincident to 4-bit number in UID next to mask value. Figure 90 shows example in executing **Inventory16slot** to a single card with various value of Mask Len and Mask value. The UID of card is "79 96 23 0C 00 01 04 E0".

A) Mask Len = 00, Mask Value = 00 (None)
The card responds in 9th slot because the UID next to Mask Value is 9 as shown in red number below.
UID = "79 96 23 0C 00 01 04 E0"

B) Mask Len = 08, Mask Value = 79 (None)
The card responds in 6th slot because the UID next to Mask Value is 6 as shown in red number below.
UID = "79 96 23 0C 00 01 04 E0"



A) Mask Len = 00, Mask Value = 00 (None) B) Mask Len = 04, Mask Value = 09

Figure 90 Example of Inventory 16 slot

The response from this command is a concatenated frame of response in each slot. For more information, please refer to “Pi-931 Module Protocol”.

Diagram in Figure 91 shows an idea of anti-collision process by using scanning technique. The collision slot, which is slot 3 and slot E in this example, in the first round, is possibly that multiple cards have responded in the same slot. Hence, the **Mask Value** is set to the number of slot that collision occurred and **Mask Len** is increased by 4. This process is repetitive until there is no collision detected and processing unit collect the UID in slot there is no collision. Practically, error may cause errors in collision slot.

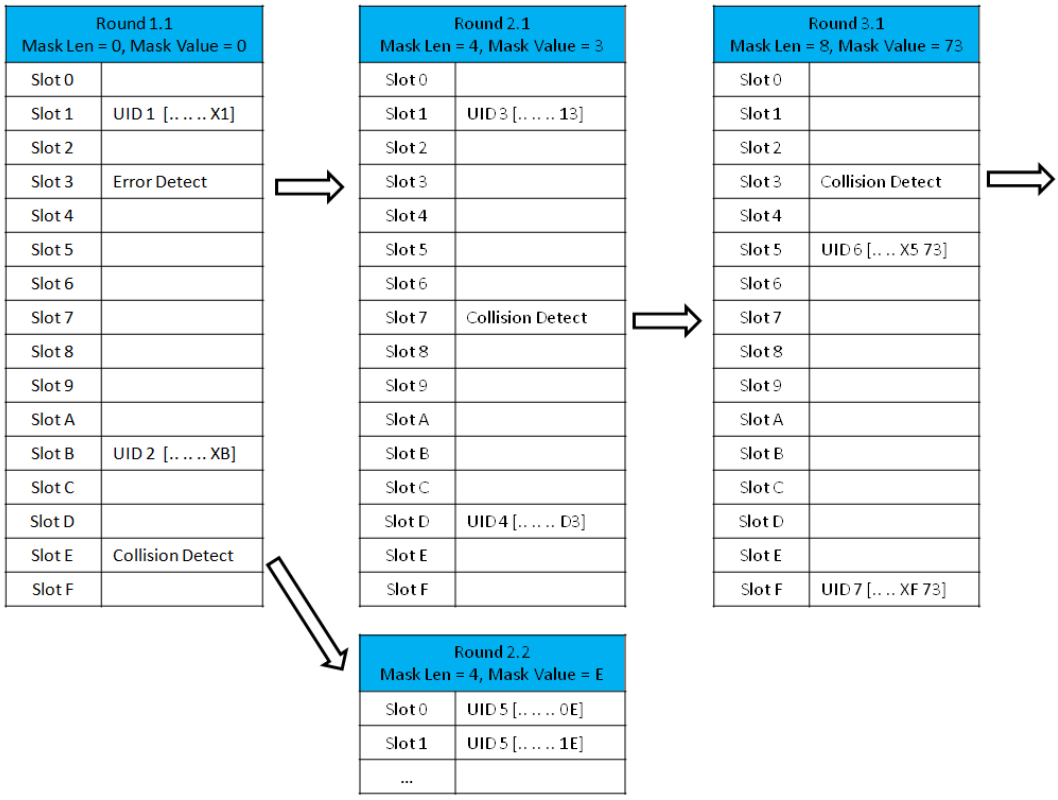


Figure 91 An example of Using Inventory 16 slot in anti-collision

6.3.1.3 Stay Quiet

The **Stay Quiet** put UID-specific card in quiet state. Therefore, UID is mandatory parameter for this command.

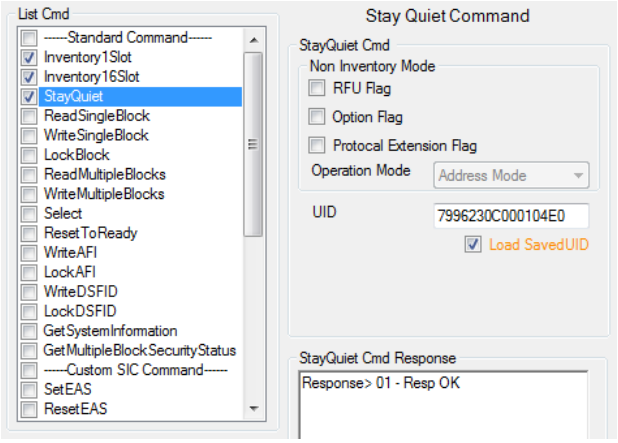


Figure 92 **Stay Quiet** Command

6.3.1.4 ReadSingleBlocks

The **ReadSingleBlocks** performs data reading from a specified block. Block address is only a require parameter.

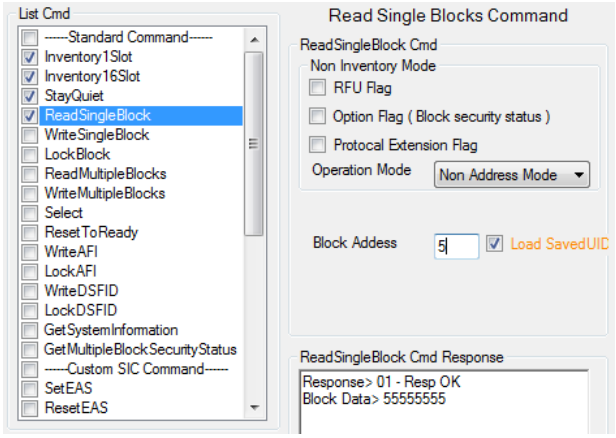


Figure 93 **ReadSingleBlock** command

6.3.1.5 WriteSingleBlocks

The **WriteSingleBlocks** performs data programming to specified block. Block address and data to be written are required parameters. The GUI, as shown in Figure 94, provides 4-byte data to be default block length. For some special card in which block data is wider than the default, user can check “Over 4 bytes” box to accept input up to 8 bytes.

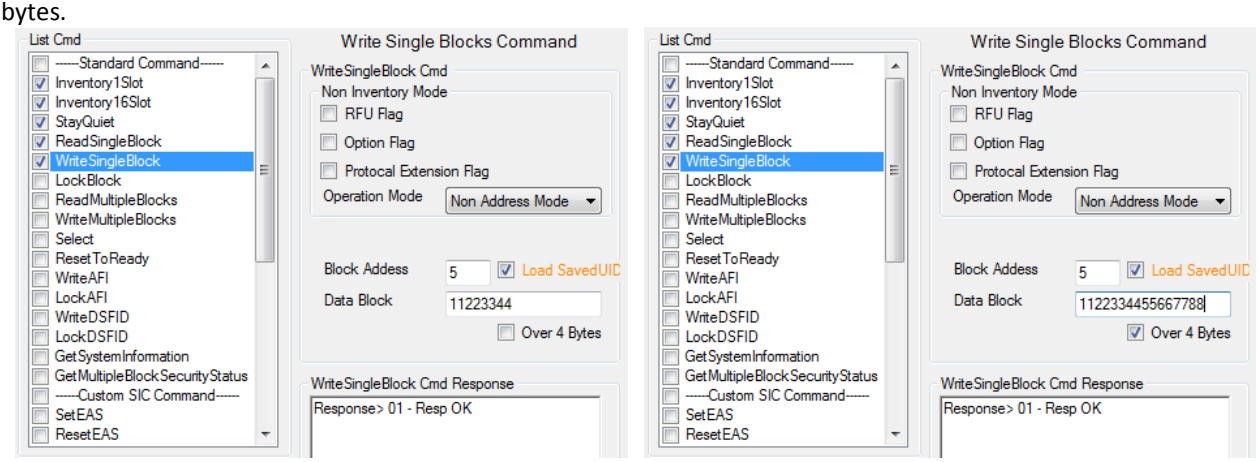


Figure 94 **WriteSingleBlock** command

6.3.1.6 Lock Block

The **LockBlock** command performs content lock in a specific block from modification later. The required parameter is a block address.

6.3.1.1 ReadMultipleBlocks

The **ReadMultipleBlocks** performs data reading from a specified range of data. Parameters as follows must be specified before executing.

Block Address : The first block address to be written

Num of Blocks : The number of additional block from first block to be written

The demonstration software can divide and display data for each block as shown Figure 96, providing that the block size, parameter in software, be specified correctly. Note that cards from some manufacturers might not support this command.

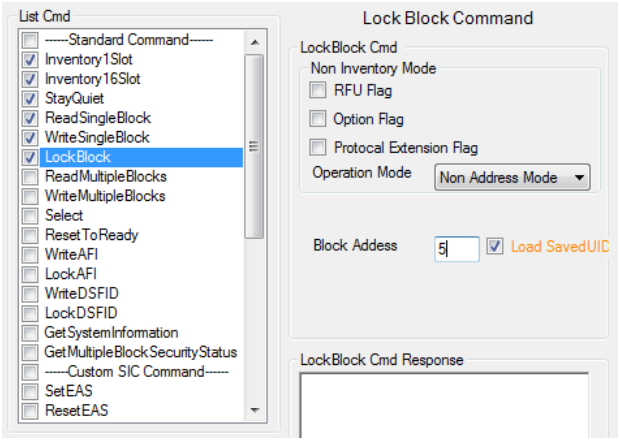


Figure 95 **Lock Block** command

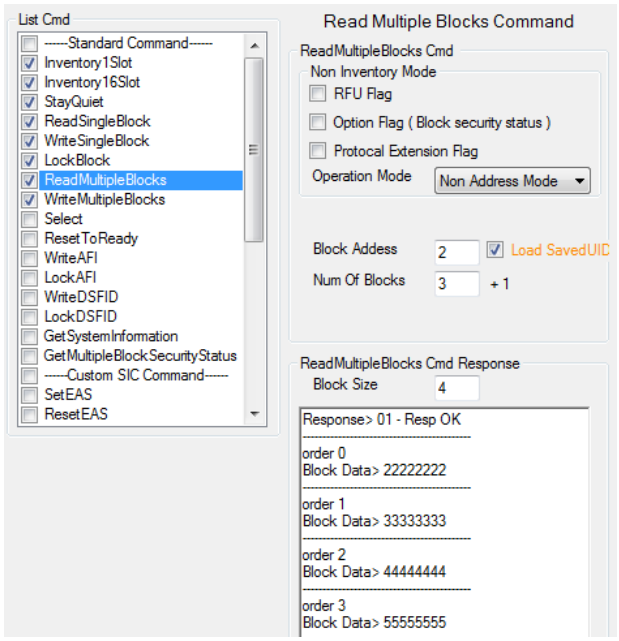


Figure 96 **ReadMultipleBlock** command

6.3.1.2 WriteMultipleBlocks

The **WriteMultipleBlocks** performs data reading from a specified range of data. Parameters as follows must be specified before executing.

Block Address : The first block address to be written

Num of Blocks : The number of additional block from first block to be written

Block Size : The number of bytes in one block

Data Multi Block : Consecutive data for all block to be written

The demonstration software can divide and display data for each block as shown Figure 96, providing that the block size, parameter in software, be specified correctly. Note that cards from most manufacturers might not support this command.

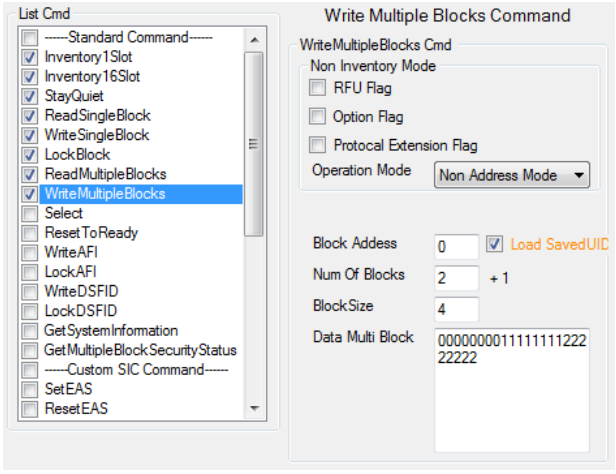


Figure 97 **WriteMultipleBlocks** command

6.3.1.3 Select

The **Select** put UID-specific card in selected state. UID is mandatory parameter for this command.

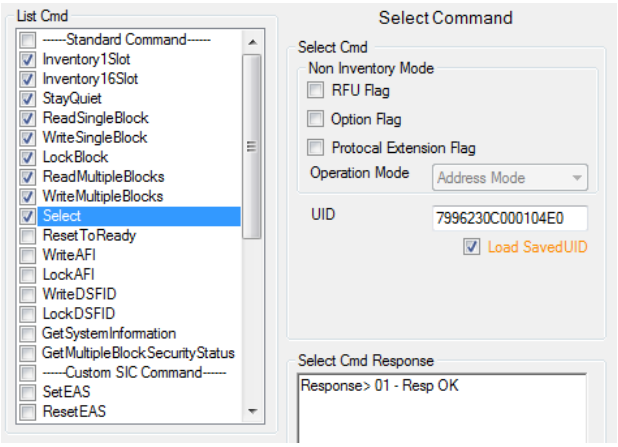


Figure 98 **Select** command

6.3.1.4 ResetToReady

The **ResetToReady** puts card into ready state. No parameter, except UID in selected mode, is required for this command.

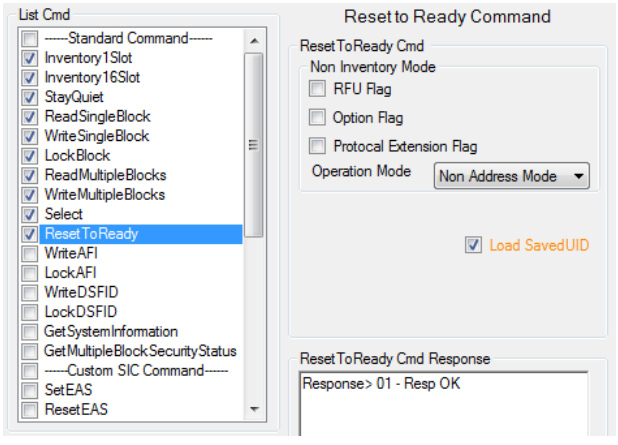


Figure 99 **ResetToReady** command

6.3.1.5 WriteAFI

The **WriteAFI** performs AFI programming. One bytes data for AFI shall be supplied before executing.

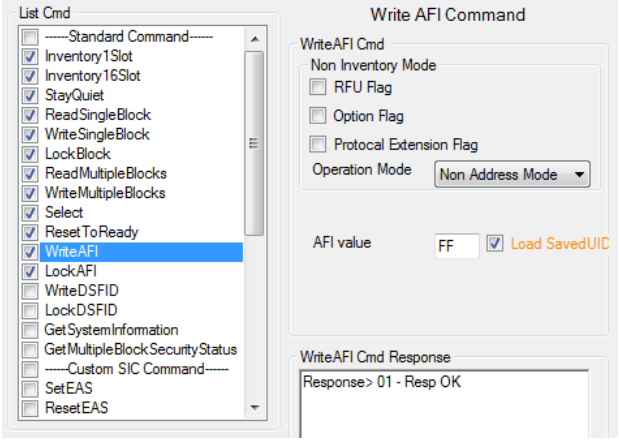


Figure 100 **WriteAFI** command

6.3.1.6 LockAFI

The **LockAFI** performs AFI programming. Once the AFI was locked, AFI content cannot be modified later. No parameter, except UID in selected mode, is required for this command.

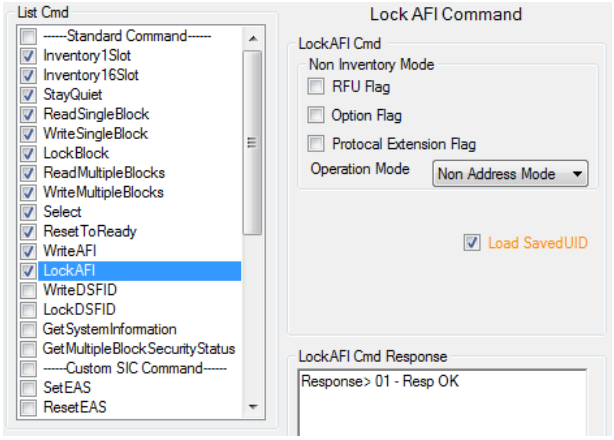


Figure 101 **LockAFI** command

6.3.1.7 WriteDSFID

The **WriteDSFID** performs DSFID programming. One bytes data for DSFID shall be supplied before executing.

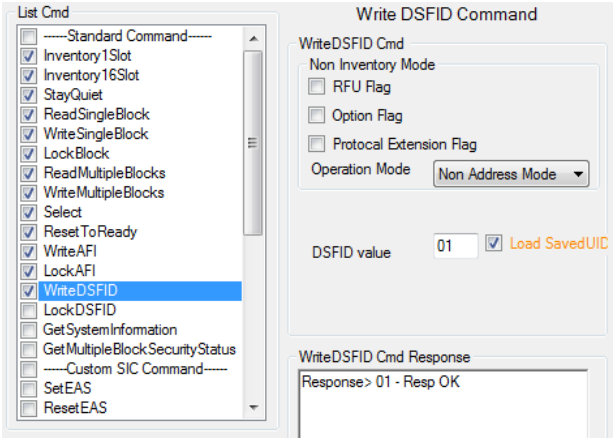


Figure 102 **WriteDSFID** command

6.3.1.8 Lock DSFID

The **LockAFI** performs DSFID programming. Once the DSFID was locked, DSFID content cannot be modified later. No parameter, except UID in selected mode, is required for this command.

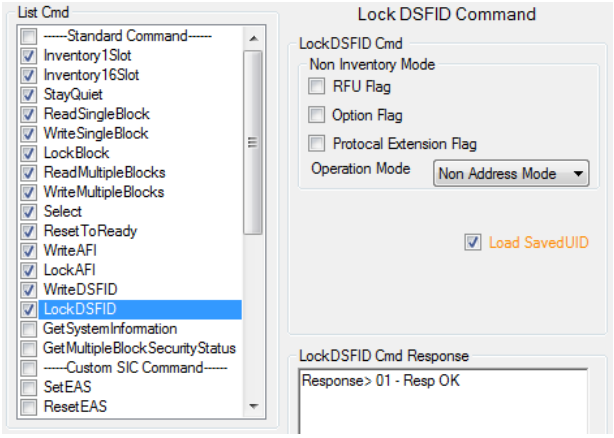


Figure 103 **LockDSFID** command

6.3.1.9 Get System Information

The **GetSystemInformation** retrieve card system information. Information as listed are displayed.

UID : Unique ID

IC reference : 8-bit IC reference number

VICC Memory Size : Card memory size information consisting of block size and number of block

AFI : Application family identifier

DSFID : Data storage format identifier

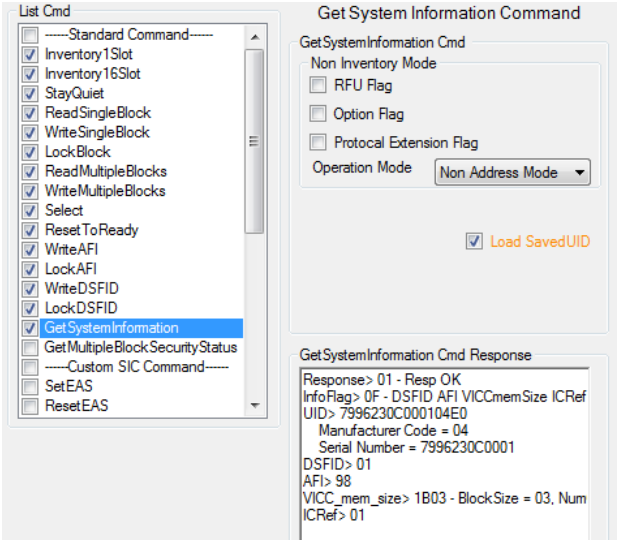


Figure 104 **GetSystemInformation** command

6.3.1.10 Get Multiple Block Security status

The **GetMultipleBlockSecurityStatus** retrieve condition if specific blocks are locked. Parameters as follows must be specified before executing.

Block Address : The first block address to be written

Num of Blocks : The number of additional block from first block to be written

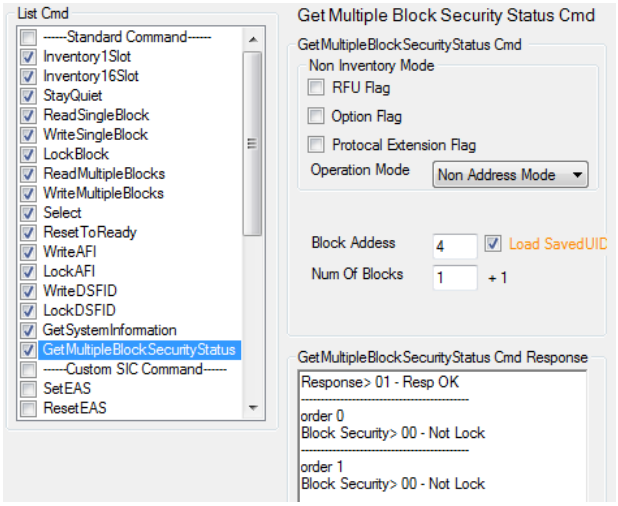


Figure 105 **GetMultipleBlockSecuritystatus** command

6.3.2 Example of ISO15693 standard command usage

Figure 106 shows example of operating standard command. The operations in this example go through ready mode, selected mode and quiet mode. In each mode, inventory command and read block 0 in three non-inventory mode are performed. User can notice that the operating results coincide with the Table 6-2. For more information about standard command, please refer to ISO15693-3 protocol.

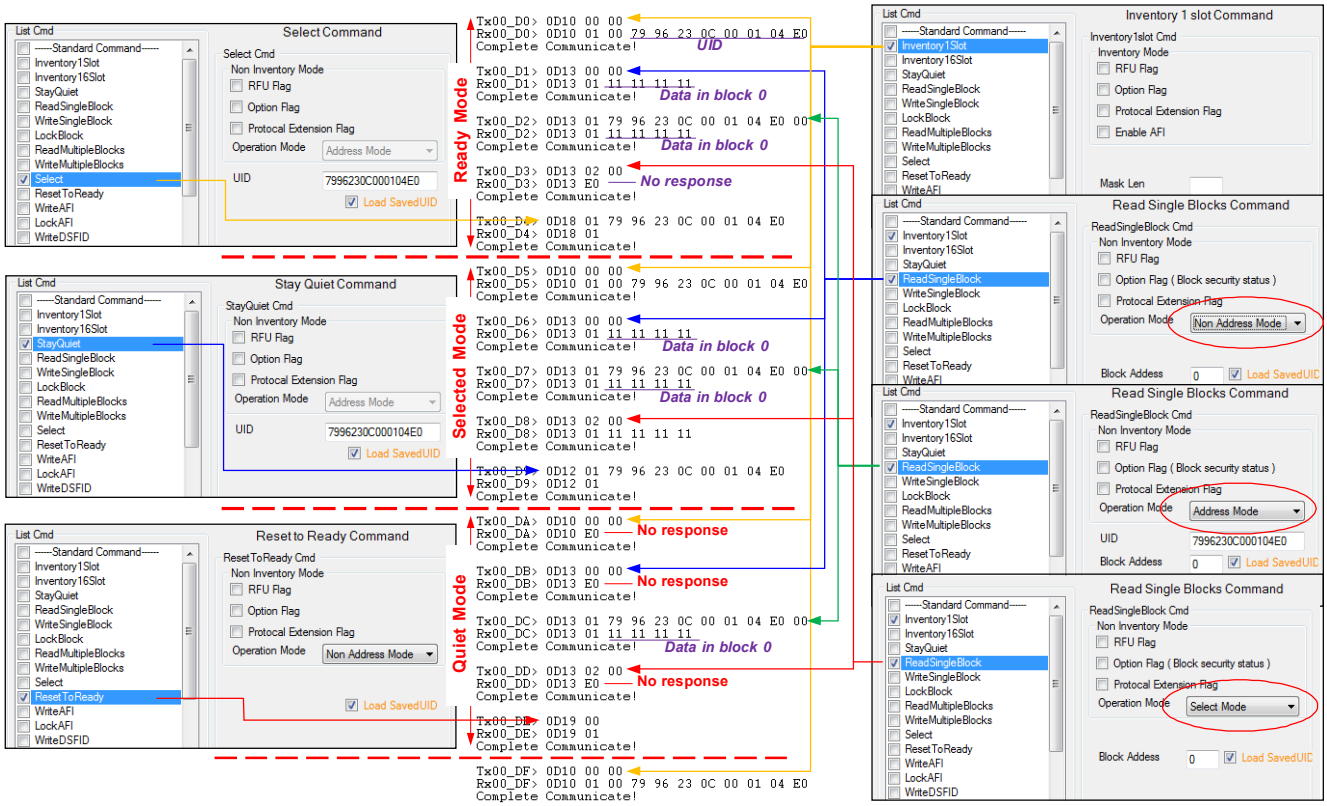


Figure 106 Example of ISO15693 standard command usage

6.3.3 Custom command for SIC5600

SIC5600 is an ISO15693 card IC from silicon craft. To communicate with SIC5600 requires SIC-defined custom commands. The supported custom command in Pi-931 is shown in Figure 107 and summarized in Table 6-4. For more information about SIC5600, please refer to SIC5600 datasheet.

Table 6-4 Custom Command for SIC5600

Command Name	Description	Mandatory Input*
Set EAS[1]	Enable EAS mode (If EAS mode is not locked)	-
Reset EAS	Disable EAS mode (If EAS mode is not locked)	-
Lock EAS[2]	Lock current status of EAS mode	-
EAS Alarm	Invoke SIC5600 to transmit EAS code (Card must be preset in EAS mode)	-
Kill [3]	Destroy a specific-UID card	UID, Kill Code
Load Password	Login to enable accessing card content	UID, PWD Mode, Random Number, Password
Write Password	Write password 0, password 1 or kill code	UID, Selected Password, Password
Lock Password[2]	Lock current password or kill code	UID, Selected Password
Set Password Mode[4]	Setup password mode or kill enable	UID, PWD Mode
Lock Password Mode[2],[4]	Lock current status of the password mode	UID
Get Password Mode[4]	Get current status of the password mode	UID
Set OTP [5]	Set OTP mode	UID, OTP Mode
Get OTP [5]	Get current status of the OTP mode	UID
Write XUID [6]	Write XUID to a specific-UID card	UID, XUID
Read XUID [6]	Write XUID from a specific-UID card	UID

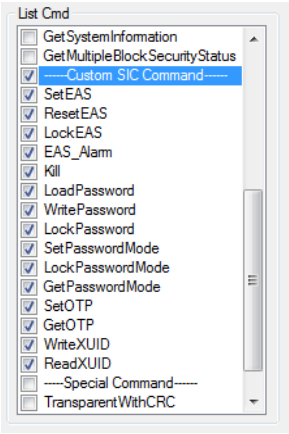
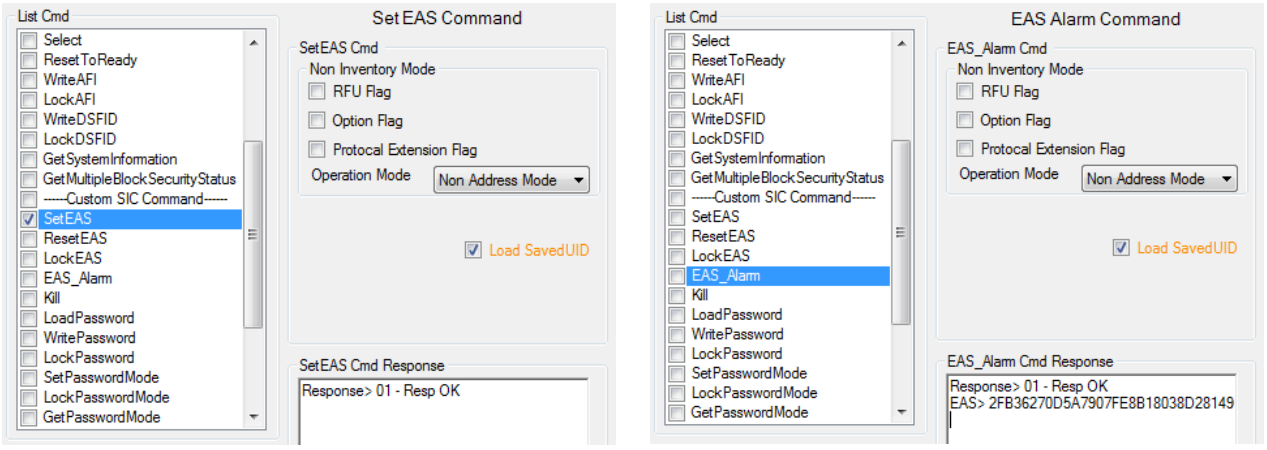


Figure 107 Custom commands for SIC5600

6.3.3.1 Set EAS

EAS (Electronic Article Surveillance) is a configurable mode in SIC5600. If EAS mode is set, **EAS Alarm** command can invoke card to transmit EAS code. The command **Set EAS** requires no input parameter. Figure 108A shows successful response from **Set EAS** and Figure 108B show response of **EAS Alarm** after performing **Set EAS**.

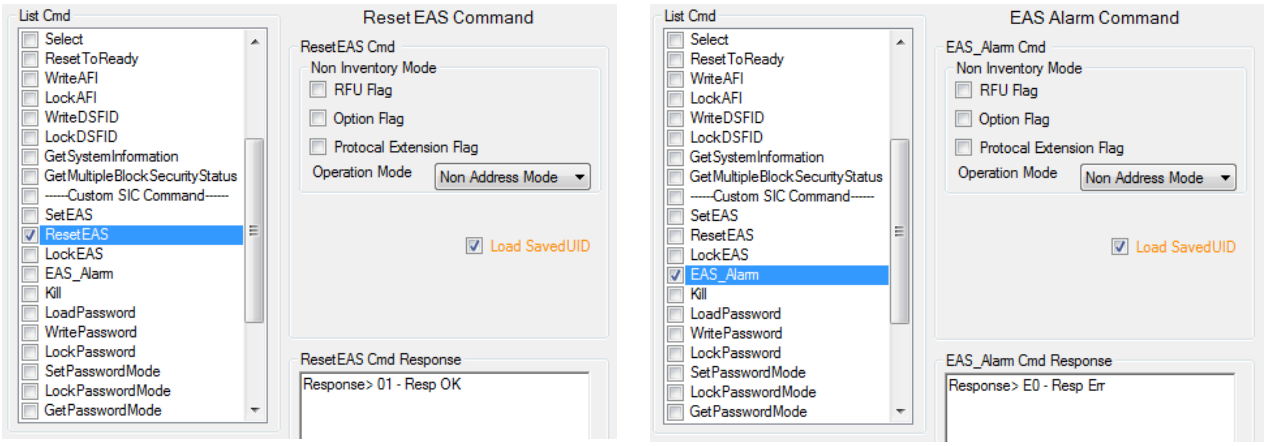


A) Perform **Set EAS** B) **EAS Alarm** Response after performing **Set EAS**

Figure 108 **Set EAS** command

6.3.3.2 Reset EAS

If EAS mode is reset, **EAS Alarm** command can not invoke card to transmit EAS code. The command **Reset EAS** requires no input parameter. Figure 109A shows successful response from **Reset EAS** and Figure 109B shows no response of **EAS Alarm** after performing **Set EAS**.

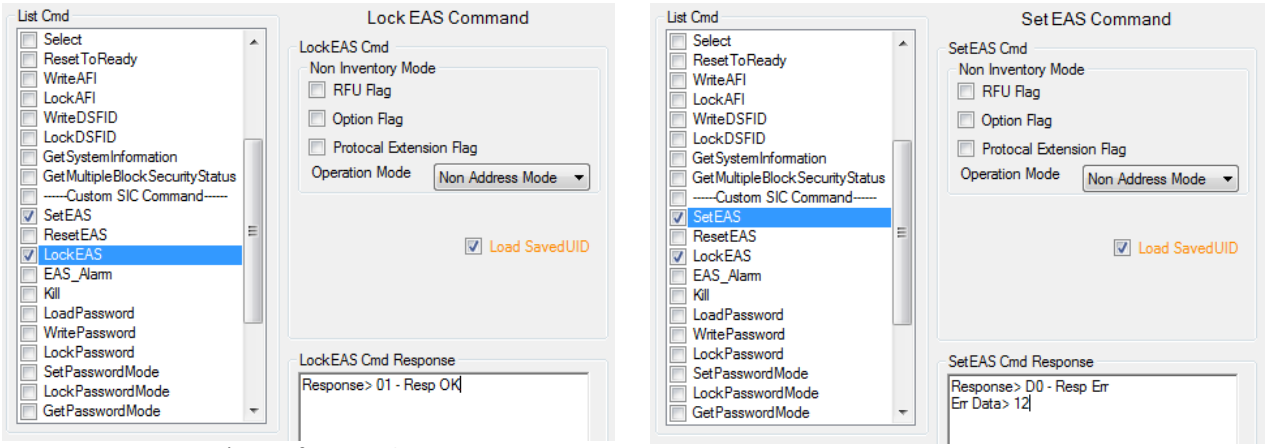


A) Perform **Reset EAS** B) **EAS Alarm** response after performing **Reset EAS**

Figure 109 **Reset EAS** command

6.3.3.3 Lock EAS

Command **Lock EAS** locks EAS setting configuration permanently. After performing Lock EAS, **Set EAS** and **Reset EAS** cannot successfully execute as shown in Figure 110.



A) Perform **Lock EAS** B) **Settees** response after performing **Lock EAS**

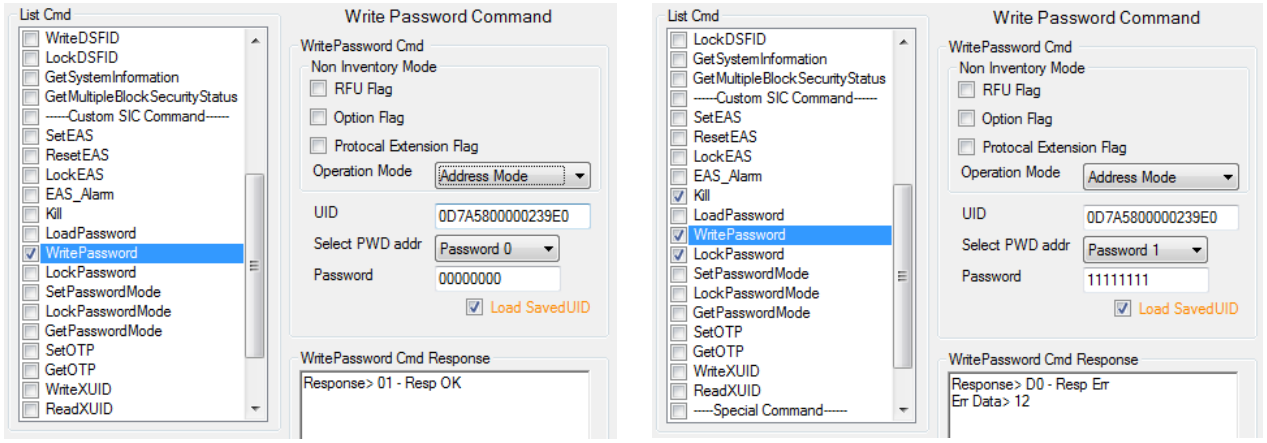
Figure 110 **Lock EAS** command

6.3.3.4 EAS Alarm

Command **EAS Alarm** can invoke card to transmit EAS code. As shown in Figure 108B and Figure 109B, EAS value for SIC5600 is "2FB36270D5A7907FE8B18038D281497682DA9A866FAF8BB0F19CD112A57237EF".

6.3.3.5 Write Password

Command **Write Password** used to program Password 0, Password 1 and Kill code. UID, password value and target password are required parameters for this command. If password being written was locked, programming will not be successful with return code as shown in Figure 111B.



A) **Write Password** successful B) Perform **Write Password** while password was locked

Figure 111 **Write Password** command

6.3.3.6 Lock Password

Command **Lock Password**, the GUI is shown Figure 112, locks current target password value permanently. UID and target password are required parameters for this command.

6.3.3.7 Set Password Mode

Command **Set Password Mode** is for enabling Kill-enable bit and defining password mode. If Kill-enable bit is set, **Kill** command can be disable card. Password mode, consists of two control parameters namely Password Allocation (PA) and Security Mode(SM), defines password to be used and accessibility for read and writes SIC5600 content. Functionality of each password mode, which is combination Password Allocation, and Security Mode is summarized in Table 6-5. Figure 113A shows effect from setting Password 0 for read protection and Password 1 for

write protection. As a result, reading block 0 without loading password is inhibit as shown in Figure 113B. UID, Kill-enable and password mode are required parameters for this command.

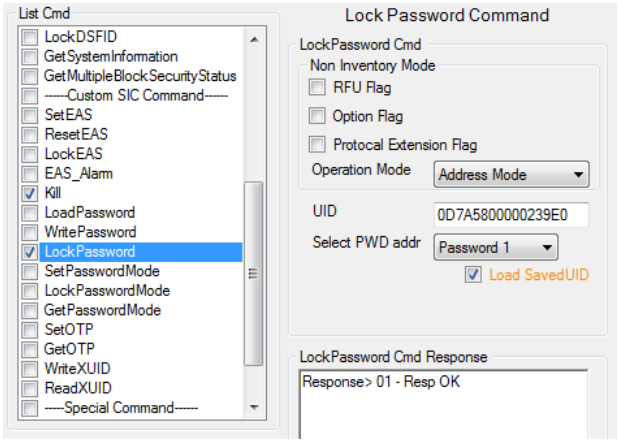
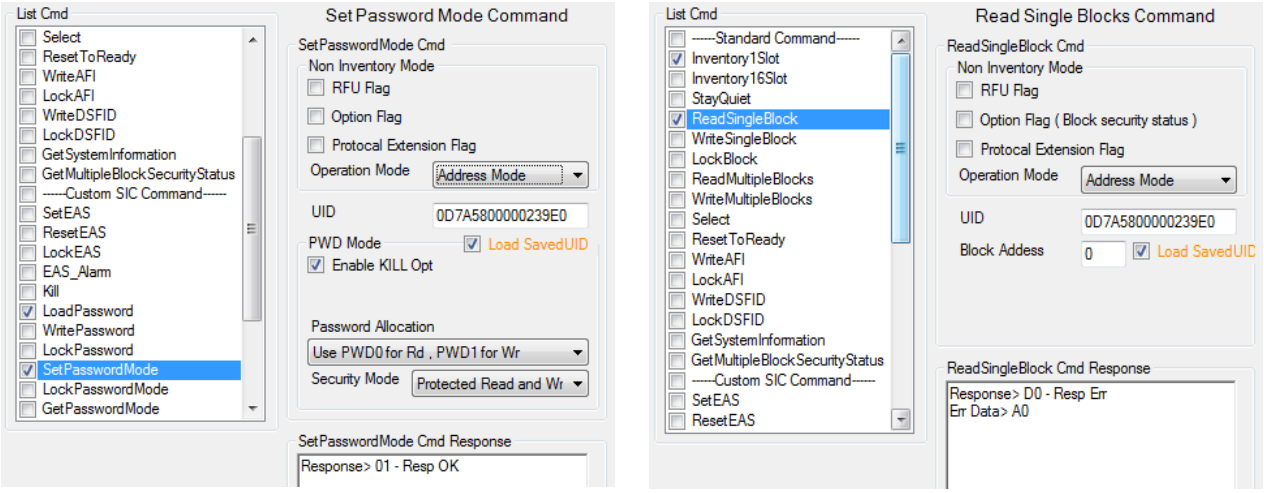


Figure 112 **Lock Password** command



A) Set Password Mode (PA = 00 , SM = 11) B) Reading during read protection

Figure 113 **Set Password Mode** command

Table 6-5 Summarized protection function controlled by PA (Password Allocation) and SM (Security Mode)

PA (Password Allocation)		SM (Security Mode)		Protection Function
00b	Use Password 0 for read protection and Password 1 for write protection	00b	No protection	No protection
		01b	Read Protection Only	Read Protection by Password 0, M = 0
		10b	Write Protection Only	Write Protection by Password 1, M = 1
		11b	Read and Write Protection	Read Protection by Password 0, M = 0 and Write Protection by Password 1, M = 1
01b	Use Password 0 for all protection	00b	No protection	No protection
		01b	Read Protection Only	Read Protection by Password 0
		10b	Write Protection Only	Write Protection by Password 0
		11b	Read and Write Protection	Read & Write Protection by Password 0
10b	Use Password 1 for all protection	00b	No protection	No protection
		01b	Read Protection Only	Read Protection by Password 1
		10b	Write Protection Only	Write Protection by Password 1
		11b	Read and Write Protection	Read & Write Protection by Password 1
11b	Use Both Password 0 and Password 1 (64 Bits) for all protection	00b	No protection	No protection
		01b	Read Protection Only	Read Protection by Password 0 + Password 1
		10b	Write Protection Only	Write Protection by Password 0 + Password 1
		11b	Read and Write Protection	Read & Write Protection by Password 0 + Password 1

6.3.3.8 Lock Password Mode

Command **Lock Password Mode**, the GUI is shown Figure 114, freezes current password mode value permanently. UID is a required parameter for this command.

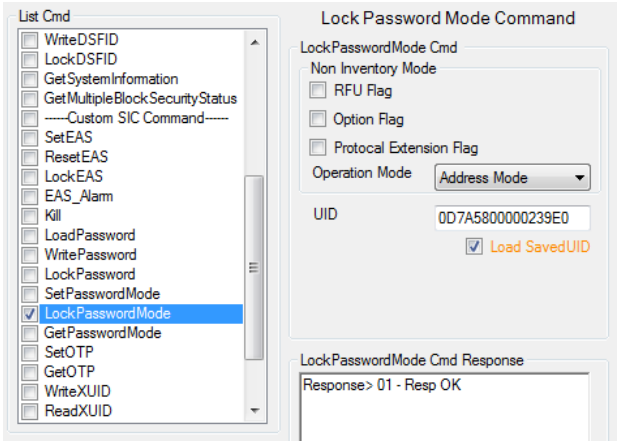


Figure 114 **Lock Password Mode** command

6.3.3.9 Get Password Mode

Command **Get Password Mode**, the GUI is shown Figure 115, is used to obtain current password mode. UID is a required parameter for this command. The structure of returned password mode is shown in Table 6-6 to Table 6-10.

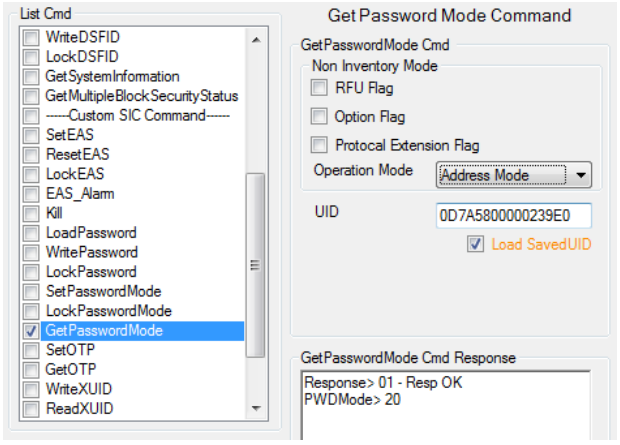


Figure 115 **Get Password Mode** command

Table 6-6 Password mode (PWD Mode)								
Bit	7	6	5	4	3	2	1	0
Name	0	0	Kill Enable	M	PA (Password Allocation)		SM (Security Mode)	

Table 6-7 Kill Enable : Set up Kill-Enable bit	
0b	Reset Kill-Enable bit in SIC5600 to protect card from Kill command
1b	Set Kill-Enable bit in SIC5600 to enable Kill command to operate.

Table 6-8 M : Specify password in card used to compare with transmitted password during loading password	
0b	Use Password 0
1b	Use Password 1

Table 6-9 PA (Password Allocation) : Define passwords to be used in protection	
00b	Use Password 0 for read protection and Password 1 for write protection. If card was set in this mode, input PWD shall match Password 0 during loading password with M value of 0 to read content. To write data to card, input PWD shall match Password 1 during loading password with M value of 1.
01b	Use Password 0 for protection. If card was set in this mode, input PWD shall match Password 0 during loading password with M value of 0 to read/write content.
10b	Use Password 1 for protection. If card was set in this mode, input PWD shall match Password 1 during loading password with M value of 1 to read/write content.
11b	Use Both Password 0 and Password 1 (64 Bits) for protection. If card was set in this mode, input 8-bytes PWD matched Password 0 and Password 1 must be used for loading password with any value of M to read/write content.

Table 6-10 SM (Security Mode(1:0)) : Define protection function of the password	
00b	No protection
01b	Read Protection Only
10b	Write Protection Only
11b	Read and Write Protection

6.3.3.10 Load Password

Command **Lock Password**, the GUI is shown Figure 116A, is used to login to grant accessibility to read or write. Parameters as follows must be specified before executing.

UID : Card Unique ID

Access Password : Card Password (0 or 1) used to compare with transmitted password

Password Allocation : Password protection function in card being access

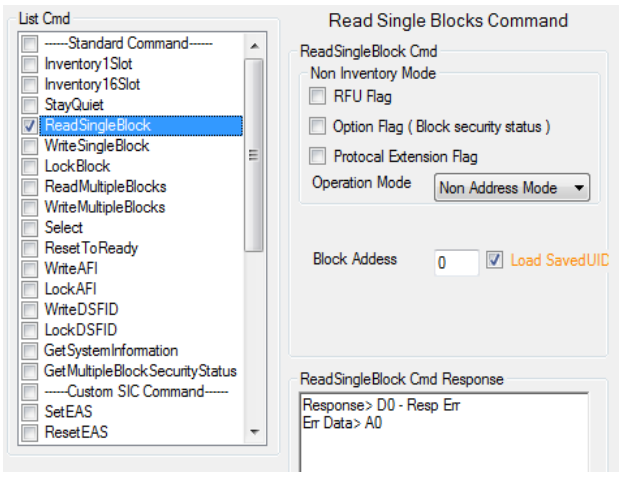
Security Mode : Security mode in card being access

Random Number : 4-byte or 8-byte random number used to encrypt transmitted data on air

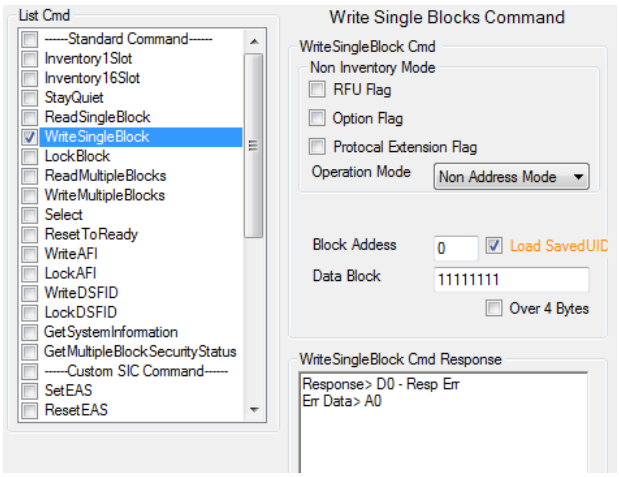
Password : 4-byte or 8-byte password value matched access password

If Password Allocation is “Use PWD 64 bit (PWD0+PWD1)”, 8-byte random number and 8-byte password value matched password 0 consecutive with password 1 shall be supplied. Practically, RNG should be altered every time to prevent playback attack

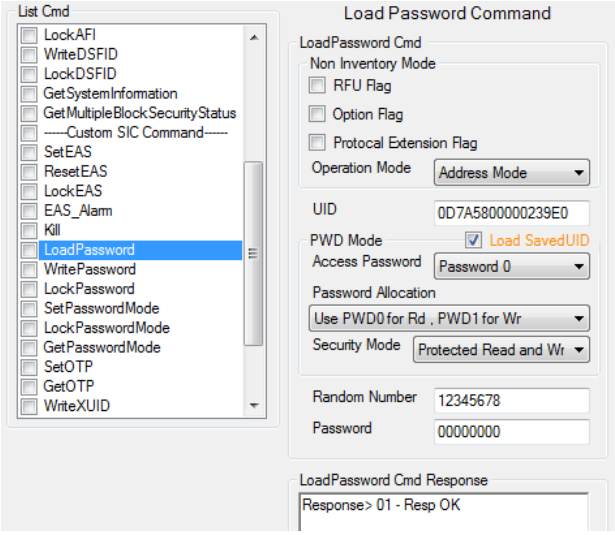
The example of steps in executing **Lock Password** and its results are shown in Figure 116. Assume that card being accessed was previously programmed with password 0 of 00000000 and password 1 of 00000000, and password mode was set in “Password 0 for read protection and Password 1 for write protection” (from Figure 113).



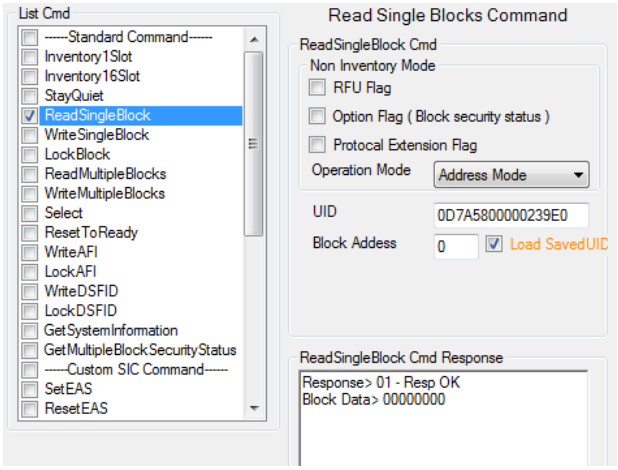
A) Reading data from block 0 is protected



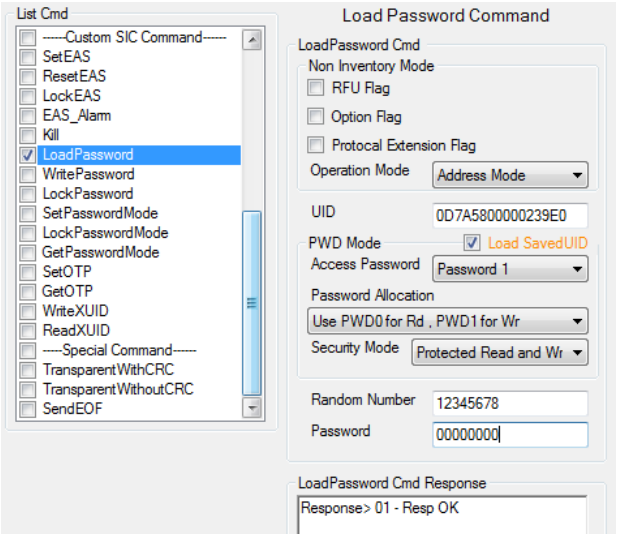
B) Programming data from block 0 is protected



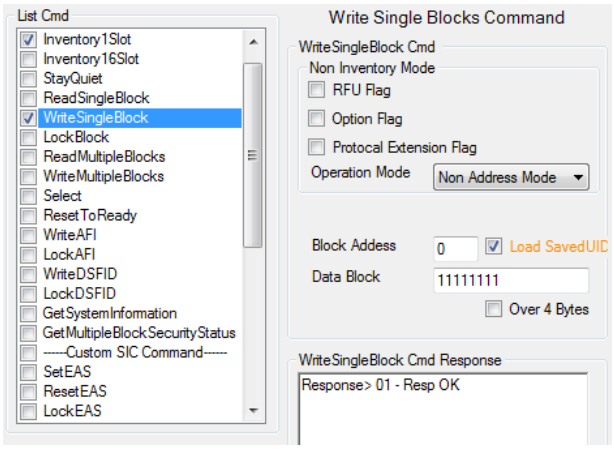
C) Load password with password 0 to access read Successful



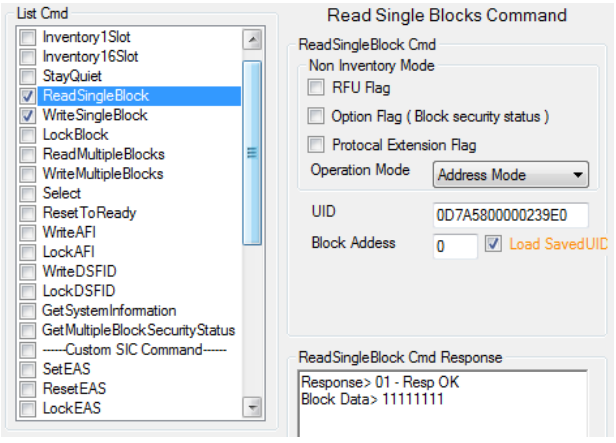
D) Successful reading data from block “0”



E) Load password with password 1 to access write Successful



F) Successful programming data from block “0”

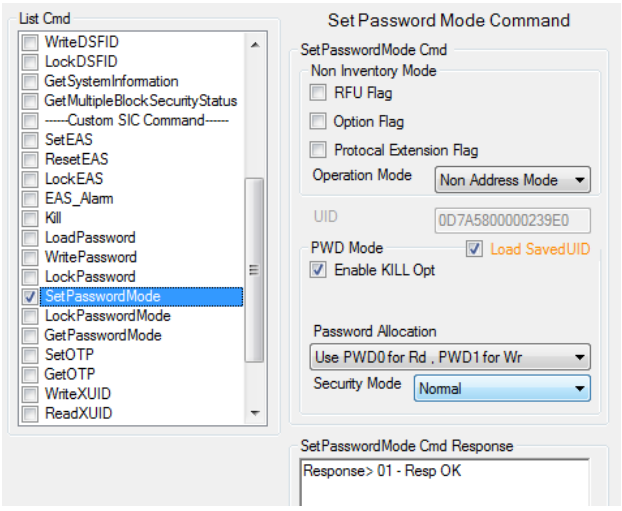


G) Confirm that data was programmed into block 0

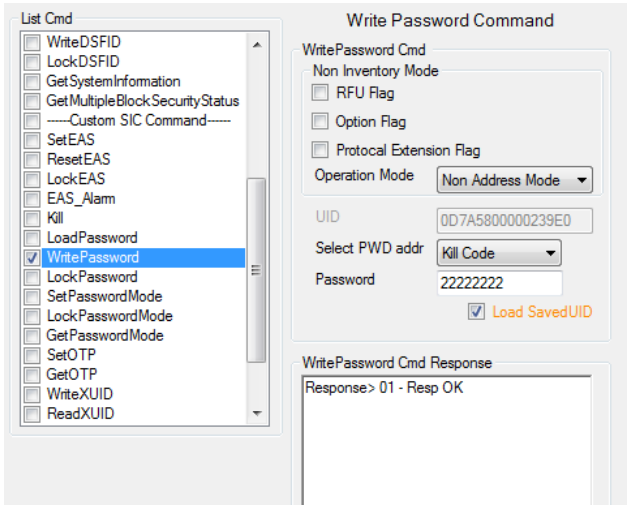
Figure 116 **Load Password** command

6.3.3.11 Kill

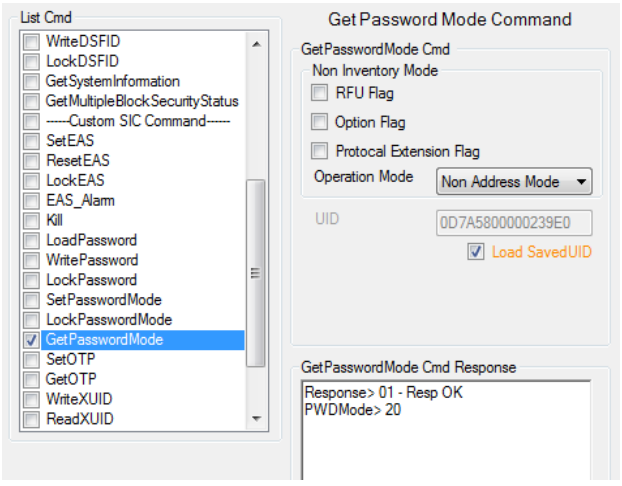
The Kill command will permanently disable card function. The card has been killed will not response to any request from reader. The Kill GUI is shown in Figure 117E. The UID and Kill code are required for this command. Following steps, depicting in Figure 117, demonstrate preparation for kill and Kill operation.



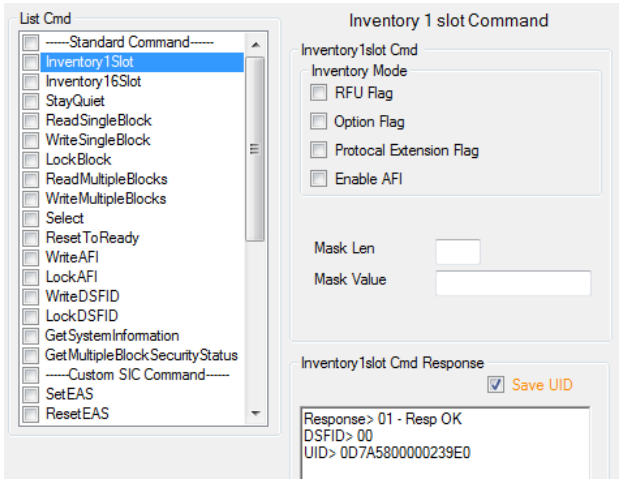
A) Set Kill-Enable by **Set Password Mode**.



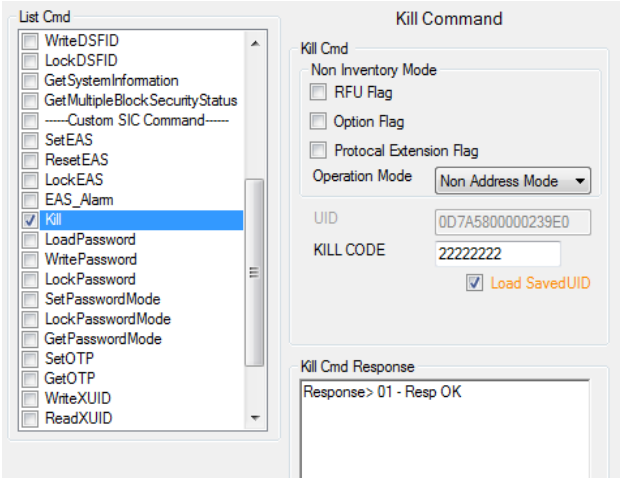
B) Write Kill Code by **Write Password Command**.



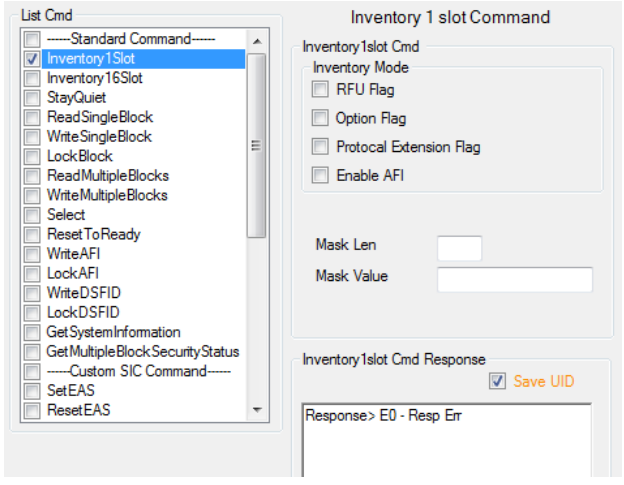
C) Check Password Mode if Kill-Enable was set.



D) Check if card is still alive by inventory



E) kill card by Kill command

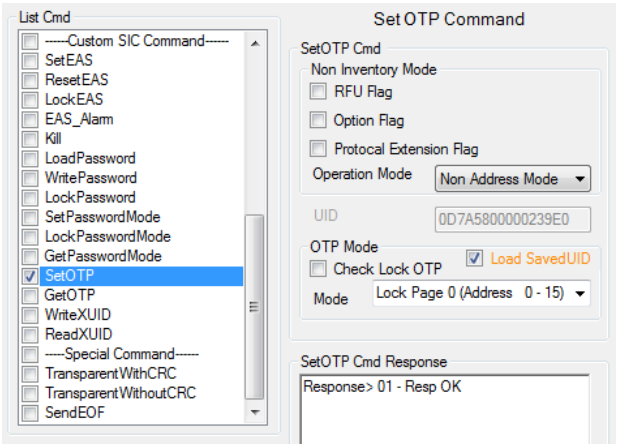


F) No response confirms that card was killed

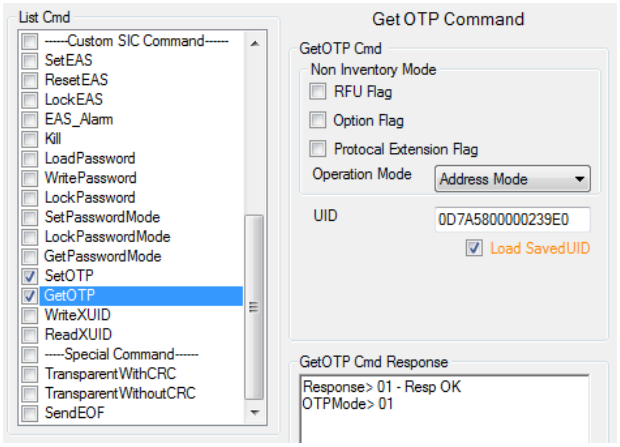
Figure 117 Example of **Kill** operation

6.3.3.12 Set OTP

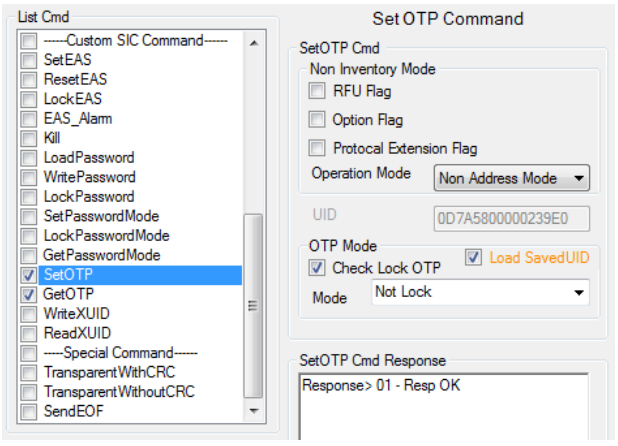
OTP is one time programmable feature. If an OTP bit is set, the data in associated section can not modified forever. Note that if the OTP is set to once, it can not reset to zero. In the provided GUI of Set OTP as shown in Figure 118, user can lock target page or lock OTP configuration itself.



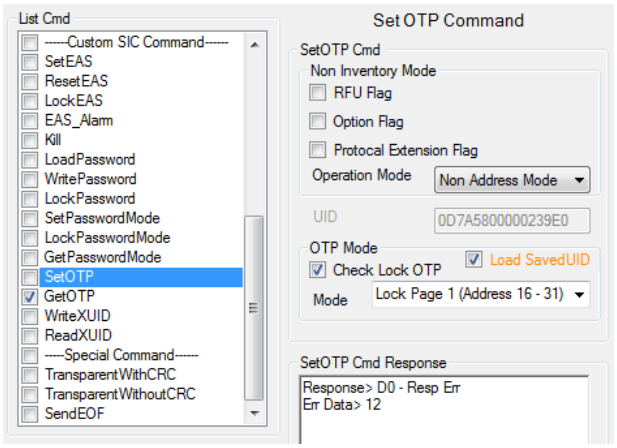
A) Perform lock Page 0 By **Set OTP**



B) Perform lock Page 0 By **Get OTP**



C) Perform lock OTP configuration By **Set OTP**



D) Perform lock Page 1 By **Set OTP** failed because OTP configuration was locked.

Figure 118 **Set OTP** command

6.3.3.13 Get OTP

The **Get OTP** command is used to obtain current status of OTP Mode. As shown in Figure 119, OTP Mode of "00" (hexadecimal) was reported. The structure of OTP Mode is shown in Table 6-11 to Table 6-13.

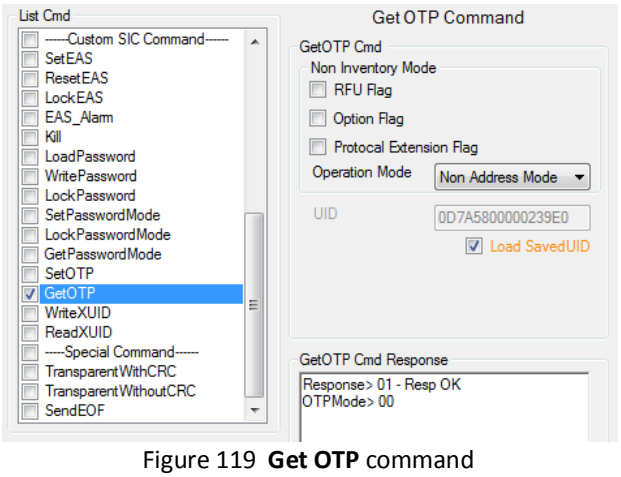


Figure 119 **Get OTP** command

Table 6-11 OTP Mode								
Bit	7	6	5	4	3	2	1	0
Name	0	0	0	0	0	L	OTP MODE(1:0)	

Table 6-12 L : Lock Control to Lock current status of OTP MODE	
0b	Not Lock OTP
1b	Lock OTP. If Lock OTP bit was set to one, it can not reset back to zero and OTP MODE value can not be altered later.

Table 6-13 OTP MODE(1:0)	
00b	Not Lock
01b	Lock Page 0 (Address 0 - 15)
10b	Lock Page 1 (Address 16 - 31)

6.3.3.14 Write XUID

XUID is a one-time writable 6-byte extended Unique ID which can be in some specific user purpose. If XUID has been once written, XUID cannot be modified later. User can input required XUID value in the GUI of **Write XUID** as shown in Figure 120. UID is required for this command.

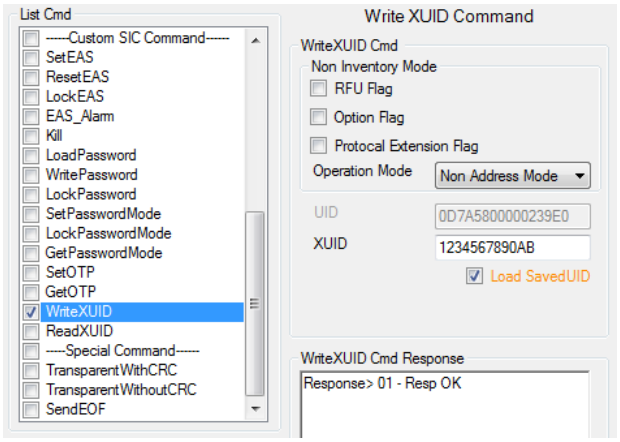


Figure 120 **Write XUID** command

6.3.3.15 Read XUID

The Read XUID is used to retrieve XUID. UID is required for this command. Result of 112-bit complete UID and XUID is reported.

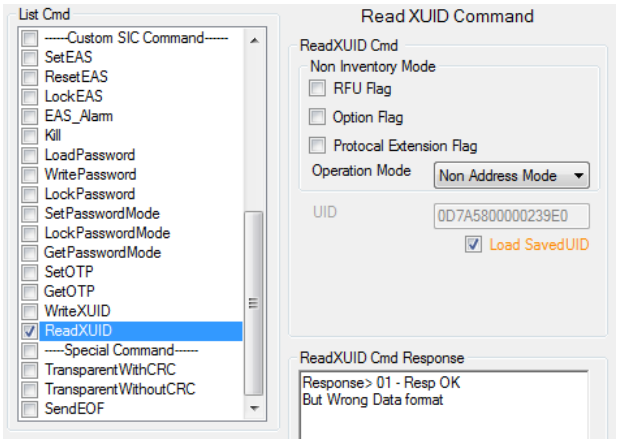


Figure 121 **Read XUID** command

6.3.4 Special command

Similar to ISO14443A and B, the special command available in ISO15693 are **TransparentWithCRC**, **TransparentWithoutCRC** and **SendEOF** as shown in Figure 122.

For both transparent commands, user can input raw hexadecimal code to invoke operation as stated in the RF protocol. The example of Reading data from block 0 of ISO15693 card is shown in Figure 124. Also, User shall review result in transaction logs windows. Moreover, user might need to adjust timeout for card response to be suitable to user applications. The option for timeout in the GUI is shown in Figure 123.

The **SendEOF** is used to transmit a gap in ISO15693 anti-collision process to indicate transition to next slot. However, inventory 16 slot command has already embeds transmitting gap in its process. So, the **SendEOF** can be used in conjunction with transparent command.

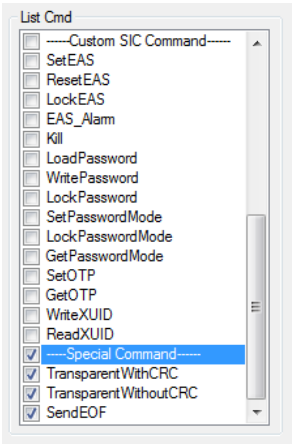


Figure 122 Special commands for ISO15693

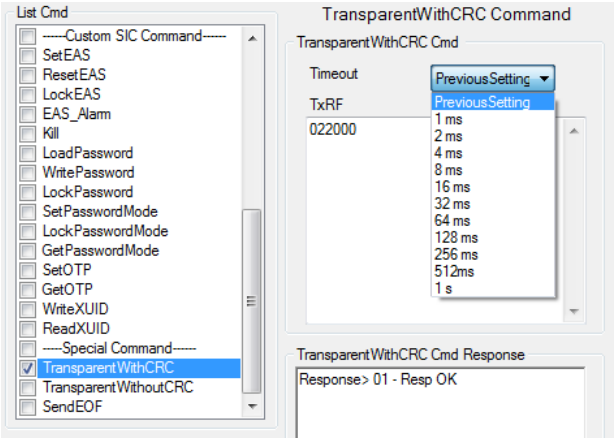


Figure 123 Timeout setting for **TransparentWithCRC** (**TransparentWithoutCRC**) command

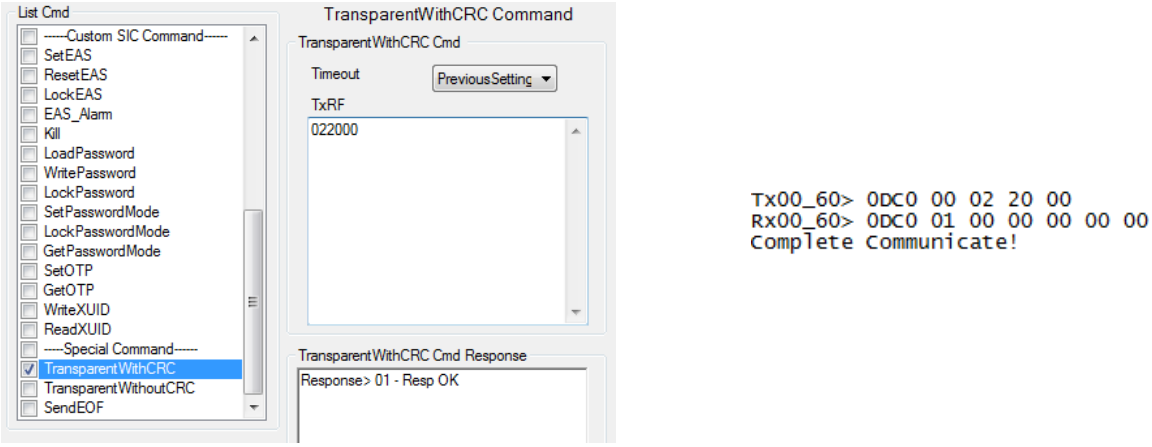


Figure 124 Reading data from block 0 by using **TransparentWithCRC**

6.4 Pico Tag

PicoTag, PicoPass are families of contactless memory IC from INSIDE CONTACTLESS. PicoPass is secure version equipped with INSIDE security encryption mode. By using PI931, user cannot access Read/Write area of PicoPass and PicoCrypt except UID. For PicoTag 16K and 2K, the read/write area is unprotected.

Table 6-14 Pico Product Family

Product	ISO standards	Cryptographic Security	Memory size and personalization
PicoPass 16KS	ISO15693 & ISO14443B	No	(8 x 2K pages) or (1 x 16K page)
PicoPass 2KS			2K bits
PicoTag 16K	ISO15693	No	1 x 16K page
PicoTag 2K			2K bits
PicoTag 2KS		Yes	2K bits

The radio frequency power and signal interface for data transmission of PicoTag and PicoPass rely on ISO15693-2 and ISO14443B-2. In another word, bit definition of RF signal in both transmission and reception follows ISO15693-2 and ISO14443B-2. Pico-family commands are proprietary and do not conform ISO15693-3. These commands are called "ISO15693-2 and ISO14443B -2 commands". However, PicoPass does support ISO14443B-3 but the IC must be prior set in that mode. The Pi931-XX, where the SIC9310 is used as an RF front-end chip supporting such standards, can communicate to PicoPass contactless IC.

Figure 125 shows GUI in Pico Tag tap. In this tap, user can transmit single or multiple commands, setup transaction speed as well as review card response. Command list consists of standard commands and transparent commands. The commands in this tag are actually realized from transparent command of ISO15693 and ISO14443B. First of all, user shall set up speed in transaction which can either be ISO15693 or ISO14443B. The options are shown in Figure 126.

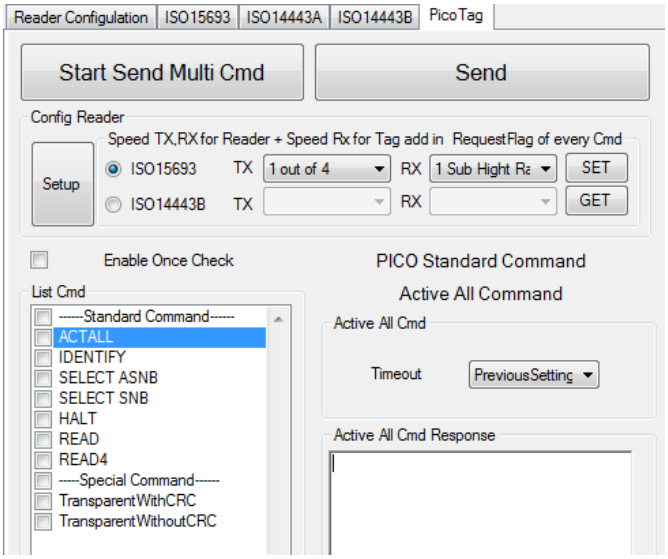


Figure 125 Pico Tag command tap

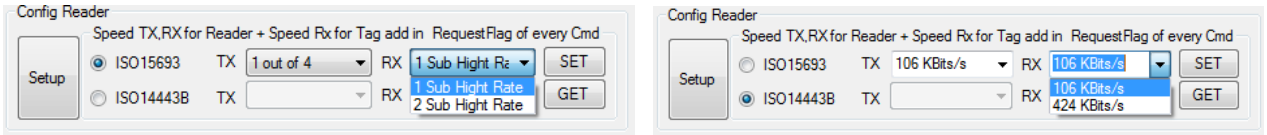


Figure 126 Speed setup for PICO Tag

6.4.1 Pico Tag standard commands

The standard commands used in PicoTag/PicoPass are illustrated in Figure 127, showing relation of each command in card-state transition diagram. Hence, each command shall be applied at appropriate state. The commands in this section are **ACTALL, IDENTIFY, SELECT ASNB, SELECT SNB, HALT, READ** and **READ4**. For more information about the commands, please refer to PicoTag/PicoPass datasheet.

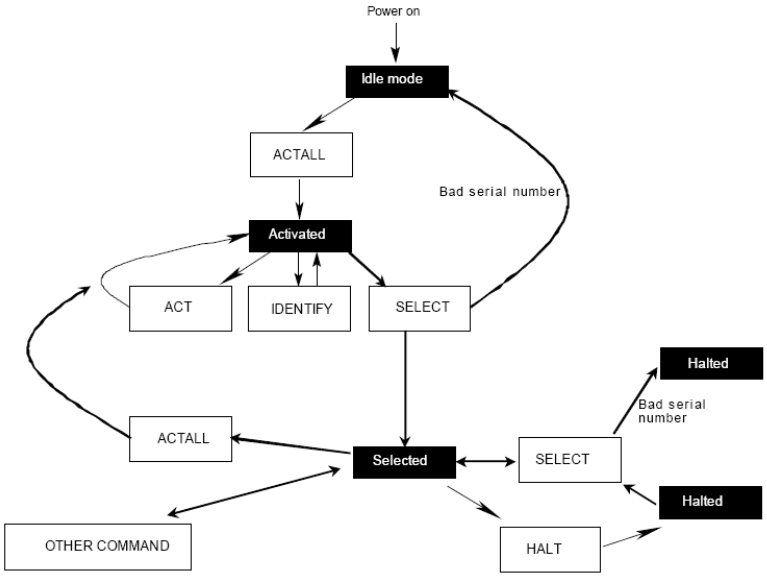


Figure 127 State diagram of PicoTag / PicoPass

6.4.1.1 ACTALL

The **ACTALL** command GUI is shown Figure 128. This command is used to switch card state from Idle to Activated. This command requires no input. Because response for this command is only SOF of standard we select, then the PI931xx report "E1" response error for ISO15693 and "E0" not response for ISO14443B. This is due to the SOF only is treated to be incomplete frame. Please don't care the answer in this state please go to next command.

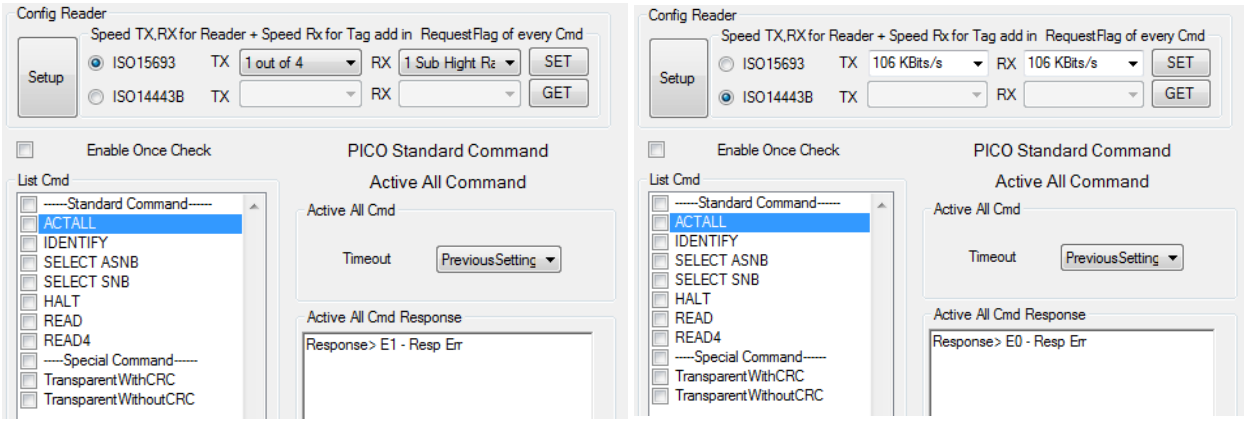


Figure 128 **ACTALL** command and response

6.4.1.2 IDENTIFY

The **IDENTIFY** command is used to get the ASNB (Anti-collision serial number). The card/tag that be able to receive this command shall be in activated state. Result from this command is shown in Figure 129.

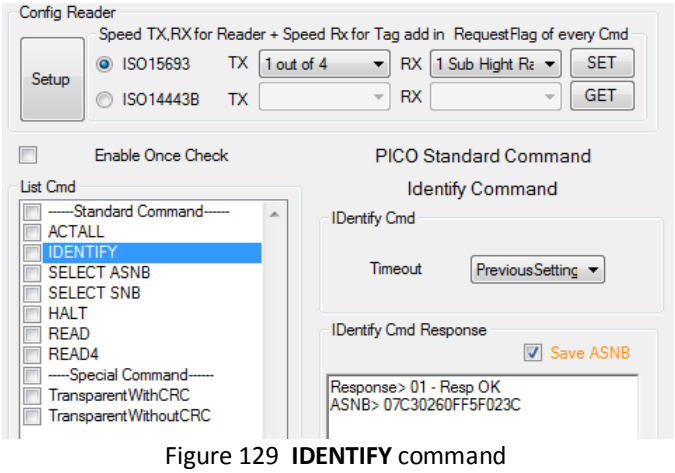


Figure 129 **IDENTIFY** command

6.4.1.3 SELECT ASNB

The **SELECT ASNB** is used to get the real UID. The required parameter is ASNB from **IDENTIFY** command. If PICOtags receive this command correctly, command **READ** and **READ4** will be operable.

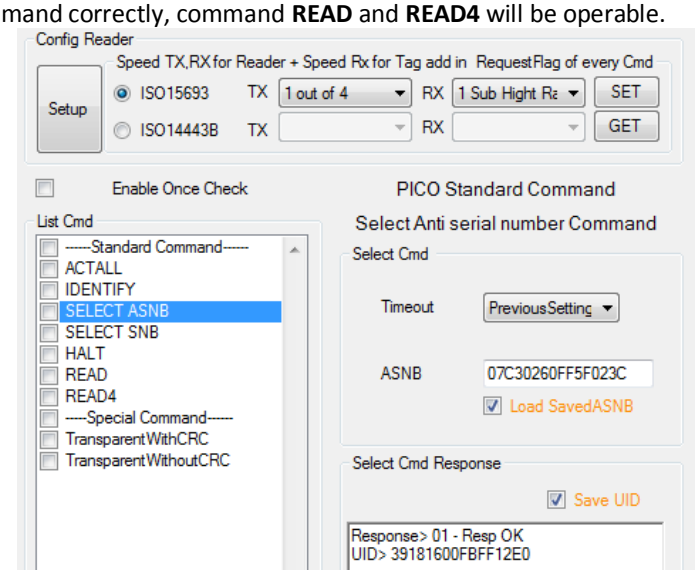


Figure 130 **SELECT ASNB** command

6.4.1.4 SELECT SNB

The **SELECT SNB** is used to get the real UID. The required parameter is UID from **IDENTIFY** command. If PICOtags receive this command correctly, command **READ** and **READ4** will be operable.

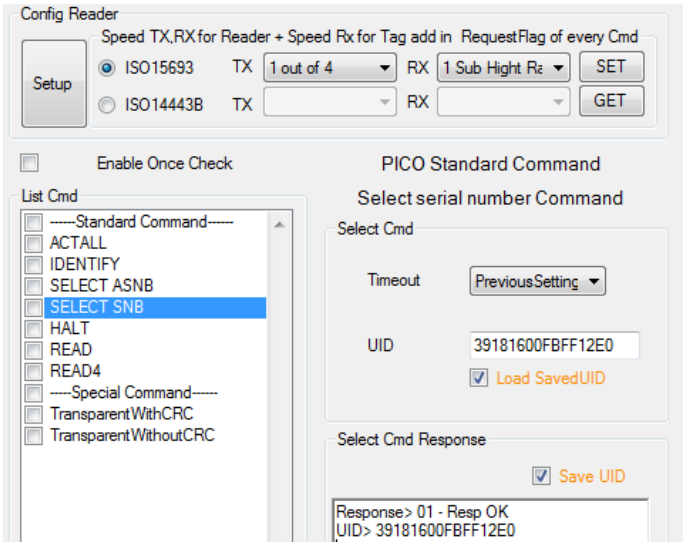


Figure 131 **SELECT SNB** command

6.4.1.5 HALT

The **HALT** command is used to put card in HALT state. In this state, card is silent and waits until receive **SELECT ASNB** and **SELECT SNB** again. Similar to **ACTALL** command, the response from this command is SOF only. Then, it is treated to be incomplete frame. Pi931 respond “E0” and “E1” for ISO14443B and ISO15693 respectably. The **HALT** command and response is shown in Figure 132.

6.4.1.1 READ

The **READ** command is used to read data from specified block in card’s memory. Card must prior be set in selected state. The block address is required parameter. However, for some family such as PicoPass, content in memory is protected by proprietary encryption. Hence, reading memory block apart from UID and configuration (Block 0 and Block1) results in all “FF”. Pi931 can not perform authentication for PicoPass. Example of READ command result is shown in Figure 133.

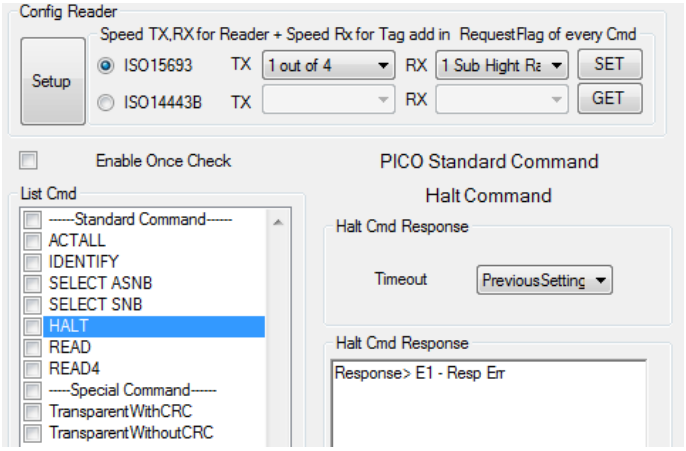


Figure 132 **HALT** command

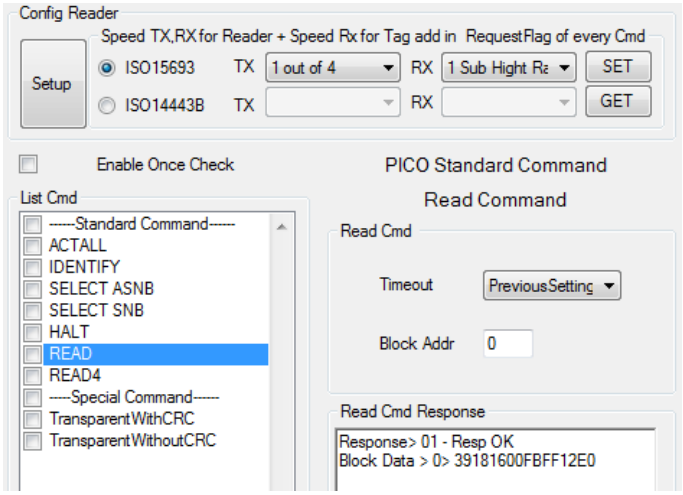


Figure 133 **READ** command

6.4.1.1 READ4

The **READ4** command is similar to **READ** but card answers 4 consecutive block. The required input parameter is starting block address. Example of READ command result is shown in Figure 134.

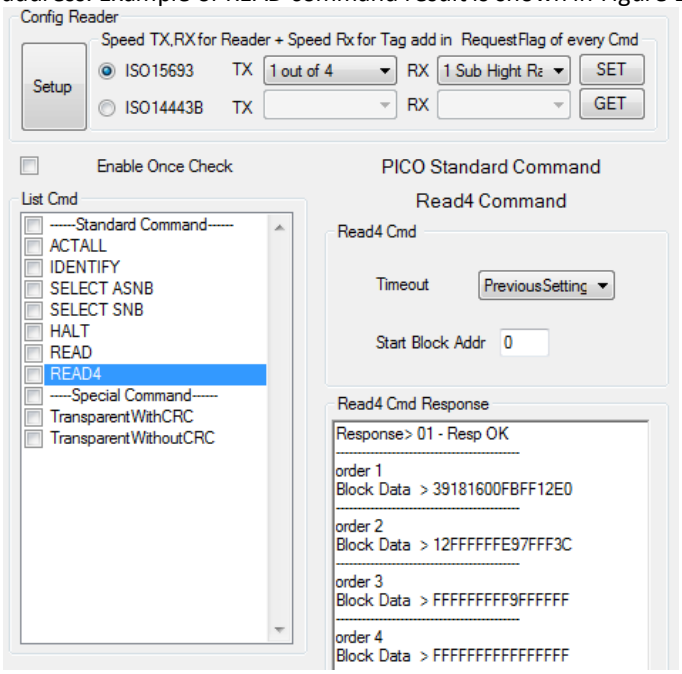


Figure 134 **READ4** command

6.4.2 Special command

TransparentWithCRC, **TransparentWithoutCRC** are provided to support unavailable command in command list. To use these command, user have to know command code as stated in datasheet.

6.5 Felica

Felica is contactless smart card IC from SONY. Line coding of Felica is BPSK both transmission and reception. For more information about Felica frame format, please refer to ISO18092. Note that coding and decoding demonstrates in this section is not performed by the SIC9310 reader IC but it is implemented through raw signal decoding by microcontroller. Due to proprietary encryption, user cannot access Read/Write except UID. Figure 135 shows GUI in Felica tap. In this tap, user can read UID and transmit arbitrary command via C_TransparentWithCRC. First of all, user shall set up speed in transaction to be 212 kbps.

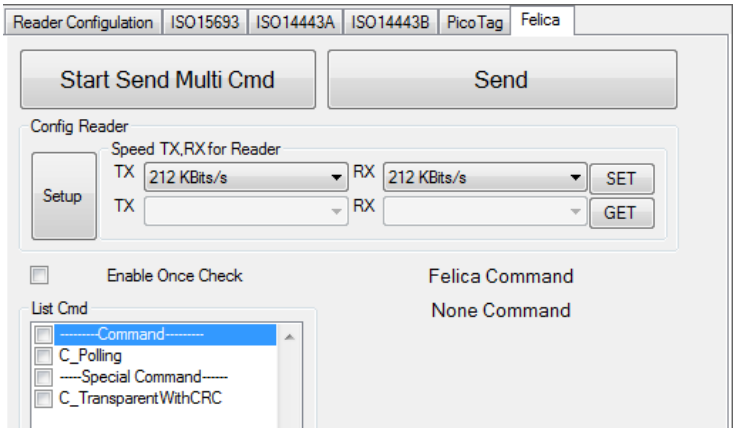


Figure 135 Felica command tap

6.5.1 Polling command

The **C_Polling** command is used to get UID from Felica card. This command can perform reading both single tag and multiple tags in case of anti-collision. The required parameters are system code, reserved and the number of slot in anti-collision process. Figure 136 shows reading a Felica card in 1 slot while Figure 137 shows reading multiple cards in 16 slots. Note that if there are slots that collision occurs, PI931 reports an error code (such as “E0”, “E1”) depending on signal characteristic at that time.

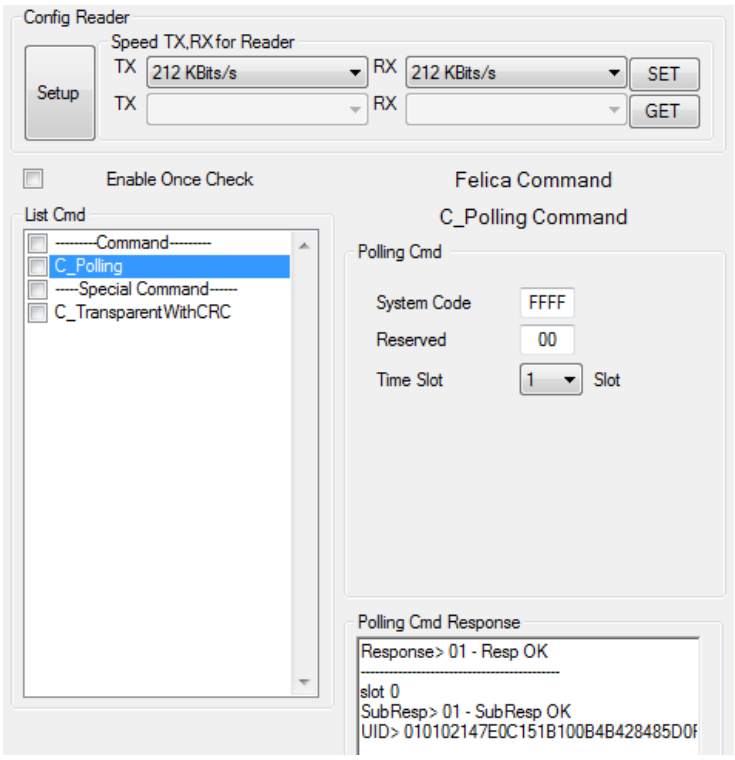


Figure 136 **C_Polling** command in reading a Felica card

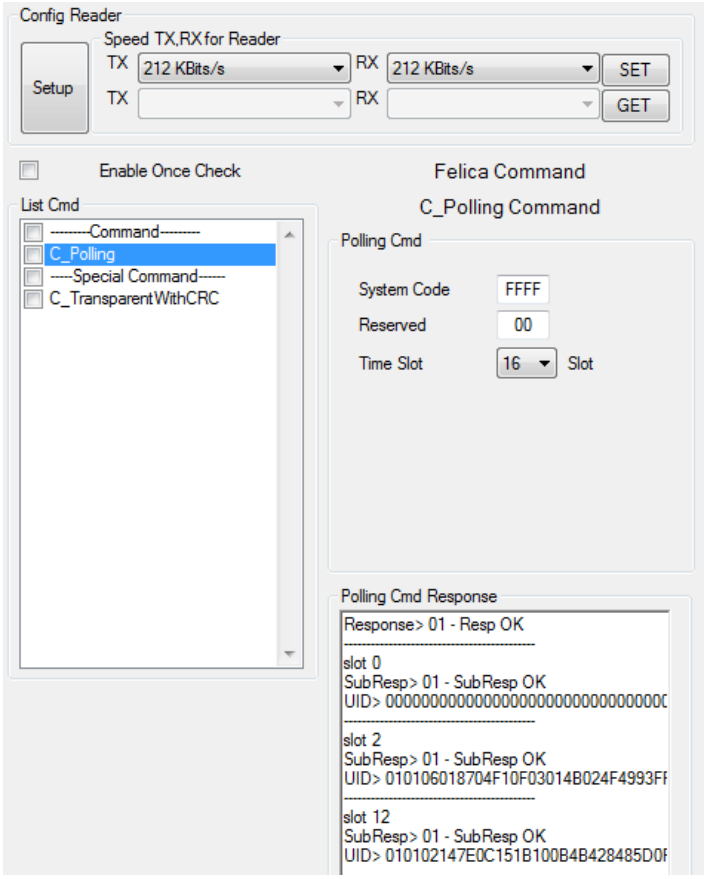


Figure 137 **C_Polling** command in reading multiple Felica card in 16 slots

6.5.2 TransparentWithCRC

The **TransparentWithCRC** command is used to transmit command in hexadecimal code directly to air. Two byte calculated CRC following Felica standard are appended at the end of transmit data frame. For more information, please refer to commands data format in Felica card manual for constructing data bytes.