# TRF7960EVM

# User's Guide

Literature Number: SLOU192B
November 2006–Revised June 2007

Texas Instruments

# Contents

# List of Figures

Tag-it is a trademark of Texas Instruments.

# Read Me First

This manual is written to provide information about the TRF7960 evaluation module. The user should keep in mind the following points.

- It is recommended that the user initially review the data sheet of the device under test.
- To better understand the TRF7960 EVM, it is recommended to review the schematic and layout files.

## About This Manual

### *Conventions*

The following pictograms and designations are used in this manual:

| | |
|---|---|
|  | **WARNING:**<br>**A WARNING IS USED WHERE CARE MUST BE TAKEN, OR A CERTAIN PROCEDURE MUST BE FOLLOWED, IN ORDER TO PREVENT INJURY OR HARM TO YOUR HEALTH.** |
|  | **CAUTION:**<br>**This indicates information on conditions which must be met, or a procedure which must be followed. Failure to observe a caution could cause permanent damage to the system.** |
|  | **Note:**<br>Indicates conditions which must be met or procedures which must be followed to ensure proper system function. |
|  | **Information:**<br>Indicates conditions or procedures that should be followed to ensure optimal function of the system. |

### *If You Need Assistance*

Application Centers are located in Europe, North and South America, the Far East, and Australia to provide direct engineering support. For more information, please contact your nearest TI Sales and Application Center. The contact addresses can be found on our home page: http://focus.ti.com/docs/toolsw/folders/print/trf7960evm.html.

## Numerical Representations

Extensive use is made in this user's guide of the hexadecimal numbering system when describing bytes transmitted and received. The following table is included for your reference:

| Decimal (base 10) | Hexadecimal (base 16) | Binary (base 2) | | Decimal (base 10) | Hexadecimal (base 16) | Binary (base 2) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0000 | | 8 | 8 | 1000 |
| 1 | 1 | 0001 | | 9 | 9 | 1001 |
| 2 | 2 | 0010 | | 10 | A | 1010 |
| 3 | 3 | 0011 | | 11 | B | 1011 |
| 4 | 4 | 0100 | | 12 | C | 1100 |
| 5 | 5 | 0101 | | 13 | D | 1101 |
| 6 | 6 | 0110 | | 14 | E | 1110 |
| 7 | 7 | 0111 | | 15 | F | 1111 |

## Disclaimer

Please note that the enclosed demonstration boards are experimental printed circuit boards and are therefore only intended for device demonstration and evaluation.

The circuit boards have been manufactured by one or more of Texas Instruments' external subcontractors which may not be production qualified.

Device parameters that are measured with these circuit boards may not be representative of production devices or typical production data. Texas Instruments does not represent or guarantee that a final hardware version will be made available after device evaluation.

THE DEMONSTRATION CIRCUIT BOARDS ARE SUPPLIED WITHOUT WARRANTY OF ANY KIND, EXPRESSED, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEXAS INSTRUMENTS ACCEPTS NO LIABILITY WHATSOEVER ARISING AS A RESULT OF THE USE OF THESE CIRCUIT BOARDS.

The fee associated with the demonstration boards is a nonrecurring engineering fee (NRE) to partially defray the engineering costs associated with circuit board development and applications support for the integrated circuit semiconductor product(s). The circuit board is a tool for demonstrating and evaluating the RF semiconductors supplied by Texas Instruments. The demonstration board is supplied to prospective customers to provide services and software that will help them to evaluate the RF semiconductors.

The demonstration board may be operated only for product demonstration or evaluation purposes and then only in nonresidential areas. Texas Instruments' understanding is that the customer's products using the RF parts listed shall be designed to comply with all applicable FCC and appropriate regulatory agency requirements and will, upon testing, comply with these requirements.

Operation of this device is subject to the conditions that it does not cause harmful interference and that it must accept any interference.

## Related Documentation

- *TRF7960/61 Multi-Standard Fully Integrated 13.56-MHz Radio Frequency Identification (RFID) Analog Front End and Data Framing Reader System* data sheet (SLOS504)
- ISO/IEC 15693-2:2000(E) Air Interface and Initialization
- ISO/IEC FDIS 15693-3:2000(E) Anticollision and transmission protocol
- ISO/IEC 14443-2:2001(E) Radio Frequency power and signal interface
- ISO/IEC FDIS 14443-3:2000(E) Initialization and anticollision
- 11-09-21-052 *Tag-it™ HF-I Pro Transponder Chip/Inlays Extended Commands and Options* reference guide (SCBU003)
- 11-09-21-053 *Tag-it™ HF-I Plus Transponder Inlays* reference guide (SCBU004)
- *Tag-it™ Transponder Protocol* reference manual (SCBU032)
- *Electrostatic Discharge (ESD)* application report (SSYA008)

## Trademarks

Tag-it is a trademark of Texas Instruments.

Other trademarks are the property of their respective owners.

# Introduction and Description

The Texas Instruments TRF7960 evaluation module (EVM) helps designers evaluate the performance of the TRF7960 multiple-protocol RFID transceiver.

This manual includes a list of EVM features, a brief description of the module, EVM specifications, details on connecting and using the EVM, and a discussion of the software interface for the EVM. The EVM is used to demonstrate the capabilities of the device (32 pin QFN) and help aid the user in the development process. The device incorporates an analog front end, protocol handling, framing, error checking, and multiple integrated voltage regulators with other features that allow the reader to be customized/configurable for the end application.

## 1.1 Description

The TRF7960 EVM features include:

- Support for the ISO 15693 standard
- Support for both the ISO 14443A standard and the ISO 14443B standard (up to layer 4)
- Support for the Texas Instruments Tag-it™ standard
- Self contained – has an on-board 13.56-MHz loop antenna and interface
- Communication with host software on a Windows-based PC through a standard USB cable
- Protocol indication LEDs – (stand alone mode) required to indicate detection of a tag.

The **TRF7960EVM Rev. A** has the following additional hardware features:

- Supports both PARALLEL and SPI communication interfaces between the TRF7960 and the MSP430 on-board (configurable using an on-board jumper setting)
- A faster and lower-power MSP430 on board. The TRF7960EVM (Rev. A, Figure 2-2) uses the state-of-the art MSP430F2370 with maximum speeds up to 16 MHz and is available in a tiny 40-pin QFN package.
- Power-selection jumper

**Note:** The power-selection jumper is used to connect the 5 V coming from the USB bus to VIN of the RFID reader chip. By default, when the EVMs are shipped, this jumper is connected so that when the EVM is plugged into the USB port of a PC, the TRF7960 and the all the associated circuits are powered.

## 1.2 EVM Default Configuration

As delivered, the EVM is a fully functional reader when plugged into a USB port. To evaluate the TRF7960, a graphical user interface may be installed on a host PC. A USB driver is required to allow communications from a host PC (see Section 3.1, *Software Installations*).

# Using the EVM

This section describes how to connect the EVM to the host computer. It is recommended that the user connect the EVM as described in this section to avoid damage to the EVM or the TRF7960 installed on the board.

## 2.1 Unpacking

Carefully remove the EVM and accessories from the box. The box should contain:

- EVM board (in ESD packaging):
- This manual (check the Web for the latest downloadable version of this manual SLOU192).

**CAUTION:**

**This EVM contains components that can potentially be damaged by electrostatic discharge. Always transport and store the EVM in its supplied ESD bag when not in use. Handle using an antistatic wristband. Operate on an antistatic work surface. For more information on proper handling, see the *Electrostatic Discharge (ESD)* application report,** SSYA008.

## 2.2 Hardware Description

The TRF7960 EVM has the following hardware features:

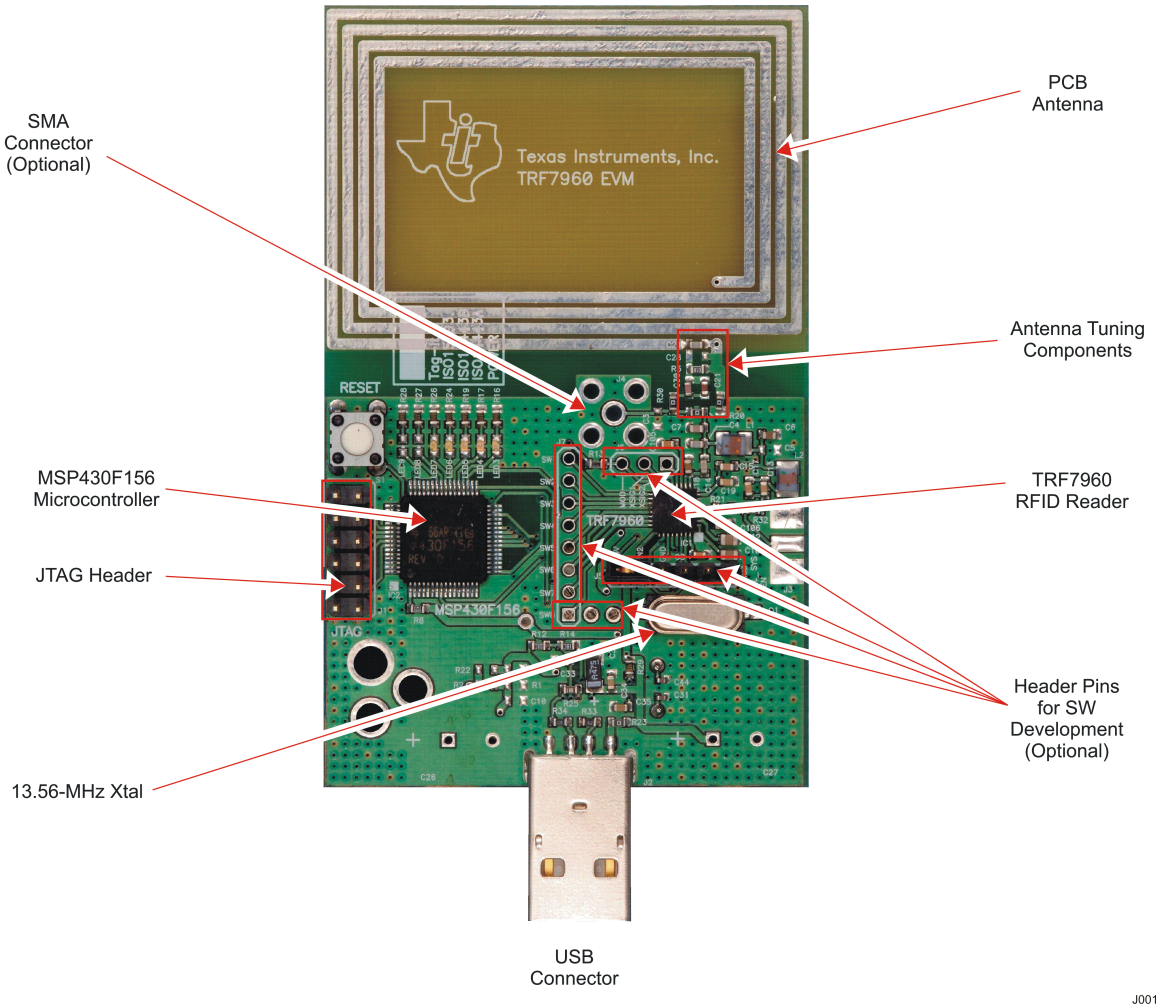



**Figure 2-1. Original Version of EVM**

- RFID interface circuit
- Reset switch
- USB interface
- Status and protocol indicator LEDs
- Onboard loop antenna
- JTAG port

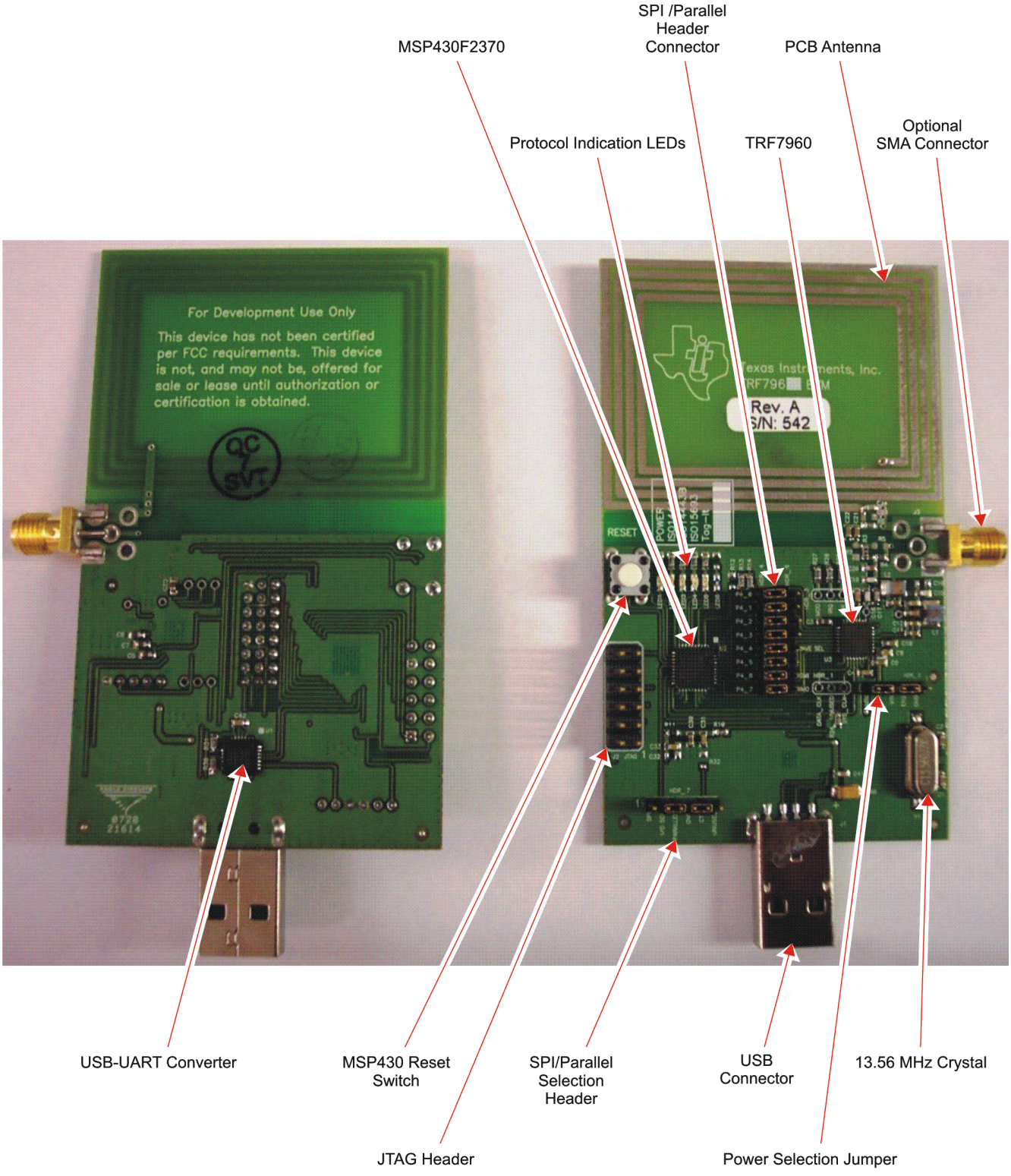



**Figure 2-2. Revision A of EVM**

### Switching Between SPI and Parallel Modes

By default, the TRF7960 EVM is shipped with jumpers installed for parallel communication between the TRF7960 and MSP430. These jumper positions must be changed to enable SPI communication. To enable SPI mode, connect pins **I/O SEL** and **SPI** using a jumper. These pins are found on the SPI/parallel selection header as shown in Figure 2-3. The second step is to replace all eight jumper positions from parallel mode (left) to SPI mode (right) as shown in Figure 2-3.

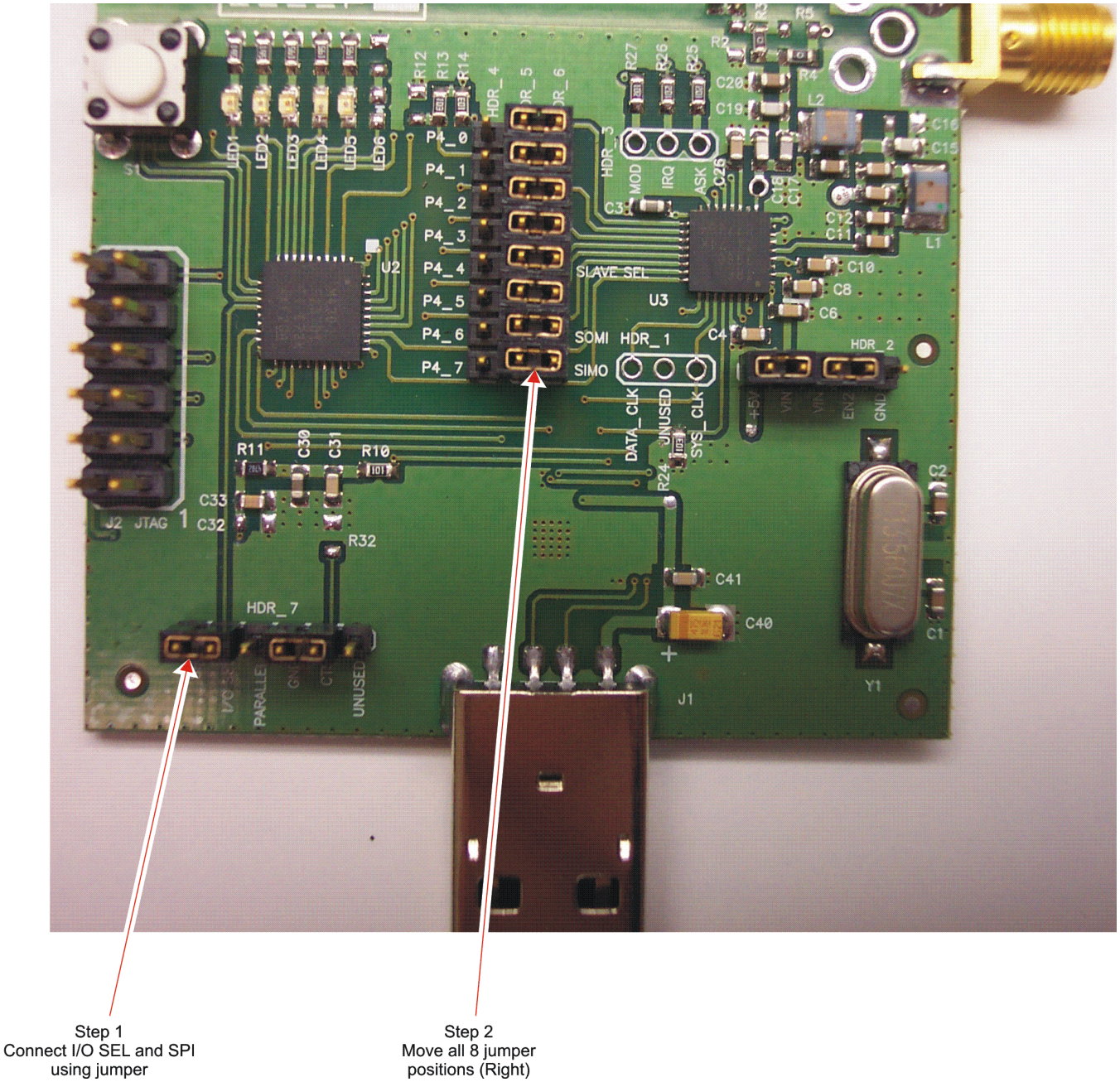



**Figure 2-3. Configuring the TRF7960 EVM for SPI Mode**

## 2.3 Connection to a Host PC

Connect the EVM to a host PC. A USB extension cable may be used, if desired. When using a graphical user interface, the ISO LEDs located on the EVM are inoperative. The ISO LEDs are only operative when the EVM is not being controlled by a host PC.

**Note:**

The TRF7960 EVM consumes (at 5-VDC operation) 120 mA of current from the USB port of the computer in full-power transmit mode. This slightly exceeds the recommended current draw from a standard USB port, which is 100 mA. An external USB hub may be required if additional power is needed.

# EVM Software

This chapter describes the installation and use of the USB drivers and EVM control program.

## 3.1 Software Installations

Do not plug the EVM into the USB port until instructed to do so. If it is already connected to a USB port, disconnect it now.

Download the USB driver and graphical user interface (GUI) software from the Web site http://focus.ti.com/docs/toolsw/folders/print/trf7960evm.html and save to a folder. Software installation is a two-step process. The first step is the installation of a third-party virtual COM port (VCP) driver, and the second part is the installation of the EVM GUI (TI proprietary).

**Note:** For the **Rev. A** version of the EVM, follow the instructions in Section 3.2 for installing USB driver and GUI. The instructions in Section 3.1 only apply to the original version of the EVM.

**Note:**

Always check the Web site http://focus.ti.com/docs/toolsw/folders/print/trf7960evm.html for the latest software and documents.

### 3.1.1 Virtual COM Port Driver Installation

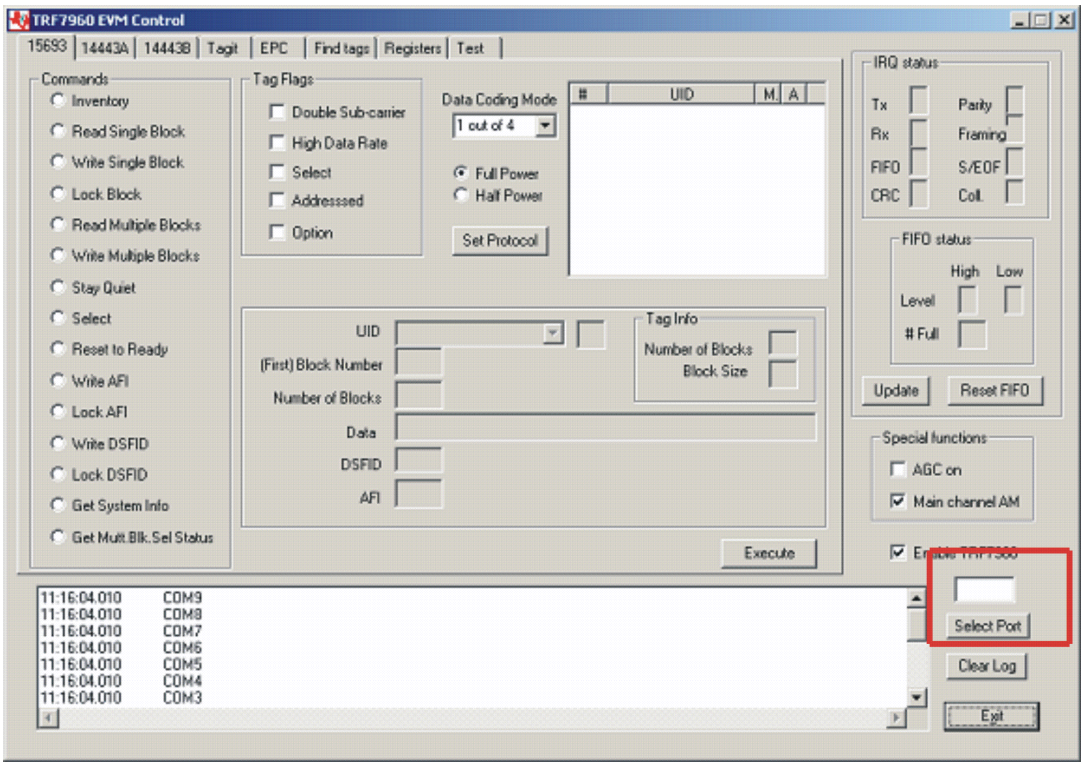
To install the virtual driver, run the program CDM_setup.exe. When the driver installation is complete, the following confirmation is displayed:


### 3.1.2 Hardware Installation

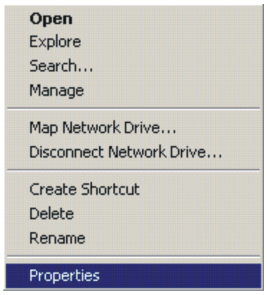
At this point, attach the EVM to an open USB port. The EVM can be plugged directly into the port or attached at the end of a USB extension cable (type A, not supplied). At this point, the power LED should be lit. Any RFID tag corresponding to a supported protocol can be detected and is indicated by the corresponding LED.

### 3.1.3 Software GUI Installation

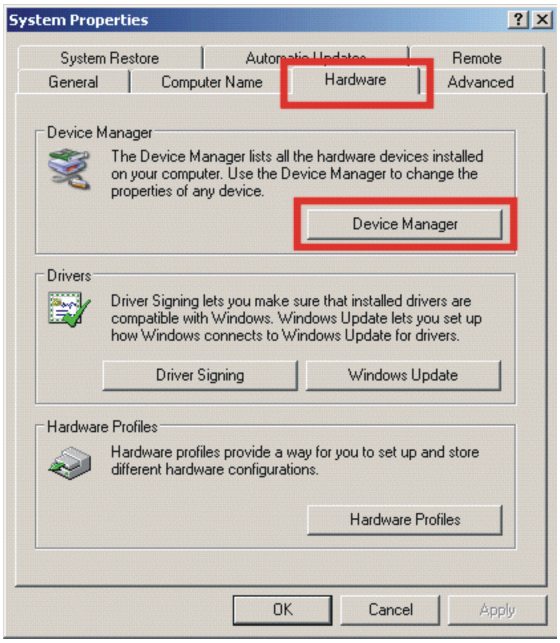
The software GUI is the file named TRF7960EVM_GUI_V1.1.zip. It can be unzipped using a standard unzip program and is a self-contained executable. Create a folder where desired on the host PC, and unzip the executable into that folder. The program can be run from the folder, or a shortcut can be created and placed on the desktop of the host computer. In most cases, the program automatically detects the COM port. In case the program could not detect the COM port, enter the COM port number (e.g., COM3) in the *Select Port* window at the bottom right of the GUI as shown following, and click on the Select Port button).



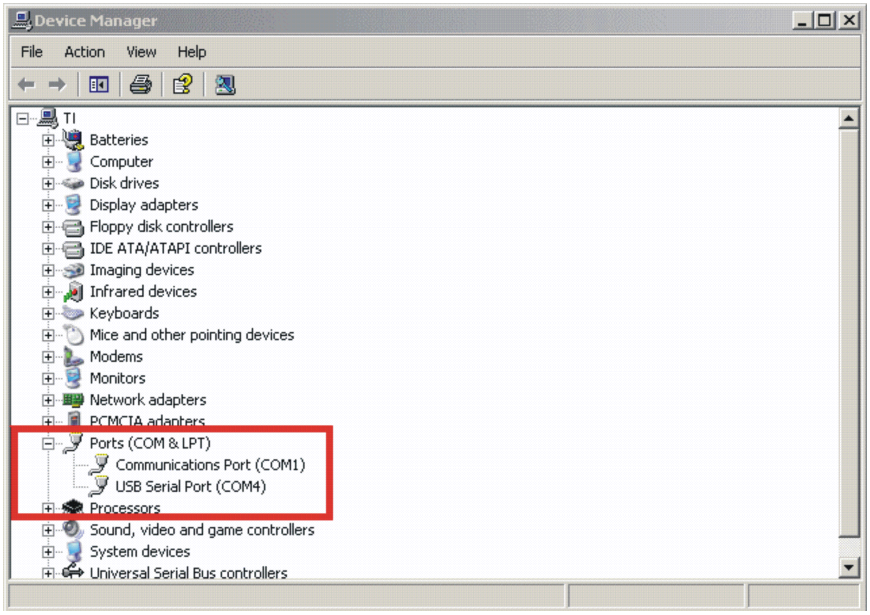

To determine the USB serial port that corresponds to the EVM, right-click on the *My Computer* icon on the desktop. When the drop-down menu appears, click on *Properties*.




On the properties window, select the *Hardware* tab:




Next, click on *Device Manager*, then click the + sign next to *Ports* to expand the ports:




If the driver installation was successful and the EVM is plugged in, *USB Serial Port* should appear in the list of ports, followed by a port number (in this example, COM4). The actual port number may be different. Make note of the COM port number and enter it in the *Select Port* window of the GUI. Then select the *Select Port* on GUI (do not press the *Enter* key). Note: If the *Enter* key is pressed, the program ends and the GUI closes.

**Note:**

Running the GUI disables the protocol LEDs on the EVM. LED operation can only be restored by exiting the GUI, pushing the reset button on the EVM, or cycling power.

## 3.2 Software Installation for Rev. A EVM

Follow the steps in the following sections for the Rev. A version of the EVM.

### 3.2.1 USB Driver Installation

Do not plug the EVM into the USB port until instructed to do so. If it is already connected to a USB port, disconnect it now.

Download the Silicon Labs USB-UART bridge VCP and graphical user interface (GUI) software from the Web site http://focus.ti.com/docs/toolsw/folders/print/trf7960evm.html and save to a folder. The USB VCP driver is named USB_DRIVER_SETUP_Silabs.zip

Software installation is a two-step process. The first step is the installation of a Silicon Labs virtual COM port (VCP) driver, and the second part is the installation of the EVM GUI (TI proprietary).

### 3.2.2 Virtual COM Port Driver Installation

The Silicon Labs USB-UART bridge VCP driver can be downloaded from the TRF7960EVM Web site mentioned previously, or directly from the Silicon Labs Web site at the following address.

http://www.silabs.com/tgwWebApp/public/web_content/products/Microcontrollers/USB/en/mcu_vcp.htm

The driver installation and setup is a two-step process.

1. Extraction
   Initial software setup requires running CP210x_Drivers.exe to extract all of the device drivers (Windows and Macintosh). After following the prompts, the utility copies the driver files to a specified directory or the default directory, “C:\SiLabs\MCU\CP210x”. Each set of drivers is extracted to an appropriately named directory, for example, WIN and MACX.
2. Installation
   Follow these steps to install the Windows XP VCOM driver:
   a. Connect the USB cable between the host computer and the TRF7960 EVM.
   b. Windows opens a *Found New Hardware Wizard* window.
   c. Select "Install from a list or specific location (Advanced)" and press Next.
   d. Select "Include this location in the search".
   e. Press Browse to locate the "C:\SiLabs\MCU\CP210x\WIN" directory. Once this directory is selected, press OK.
   f. Verify that the correct path and filename are shown and press Next.
   g. Press Finish to finish installing the "CP210x USB Composite Device".
   h. Windows opens a second “Found New Hardware Wizard” window.
   i. Select "Install from a list or specific location (Advanced)" and press Next.
   j. Select "Include this location in the search".
   k. Press Browse to locate the "C:\SiLabs\MCU\CP210x\WIN" directory. Once this directory is selected, press OK.
   l. Verify that the correct path and filename are shown and press Next.
   m. Press Finish to finish installing the "CP210x USB to UART Bridge Controller".

### 3.2.3 Hardware Installation

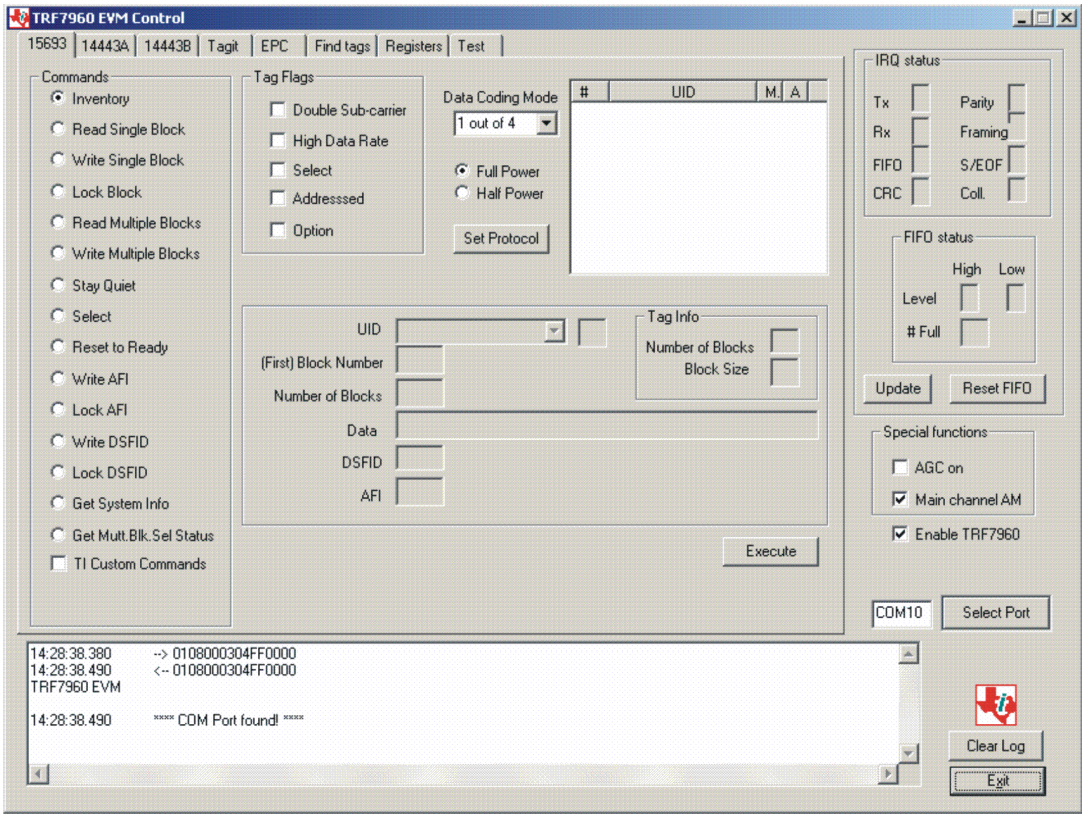
At this point, attach the EVM to an open USB port. The EVM can be plugged directly into the port or attached at the end of a USB extension cable (type A, not supplied). At this point, the power LED should be lit. Any RFID tag corresponding to a supported protocol can be detected and is indicated by the corresponding LED.

### 3.2.4 Software GUI Installation

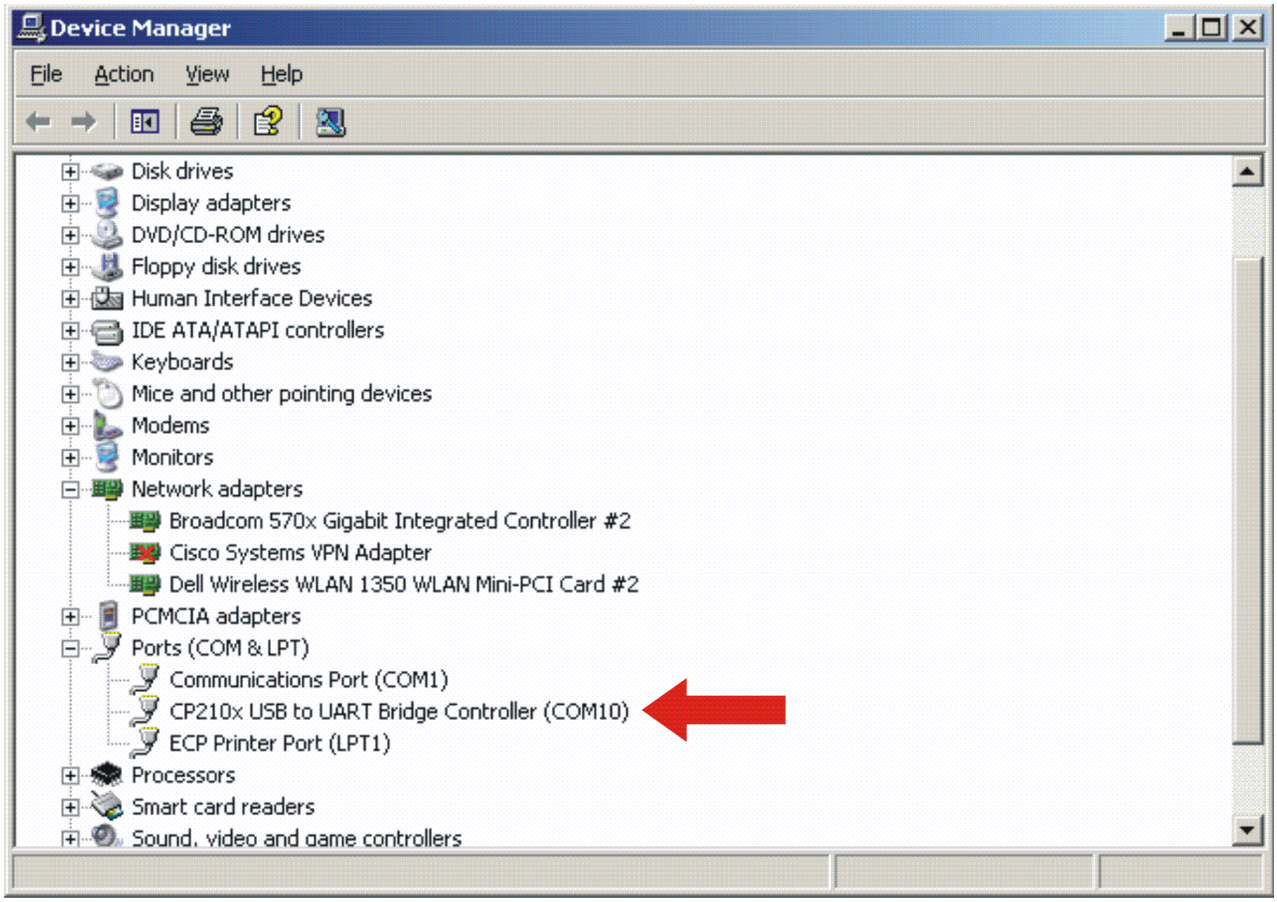
The software GUI is the file named **TRF7960EVM_REVA_GUI_V1.1.zip.** It can be unzipped using a standard unzip program and is a self-contained executable. Create a folder where desired on the host PC and unzip the executable into that folder. The program can be run from the folder, or a shortcut can be created and placed on the desktop of the host computer.

When this software is used with the TRF7960 EVM (Rev. A), the program automatically detects the COM port. The selected COM port is automatically displayed in the text box next to the Select Port button.

In case the program could not detect the COM port, enter the COM port number (e.g., COM3) in the Select Port window at the bottom right of the GUI as shown following, and click on the Select Port button).




Next, click on Device Manager, then click the + sign next to ports to expand the ports:



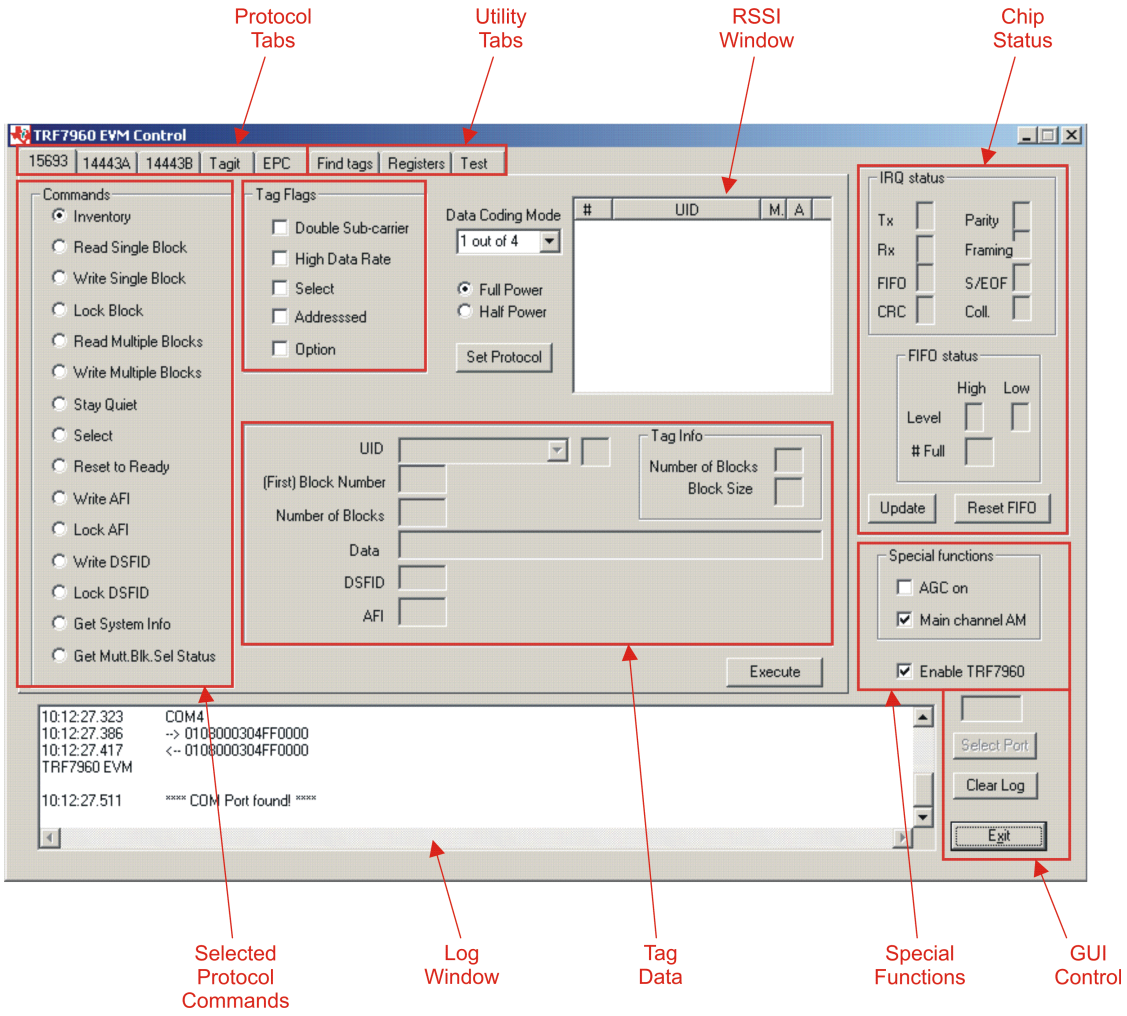

If the driver installation was successful and the EVM is plugged in, CP210x USB to UART Bridge Controller should appear in the list of ports, followed by a port number (in this example, COM10). The actual port number may be different.

If automatic detection does not take place, make note of the COM port number and enter it in the Select Port window of the GUI. Then select the Select Port on GUI (**do not press the Enter key**).

**Note:** If the Enter key is pressed the program ends and the GUI closes.

## 3.3 Software Interface

The GUI window is shown following. Each section of the window has a different function. The figure shows the arrangement for the different protocols; *Find Tags*, *Registers*, and *Test* radically change the display.




### *3.3.1 Program Control Window (Lower Right-Hand Corner)*

The Select Port window allows the user to enter manually the USB serial port used by the host computer to communicate with the TRF7960 EVM board.

Exit button – exits the TRF7960 control program

### 3.3.2 Protocol Tabs Window

The protocol tabs window selects between tag protocols and program functions. Available options are:

- (ISO/IEC) 15693 – vicinity cards
- (ISO/IEC) 14443A – proximity cards
- (ISO/IEC) 14443B – proximity cards
- Tag-it™ – a proprietary TI protocol

### 3.3.3 Utility Tabs Window

- Find Tags – a function that reads tags of all protocols
- Registers – allows the user to set TRF7960 register values manually
- Test

### 3.3.4 Flags Window

This window allows the user to set flags for the 15693 and Tag-it protocols. Different flags may be available for different commands – see Appendix A.1. The tag window automatically updates available flags depending on the request chosen.

### 3.3.5 Chip Status Window

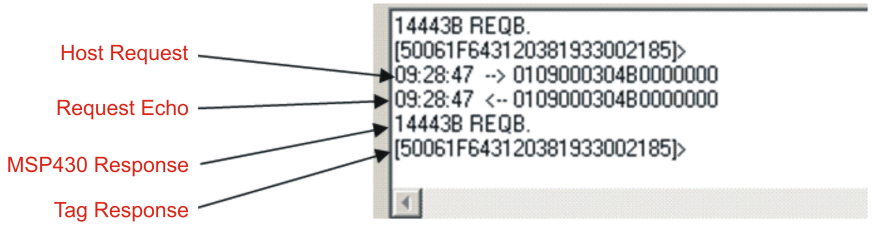
Shows the status of the TRF7960 on the EVM board.

### 3.3.6 Command (Request) Window

This window shows various request options available for each protocol.

### 3.3.7 Log Window

The log window shows all communication frames from host computer to reader board. The tag response is also displayed in the log window. The tag response (register content) is always in parentheses to distinguish it from the host-to-reader data exchange. This information is also stored in the *rfid-reader.log* file, located in the same file directory as GUI.exe, which can be opened by a normal text editor such as Notepad.




### 3.3.8 Tag Data Window

The *Tag Data* window is where the user enters addresses, data, number of bits, and other information required by certain commands. Checking certain flags in the *Flag* window may activate more fields for data entry.

**Note:**

Some tag information appears in grayed out areas of the *Tag Data* window. This data has been read from the tag and formatted for display, but it cannot be changed.

### 3.3.9 RSSI Window

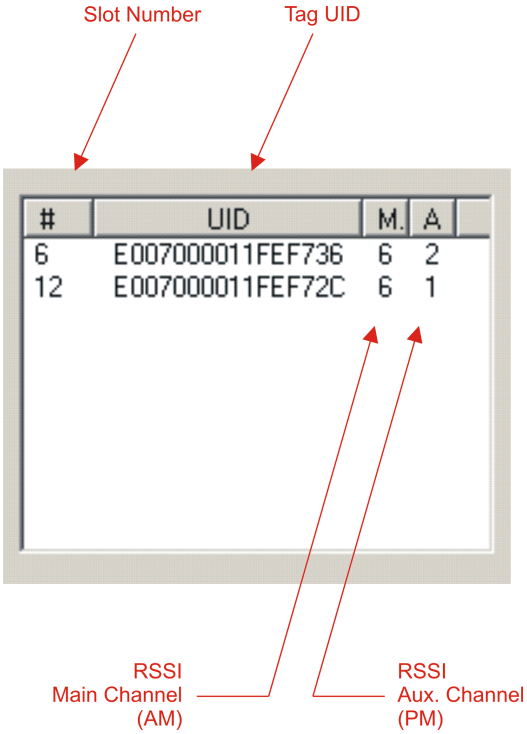
The RSSI field displays the slot number, UID and the RSSI values of the corresponding tag. If there was a collision and the reader performed a second anticollision procedure, the slot numbers are indicated with an additional character:

A = second procedure

B = third procedure

and so on

The main channel, which is AM, is used as the primary one, and PM is the auxiliary channel. The RSSI maximum value is 7 and minimum value is 0. The corresponding RSSI values depend on the system design (antenna + reader), and the levels can vary based on the quality of the reception. The specifics of the corresponding input voltage levels to RSSI levels are defined in the product data sheet.




In the preceding example, one can see that the tags in slots #6 and #12 have a main-channel RSSI value of 6, with auxiliary-channel RSSI values of 2 and 1, respectively.

### 3.3.10 Special Functions Window

Special functions, such as AGC on/off, main channel AM, and enable/disable the TRF7960. The AGC is turned off after the power-on reset (POR) and can be enabled when desired (especially in noisy environments). By default, the input channel is AM and can be switched to PM if the RSSI value for the PM channel is higher than the AM.

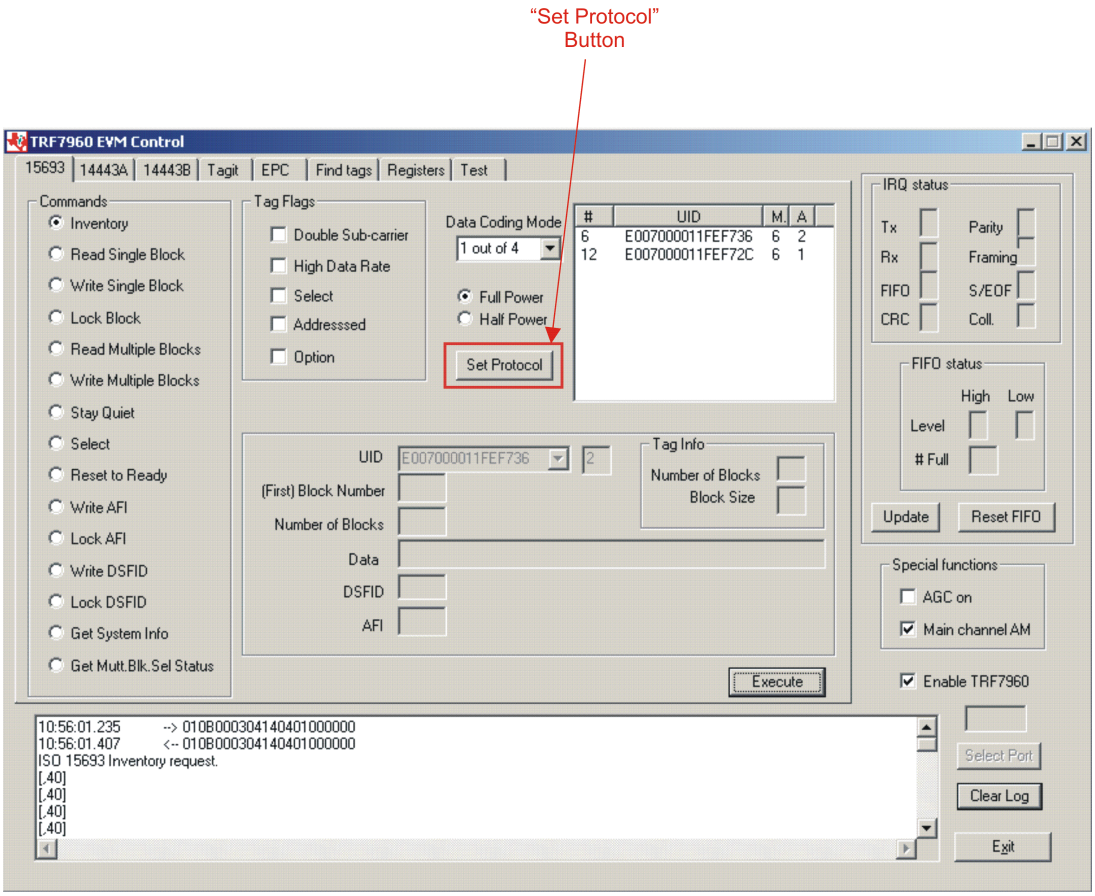
### 3.3.11 Other Functions

Other functions on the main EVM control panel are:

- Set protocol, which configures the program for the selected protocol once the protocol tab has been selected
- Execute button, which processes the selected command
- Power control (half or full), which can be used to simulate marginal reception conditions. The RF output power selection enables the user to switch between full power (200 mW) and half power (100 mW); however, the antenna matching circuit is tuned to operate with full-power selection, and performance is not optimal in half-power selection. This is due to the matching on the output of the reader IC, which currently is matched for 200 mW. (The load impedance for full power is 4 Ω and half power is 8 Ω.)
- Data coding mode, which is used in conjunction with the 15693 protocol

## 3.4 Set Protocol

Selecting a protocol with a protocol tab does not automatically set the program to that protocol. The user must manually click on the *Set Protocol* button:

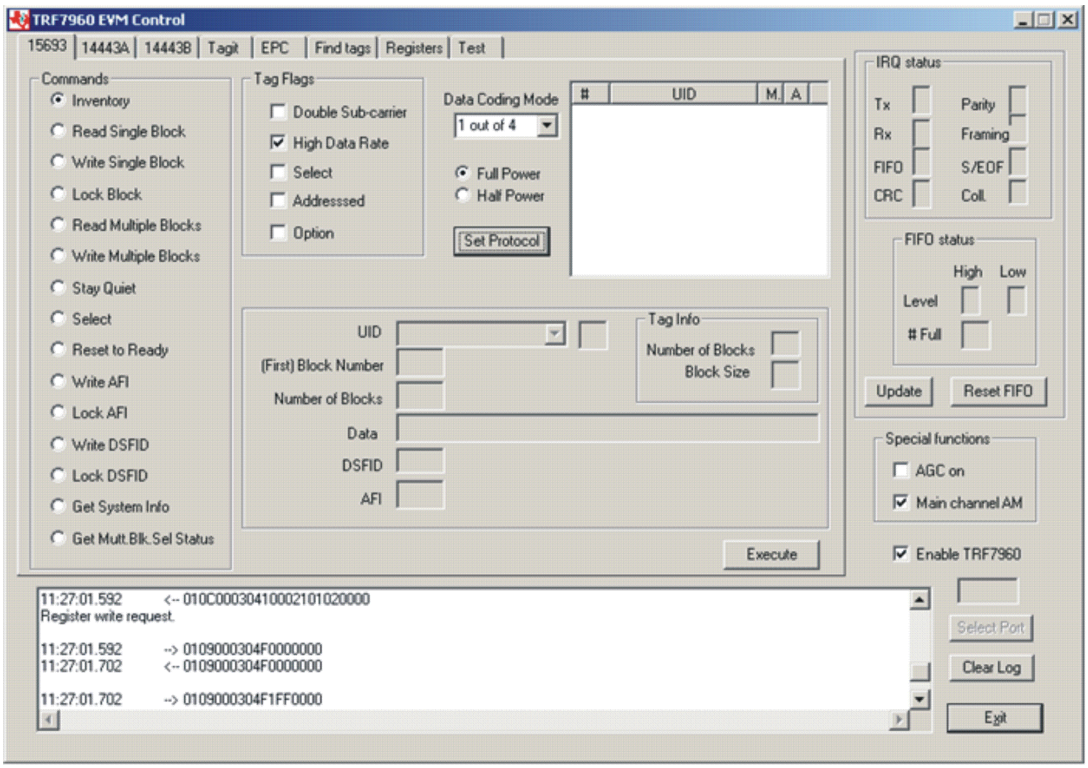



When the *Set Protocol* button is pressed, the software sets the parameters for the corresponding standard. These settings can also be modified through the *Registers* tab in the GUI.

## 3.5 ISO/IEC 15693 Protocol

This section describes commands for the 15693 protocol. After a command has been selected by clicking on the associated command button in the *Commands* window, the user should set any flags as needed (see Section A.1). If appropriate, enter data in the *Tag Data* window.

An ISO15693 set protocol command sends three commands (register write, set AGC, and set receiver mode (AM/PM).




**First Command: Register Write**
**01 0C 00 03 04 10 00 21 01 02 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0C | Packet length = 12 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 10 | Register write |
| Register 00 | 00 21 | In register 00 (chip status control register), write 21 (RF output active, +5VDC operation) |
| Register 01 | 01 02 | In register 01 (ISO control register), Write 02 (set protocol to ISO15693 high bit rate, 26.48 kbps, one subcarrier, 1 out of 4) |
| EOF | 00 00 | End of frame |

### Second Command: Set AGC
### 01 09 00 03 04 F0 00 00 00 (all bytes are continuous; spaces are added for clarity)

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | F0 | AGC toggle |
| AGC Off | 00 | AGC on = FF |
| EOF | 00 00 | End of frame |

### Third Command: Set Receiver Mode
### 01 09 00 03 04 F1 FF 00 00 (all bytes are continuous; spaces are added for clarity)

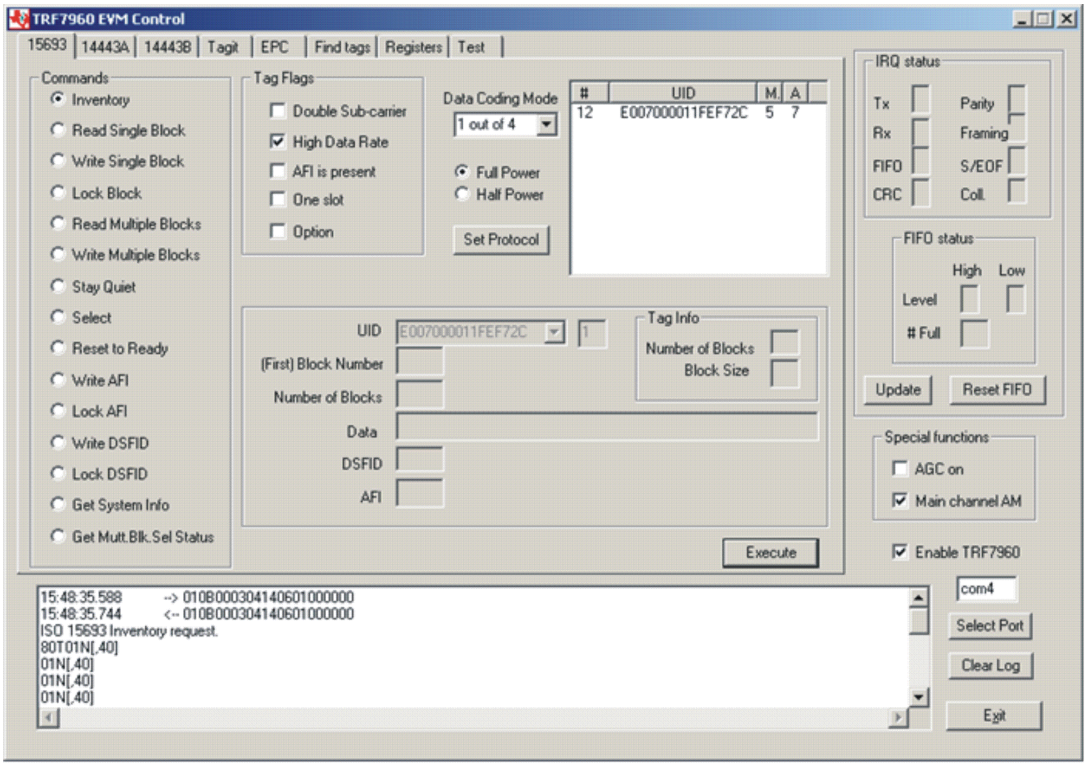
| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | F1 | AM/PM toggle |
| AM | FF | FF = AM, while a 00 = PM |
| EOF | 00 00 | End of frame |

### 3.5.1 Inventory

The *Inventory* command is used to acquire the unique IDs (UID) of ISO15693 tags in the read zone. The two inventory methods supported are 16-slotted and single-slot. A single-slot request allows all transponders in the read zone to reply to the *Inventory* request. In cases where more than one tag is present, such a request would cause a data collision, which in turn causes a reader to send a collision error message to the GUI. A 16-slot inventory sequence decreases the likelihood of a data collision by forcing compliant transponders to respond in 1 of 16 slots, based on a portion of their UIDs. To perform a slotted sequence, the *Slot Marker/End-of-Frame* request is used in conjunction with this command. Any collision that does occur in a slotted sequence can be further arbitrated by using the anticollision mask in an algorithm similar to that outlined in the ISO15693 standard.

To inventory a tag, the user should:

- Click the button for *Inventory* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Click on *Set Protocol*.
- *Execute* the command.




**Information:**

When requesting the 16-slot method, the EVM transmitter remains **ON** in order to preserve the tag states changed by the request.

**Request Packet:**

**01 0B 00 03 04 14 06 01 00 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0B | Packet length = 11 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 14 | Inventory request |
| Flags | 06 | High data rate = 1 |
| Anticollision Cmd | 01 | |
| Mask Length | 00 | |
| EOF | 00 00 | End of frame |

**Inventory Cmd (Tag Response)**

Reader/Tag response (0 through 15 slots) is as follows:

IRQ Status Register [<Tag response if any>, RSSI register value]

Example:

ISO 15693 Inventory request

80T01N[,40] Comment: (slot # 0, **80T** end of transmit, **01N** no response interrupt, **[,40]** < no tag response >, RSSI register status)

01N[,40]

01N[,40]

01N[,40]

01N[,40]

01N[,40]

01N[,40]

01N[,40]

01N[,40]

01N[,40]

01N[,40]

01N[,40]

01N[,40]

60F40E[2CF7FE11000007E0,6F] Comment: (slot # 12, **60F** receive data buffer 75% full, **40E** end of receive, **[2CF7FE11000007E0,6F]** < tag UID in reverse-byte order>, RSSI register status)

01N[,40]

01N[,40]

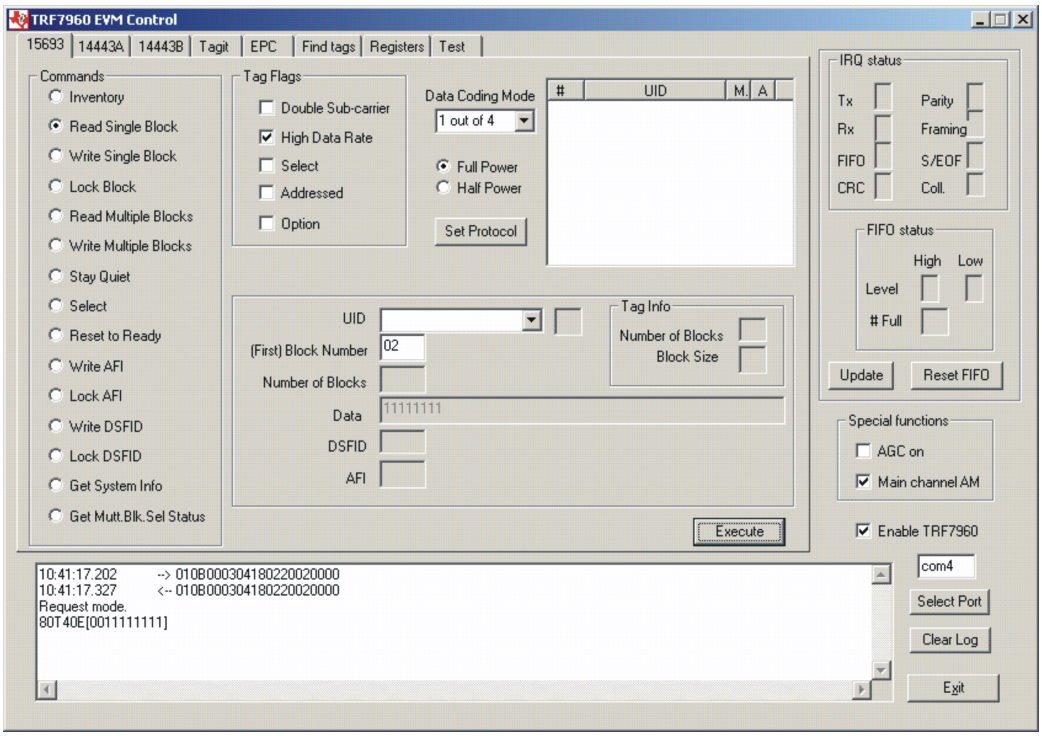
01N[,40] Comment: (slot # 15, **01N** no response interrupt, **[,40]** < no tag response >, RSSI register status)

### 3.5.2 Read Single Block

The Read Single Block command gets the data from one memory block of the responding tag. In addition to this data, a Block Security Status byte can be requested. This byte shows the write-protection of the block specified [e.g., unlocked, (user/factory) locked, etc.].

To read a single block, the user should:

- Click the button for *Read Single Block* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window and set the *Addressed* flag (if only one tag is present, only one choice is available).
- Enter two hex digits corresponding to the block number in the *(First) Block Number* field in the *Tag Data* window.
- *Execute* the command.



**Request Packet:**

**01 0B 00 03 04 18 02 20 02 00 00 (all bytes are continuous; spaces are added for clarity)**
**Note that *Option* flag is disabled.**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0B | Packet length = 11 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request Mode |
| Flags | 02 | Option flag = 0; High Date Rate flag = 1 |
| Read Single Block Cmd | 20 | |
| Selected Block Number | 02 | Note: Reading block 02, which is block #3 |
| EOF | 00 00 | End of frame |

**Read Single Block (Tag Response)**

Request Mode

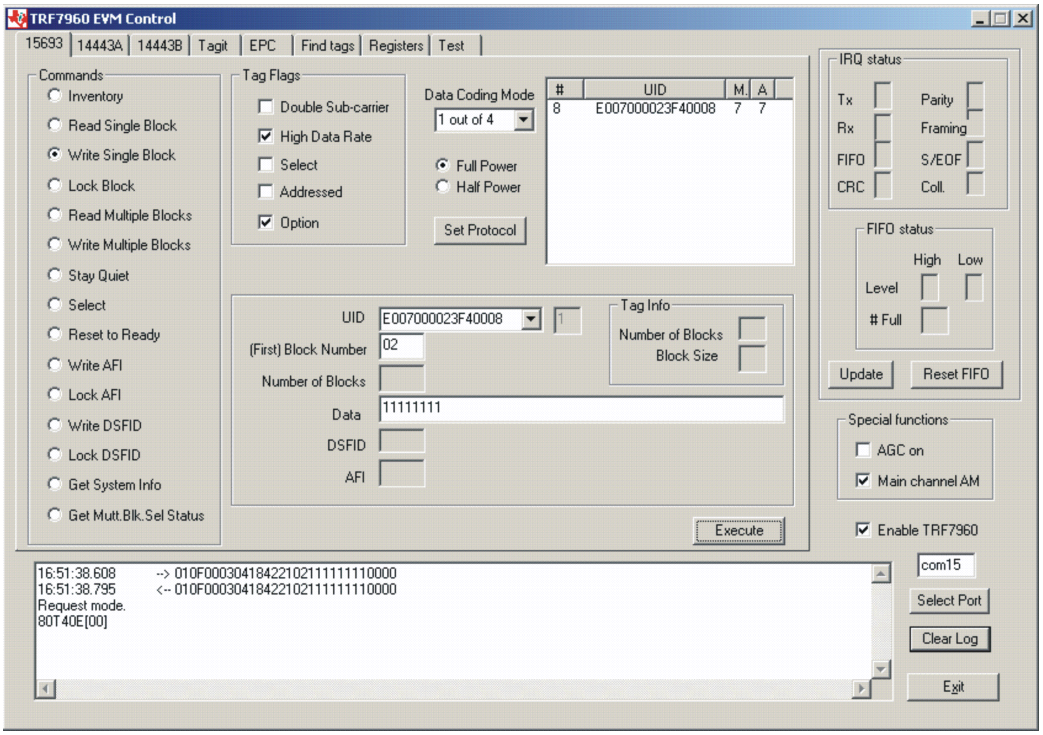
80T40E[0011111111] Comment: (**80T** end of transmit, **40E** end of receive, [**00** no tag error, **11 11 11 11** tag block data, 32 bits])

### 3.5.3 Write Single Block

The Write Single Block request writes data to one memory block of the addressed tag(s). In order to successfully write data, the host must know the size of the memory block of the tag. This information is available through the *Get System Information* request, if supported by the tag. A corrupted response or lack of response from TRF7960 does not necessarily indicate a failure to perform the write operation. Additionally, multiple transponders may process a nonaddressed request.

To write a single block, the user should:

- Click the button for *Write Single Block* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window and set the *Addressed* flag (if only one tag is present, only one choice is available).
- Enter two hex digits corresponding to the block number in the *(First) Block Number* field in the *Tag Data* window.
- Enter 8 hexadecimal digits corresponding to the data to be written in the *Data* field in the *Tag Data* window.
- *Execute* the command.




Note: The *Option* flag (bit 7) of the ISO 15693 defined Request flags must be set to 1 for all Write and Lock commands to respond properly.

**Request Packet:**

**01 0F 00 03 04 18 42 21 02 11 11 11 11 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0F | Packet length = 15 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 42 | Option flag = 1; High Date Rate flag = 1 |
| Write Single Block Cmd | 21 | Write Single Block cmd |
| Selected Block Number | 02 | Note: Write to block 02, which is block #3 |
| Block Data | 11 11 11 11 | 32 bits |
| EOF | 00 00 | End of frame |

**Write Single Block (Tag Response)**

Request Mode

80T40E[00]  Comment: (**80T** end of transmit, **40E** end of receive, [**00**] no tag error)

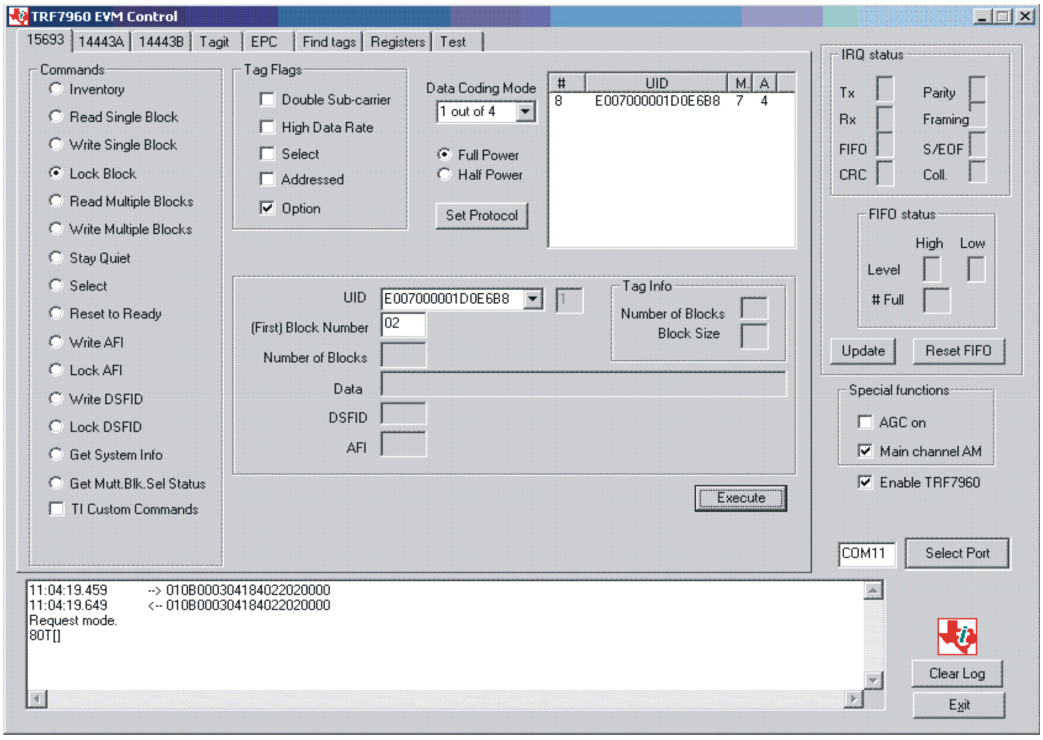
### 3.5.4 Lock Block

The Lock Block command write-protects one memory block of the addressed tag(s). A corrupted response or lack of response from the TRF7960 does not necessarily indicate a failure to perform the lock operation. Additionally, multiple transponders may process a non-addressed request.

Used to permanently lock the requested block

To lock a block, the user should:

- Click the button for *Lock Block* in the *Command* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window and set the *Addressed* flag (if only one tag is present, only one choice is available).
- Enter two hex digits corresponding to the block number in the *(First) Block Number* field in the *Tag Data* window.
- *Execute* the command.




Note: The *Option* flag (bit 7) of the ISO 15693 defined Request flags must be set to 1 for all Write and Lock commands to respond properly.

**Request Packet:**

**01 0B 00 03 04 18 40 22 02 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0B | Packet length = 11 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 40 | Option flag = 1; High Data Rate flag = 0 |
| Lock Block Cmd | 22 | Lock Block cmd (used to permanently lock a selected block) |
| Selected Block Number | 02 | Note: Lock block 02, which is block #3 |
| EOF | 00 00 | End of frame |

**Lock Block (Tag Response)**

Request Mode

80T[] Comment: (**80T** end of transmit, [] no tag response)

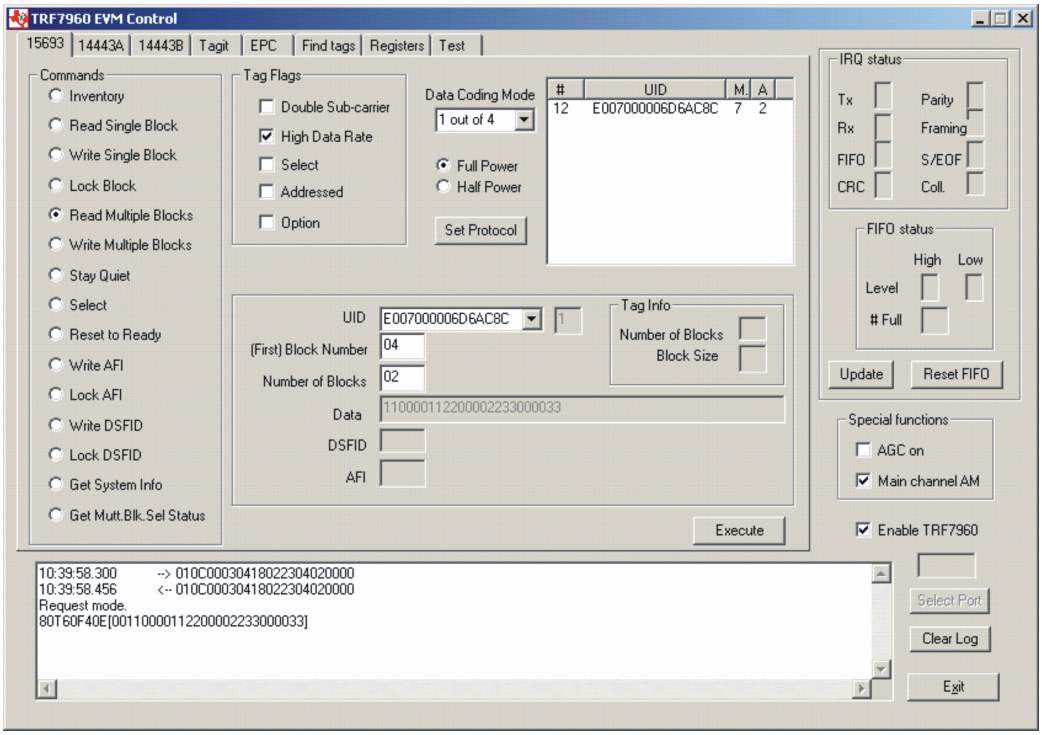
### 3.5.5 Read Multiple Blocks

The Read Multiple Blocks command gets the data from multiple memory blocks of the responding tag. In addition to this data, a Block Security Status byte can be requested for each block. This byte shows the write-protection of the block specified [e.g., unlocked, (user/factory) locked, etc.].

To read multiple a blocks, the user should:

- Click the button for *Read Multiple Blocks* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- Enter two hex digits corresponding to the starting block number in the *(First) Block Number* field in the *Tag Data* window. The blocks are numbered from *00* to *FF* (0 to 255).
- Enter two hex digits corresponding to the number of blocks to be written in the *Number of Blocks* field in the *Tag Data* window. The number of blocks in the request is one less than the number of blocks that the tag returns in its response.

  E.g., a value of *06* in the *Number of Blocks* field requests to read 7 blocks. A value of *00* requests to read a single block.
- *Execute* the command.



**Request Packet:**

**01 0C 00 03 04 18 02 23 04 02 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0C | Packet length = 12 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 02 | Option flag = 0; High Data Rate flag = 1 |
| Read Multiple Blocks | 23 | Read Multiple Blocks cmd |
| Block Number | 04 | First block number = 04 (block #5) |
| Number of Blocks | 02 | Note: Number of read blocks equals number plus one. In this example, reading 3 blocks beginning at block #5. |
| EOF | 00 00 | End of frame |

**Read Multiple Blocks (Tag Response)**

Request Mode

80T60F40E[00110000112200002233000033] Comment: (**80T** end of transmit, **60F** receive data buffer 75% full, **40E** end of receive,
[**00** no tag error, **11 00 00 11** data in block 04,
**22 00 00 22** data in block 05,
**33 00 00 33** data in block 06])

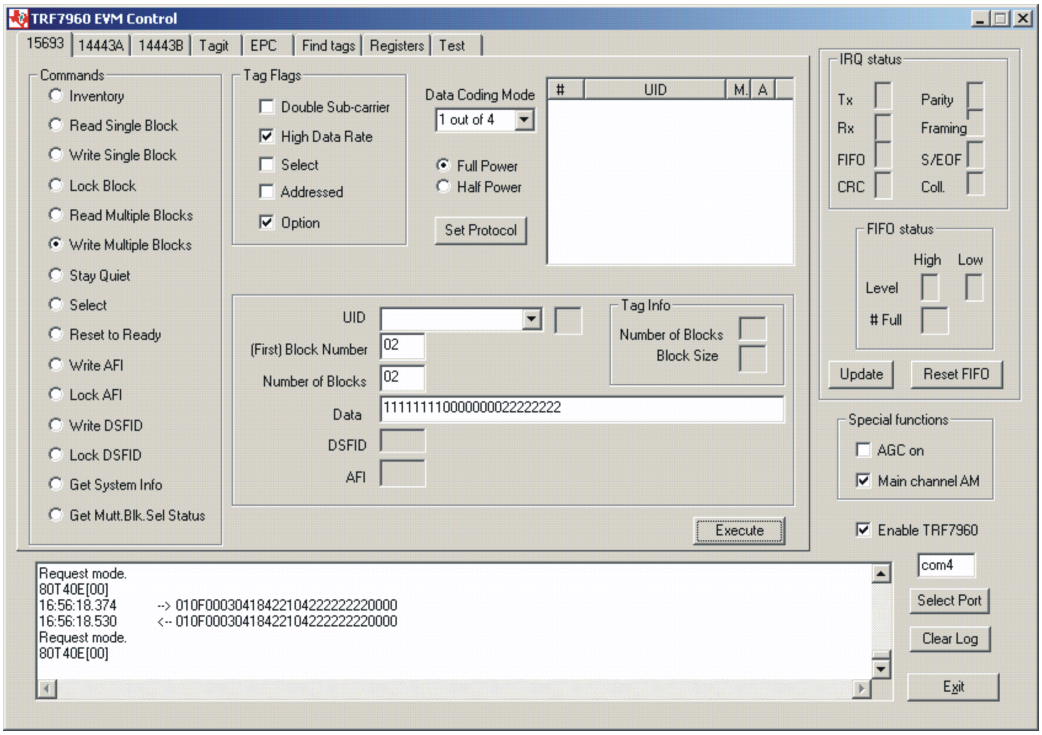
### 3.5.6 Write Multiple Blocks

The *Write Multiple Blocks* command writes data to multiple memory blocks of the addressed tags. In order to successfully write data, the host must know the size of the memory block of the tag. *Write Multiple Blocks* is an optional command, and may not be supported by the tag (see the following screen capture).

To write multiple blocks, the user should:

- Click the button for *Write Multiple Blocks* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- Enter two hex digits corresponding to the starting block number in the *(First) Block Number* field in the *Tag Data* window. The blocks are numbered from *00* to *FF* (0 to 255).
- Enter two hex digits corresponding to the number of blocks to be written in the *Number of Blocks* field in the *Tag Data* window. The number of blocks in the request is one less than the number of blocks that the tag returns in its response.

  E.g., a value of *06* in the *Number of Blocks* field requests to read 7 blocks. A value of *00* requests a read of a single block.
- *Execute* the command.




Note: The *Option* flag (bit 7) of the ISO 15693 defined Request flags must be set to 1 for all Write and Lock commands to respond properly.

Executes **Write Single Block** multiple times.

**01 0F 00 03 04 18 42 21 02** 11 11 11 11 **00 00** **Block 02 write; (block #3)**
**01 0F 00 03 04 18 42 21 03** 00 00 00 00 **00 00** **Block 03 write; (block #4)**
**01 0F 00 03 04 18 42 21 04** 22 22 22 22 **00 00** **Block 04 write; (block #5)**

**(all bytes are continuous; spaces are added for clarity)**

**Example, shown as follows, is last of single multiple write blocks:**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0F | Packet length = 15 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 42 | Option flag = 1; high-data-rate flag = 1 |
| Write Single Block Cmd | 21 | Executes Write Single Block multiple times |
| Block Number | 04 | (First) Block Number = 02 (block #3) as shown in GUI.<br>Note: Number of write blocks equals number of blocks plus one.<br>In this example writing 3 blocks, beginning at block 02. Writing first to block 02, then block 03, and finally to block 04 as shown here. |
| Blocks Data | 22 22 22 22 | 32 bits |
| EOF | 00 00 | End of frame |

**Write Multiple Blocks (Tag Response)**

Request Mode

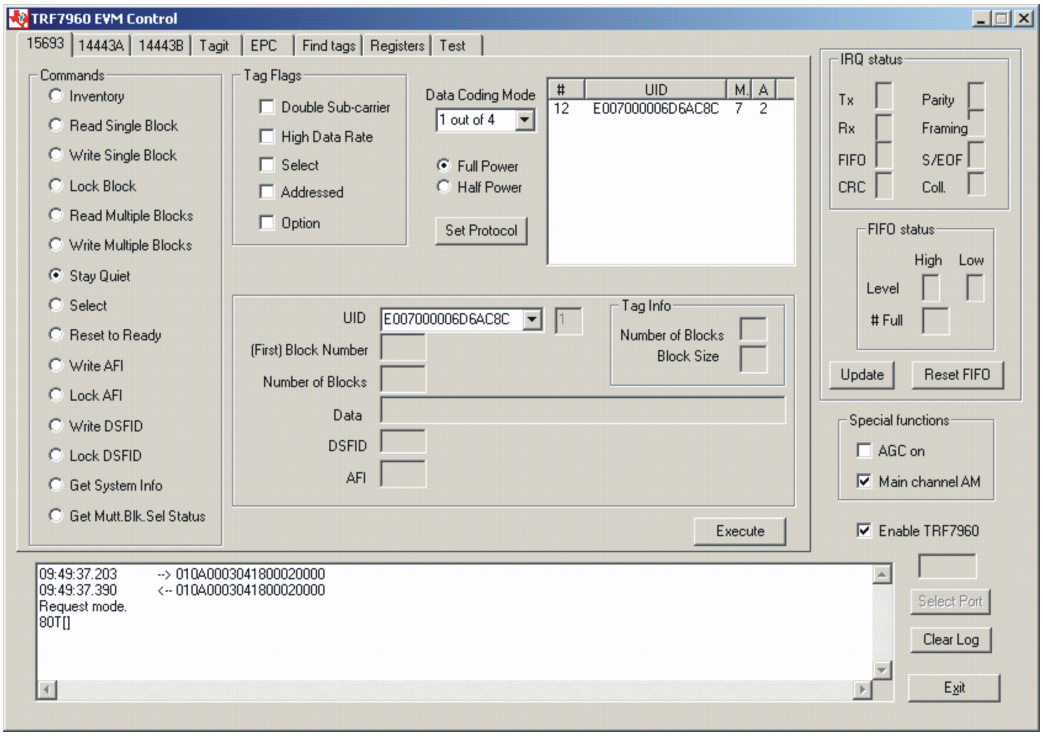
80T40E[00] Comment: (**80T** end of transmit, **40E** end of receive, [**00**] no tag error)

### 3.5.7 Stay Quiet

The *Stay Quiet* command is used to silence a tag, preventing it from responding to any nonaddressed or inventory related commands. The tag does, however, respond to requests with matching UID. As there is no response to this request from the receiving tag, only request status and errors are reported.

To command a tag to stay quiet, the user should:

- Click the button for *Stay Quiet* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window and set the *Addressed* flag (if only one tag is present, only one choice is available).
- *Execute* the command.




**Request Packet:**

**01 0A 00 03 04 18 00 02 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0A | Packet length = 10 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 00 | No flags |
| Stay Quiet Cmd | 02 | |
| EOF | 00 00 | End of frame |

**Stay Quiet (Tag Response)**

Request Mode

80T[] Comment: (**80T** end of transmit, **[]** no tag response)

### 3.5.8 Select

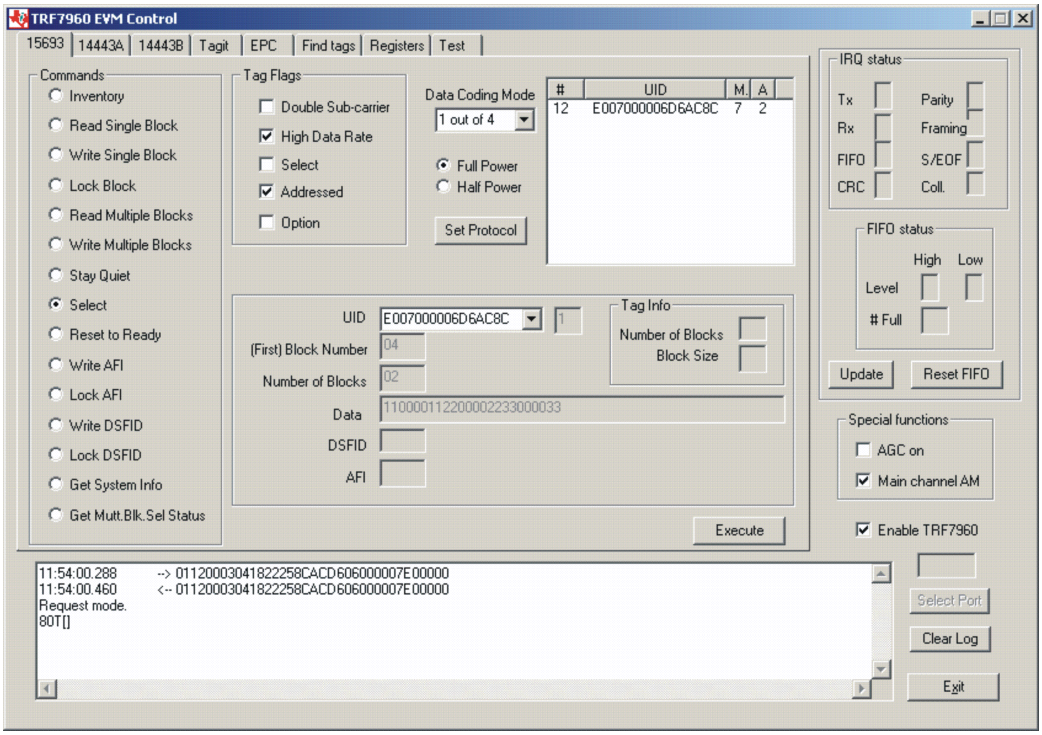
The *Select* command places the addressed tag in the *Select* state. In this state, it responds to requests with the ISO15693 *Select Flag* set. This flag is directly controlled by the <*IsSelectMsg*> field present in many ISO15693 library request messages. Any receiving tag currently in the *Select* state with UID not matching the value sent in the request command, exits that state and enters the *Ready* state but does not send a reply.

To select a tag, the user should:

- Click the button for *Select* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window and set the *Addressed* flag (if only one tag is present, only one choice is available).
- *Execute* the command.

**Information:**

The EVM transmitter remains **ON** in order to preserve the tag states changed by the request.



**Request Packet:**

**01 12 00 03 04 18 22 25 8C AC D6 06 00 00 07 E0 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 12 | Packet length = 18 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 20 | Addressed flag is set. |
| Select Cmd | 25 | |
| UID | 8C AC D6 06 00 00 07 E0 | UID (reverse byte ordered). Normal UID byte order is EO 07 00 00 06 D6 AC 8C. |
| EOF | 00 00 | End of frame |

**Select (Tag Response)**

Request Mode

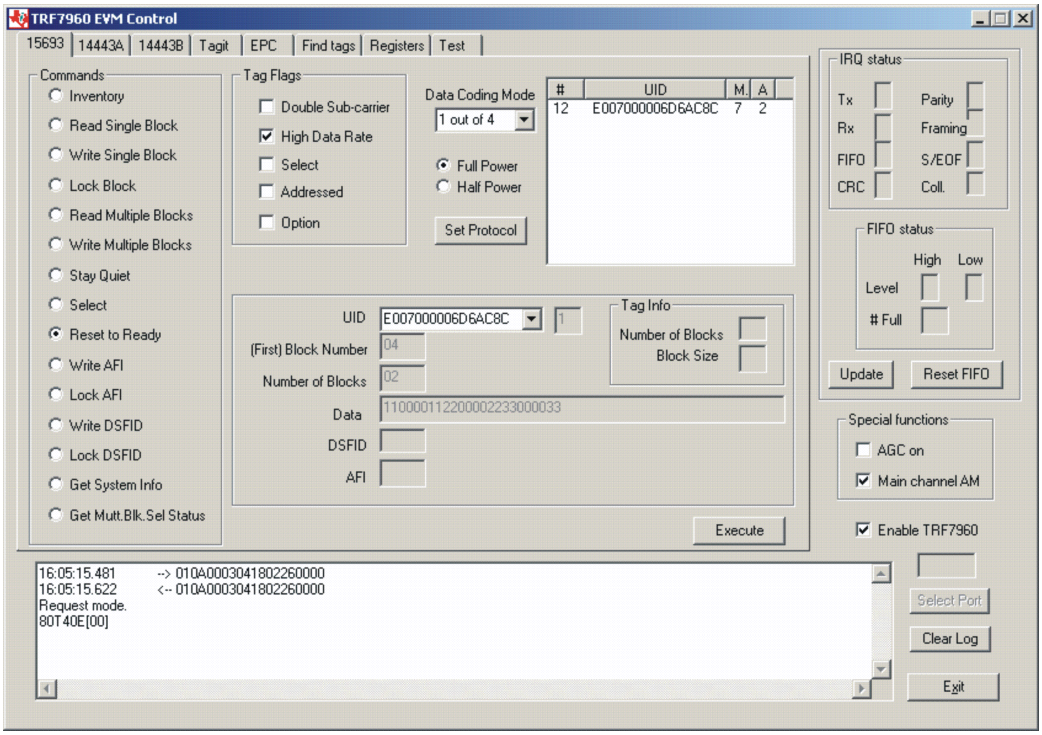
80T[] Comment: (**80T** end of transmit, **[]** no tag response)

### 3.5.9 Reset to Ready

The *Reset To Ready* command places the addressed tag in the *Ready* state. In this state, it does not respond to requests with the ISO15693 *Select Tag Flags* set, but to any nonaddressed request or request matching its UID. This command is, in effect, the complement of the *Select* command, and undoes it.

To reset a tag, the user should:

- Click the button for *Reset to Ready* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- *Execute* the command.




**Request Packet:**

**01 0A 00 03 04 18 02 26 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0A | Packet length = 10 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 00 | No flags |
| Reset to Ready Cmd | 26 | |
| EOF | 00 00 | End of frame |

**Reset to Ready (Tag Response)**

Request Mode

80T40E[00] Comment: (**80T** end of transmit, **40E** end of receive, [**00**] no tag error)

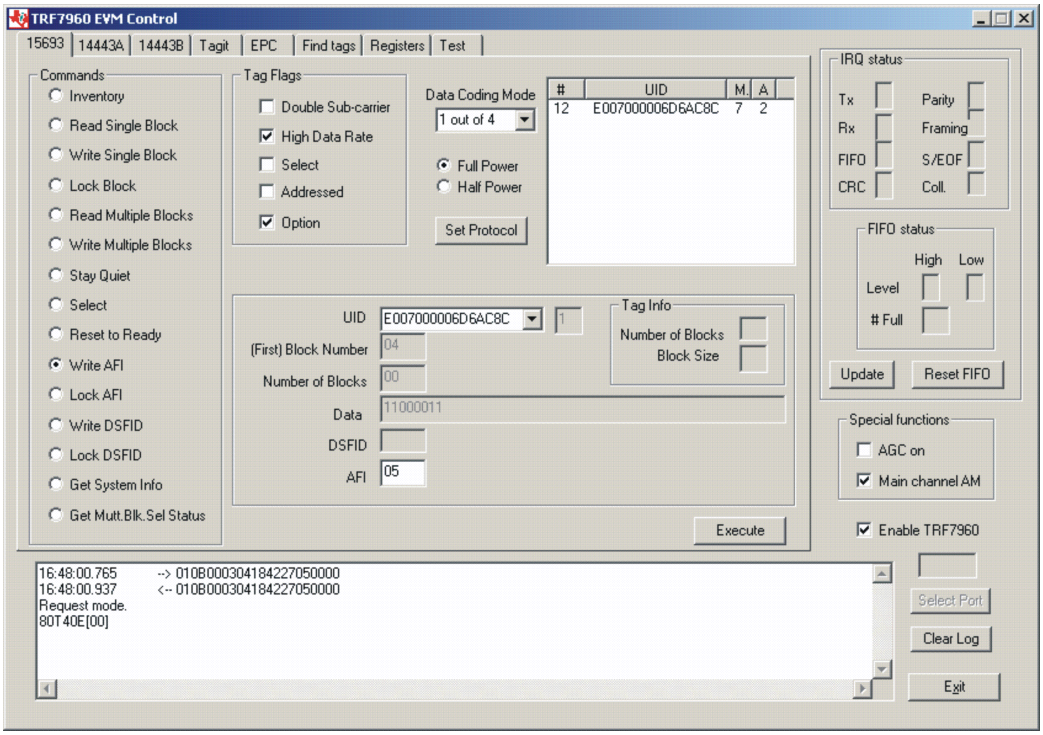
### 3.5.10 Write AFI (Application Family Identifier)

The *Write AFI* command records a new value to the AFI register (see Section A.6 for AFI codes) of the addressed tag(s). A corrupted response or lack of response from TRF7960 does not necessarily indicate a failure to perform the write operation. Additionally, multiple transponders may process a non-addressed request.

AFI represents the tag application, and is used to extract information from tags meeting the application criteria.

To write a tag's AFI, the user should:

- Click the button for *Write AFI* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- Enter the desired AFI code in the *AFI* field in the *Tag Data* window (in hexadecimal).
- *Execute* the command.




Note: The *Option* flag (bit 7) of the ISO 15693 defined Request flags must be set to 1 for all Write and Lock commands to respond properly.

**Request Packet:**

**01 0B 00 03 04 18 42 27 05 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0B | Packet length = 11 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 42 | Option flag = 1; High Data Rate flag = 1 |
| Write AFI Cmd | 27 | |
| AFI | 05 | Application family identifier, 05 = medical application |
| EOF | 00 00 | End of frame |

**Write AFI (Tag Response)**

Request Mode

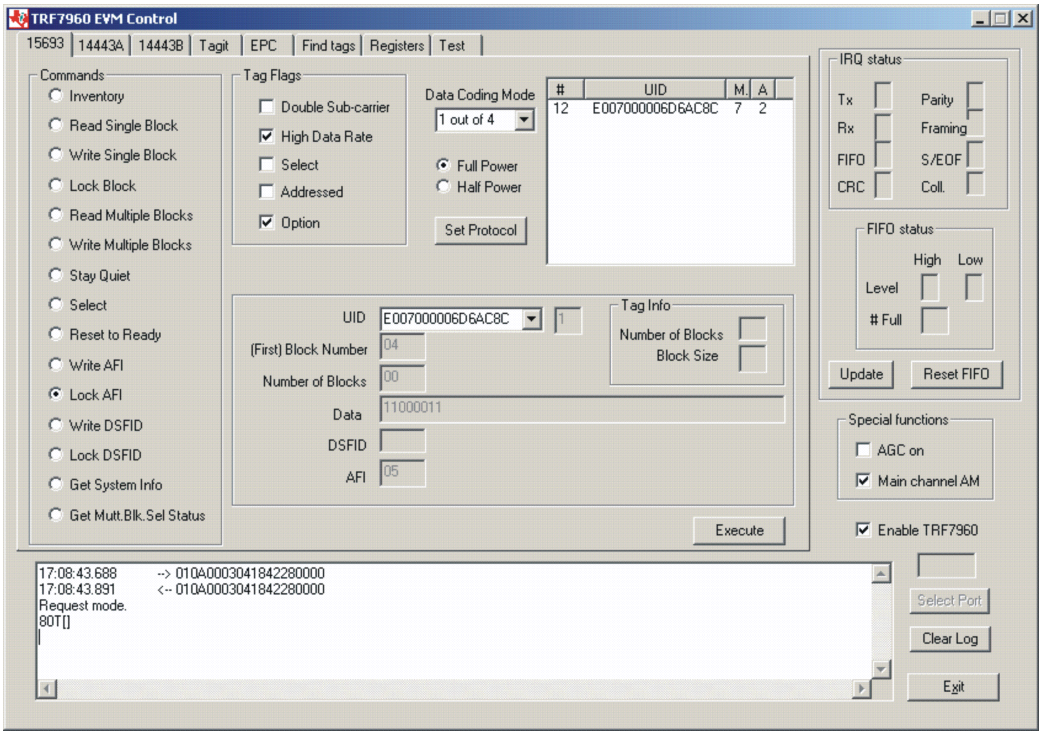
80T40E[00] Comment: (**80T** end of transmit, **40E** end of receive, [**00**] no tag error)

### 3.5.11 Lock AFI (Application Family Identifier)

The *Lock AFI* command write-protects the AFI register of the addressed tag(s). A corrupted response or lack of response does not necessarily indicate a failure to perform the lock operation. Additionally, multiple transponders may process a nonaddressed request.

To a lock tag's AFI, the user should:

- Click the button for *Lock AFI* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- *Execute* the command.




Note: The *Option* flag (bit 7) of the ISO 15693 defined Request flags must be set to 1 for all Write and Lock commands to respond properly.

**Request Packet:**

**01 0A 00 03 04 18 42 28 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0A | Packet length = 10 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 42 | Option flag = 1; High Data Rate flag = 1 |
| Lock AFI Cmd | 28 | |
| EOF | 00 00 | End of frame |

**Lock AFI (Tag Response)**

Request Mode

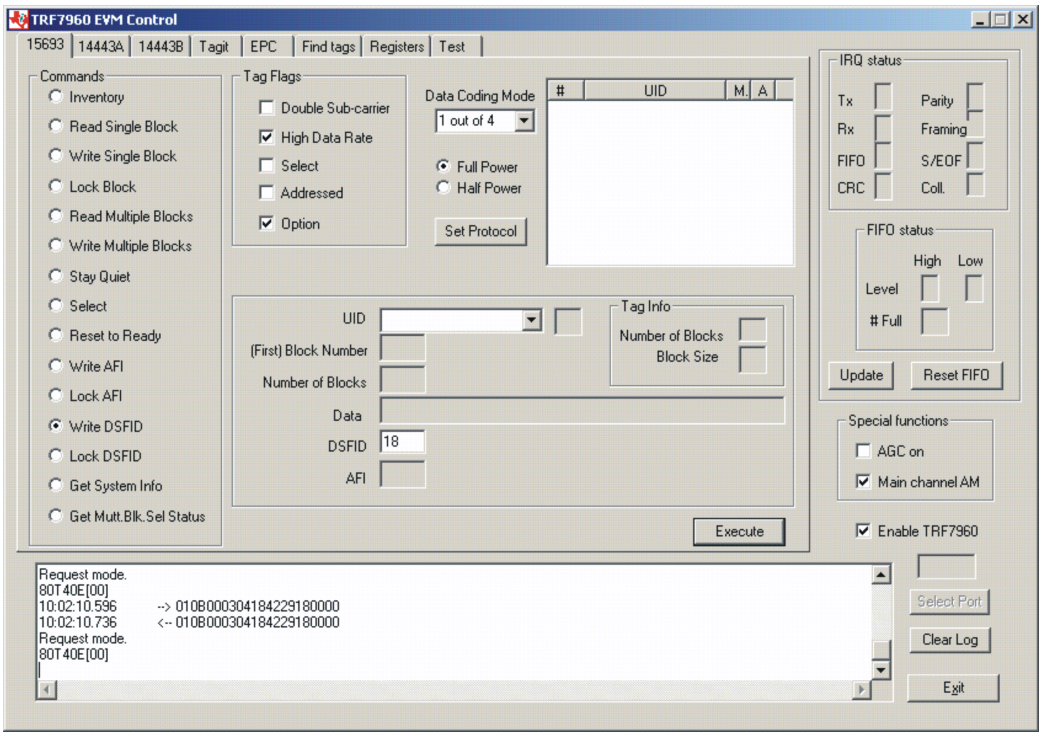
80T Comment: (**80T** end of transmit, **[]** no tag error)

### 3.5.12 Write DSFID (Data Storage Format ID)

The *Write DSFID* (data storage format ID) command writes a new value in the DSFID register of the addressed tag(s). A corrupted response or lack of response from the TRF7960 does not necessarily indicate a failure to perform the write operation. Additionally, multiple transponders may process a nonaddressed request.

To write a tag's DSFID, the user should:

- Click the button for *Write DSFID* in the *Commands* Window.
- Click on any flags that must be set in the *Tag Flags* window.
- Select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- Enter the desired DSFID code in the *DSFID* field in the *Tag Data* window (in hexadecimal).
- *Execute* the command.




Note: The *Option* flag (bit 7) of the ISO 15693 defined Request flags must be set to 1 for all Write and Lock commands to respond properly.

**Request Packet:**

**01 0B 00 03 04 18 42 29 18 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0B | Packet length = 11 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 42 | Option flag = 1; High Data Rate flag = 1 |
| Write DSFID Cmd | 29 | |
| DSFID value | 18 | Data Storage Format ID |
| EOF | 00 00 | End of frame |

**Write DSFID (Tag Response)**

Request Mode

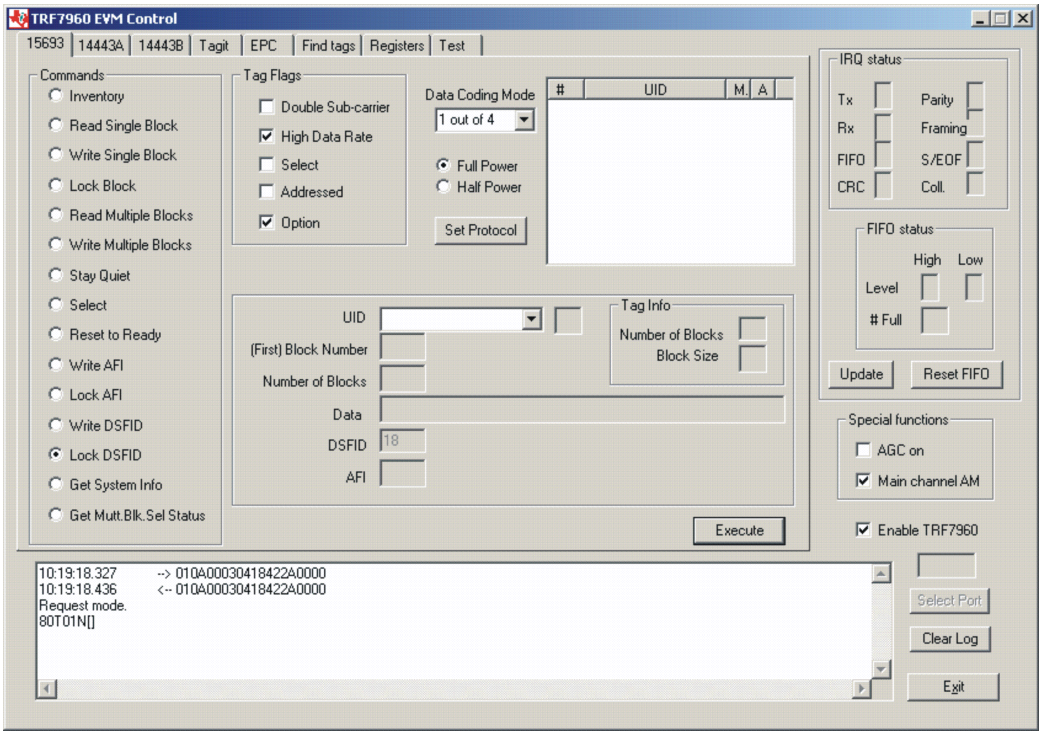
80T40E[00] Comment: (**80T** end of transmit, **40E** end of receive, [**00**] no tag error)

### 3.5.13 Lock DSFID (Data Storage Format ID)

The *Lock DSFID* command write-protects the DSFID register of the addressed tag(s). A corrupted response or lack of response from TRF7960 does not necessarily indicate a failure to perform the lock operation. Additionally, multiple transponders may process a nonaddressed request.

To a lock tag's DSFID, the user should:

- Click the button for *Lock DSFID* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- *Execute* the command.




Note: The *Option* flag (bit 7) of the ISO 15693 defined Request flags must be set to 1 for all Write and Lock commands to respond properly.

**Request Packet:**

**01 0A 00 03 04 18 42 2A 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0A | Packet length = 10 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 42 | Option flag = 1; High Data Rate flag = 1 |
| Lock DSFID Cmd | 2A | |
| EOF | 00 00 | End of frame |

**Lock DSFID (Tag Response)**

Request Mode

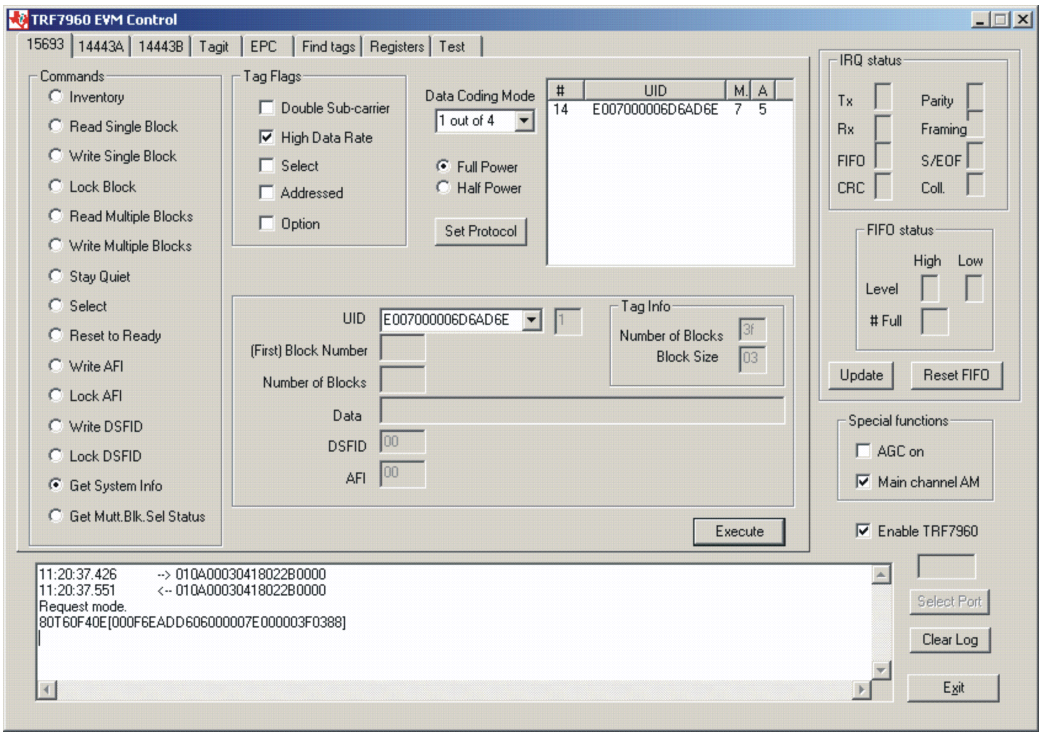
80T01N[] Comment: (**80T** end of transmit, **01N** no response interrupt, **[]** no tag response)

### 3.5.14 Get System Info

The *Get System Info* command retrieves identification, application family, data formatting, and memory block sizes as specified in the ISO15693 standard (if tag supports this command).

To get system information, the user should:

- Click the button for *Get System Info* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- *Execute* the command.



**Request Packet:**

**01 0A 00 03 04 18 02 2B 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0A | Packet length = 10 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 02 | Option flag = 0; high-data-rate flag = 1 |
| Get System Info Cmd | 2B | |
| EOF | 00 00 | End of frame |

### Get System Info (Tag Response)

Reader / Tag response (0 thru 15 slots) shown as follows:
IRQ Status Register; [ < Tag response if any >, RSSI Register value]

Example:

Request Mode

80T60F40E [000F6EADD606000007E000003F0388] Comment: **80T** end of transmit, **60F** receive data buffer 75% full, **40E** end of receive,

[**00 0F 6EADD606000007E0 00 00 3F 03 88**] tag response shown as follows:

| Field | Contents | Comment |
|---|---|---|
| Tag Error Flag | 00 | 00 = no error |
| Tag Information Flag | 0F | Tag reference field present<br>Tag memory field present<br>Tag AFI field present<br>Tag DSFID field present |
| Tag UID | 6EADD606000007E0 | Reverse byte ordered.<br>Normal UID byte order is EO 07 00 00 06 D6 AD 6E. |
| Tag DSFID Value | 00 | Data Storage Format ID |
| Tag AFI Value | 00 | |
| Tag Other Fields | 3F 03 88 | 3F meaning number of blocks = 64<br>03 meaning block size = 32 bits<br>88 defined by tag manufacturer |

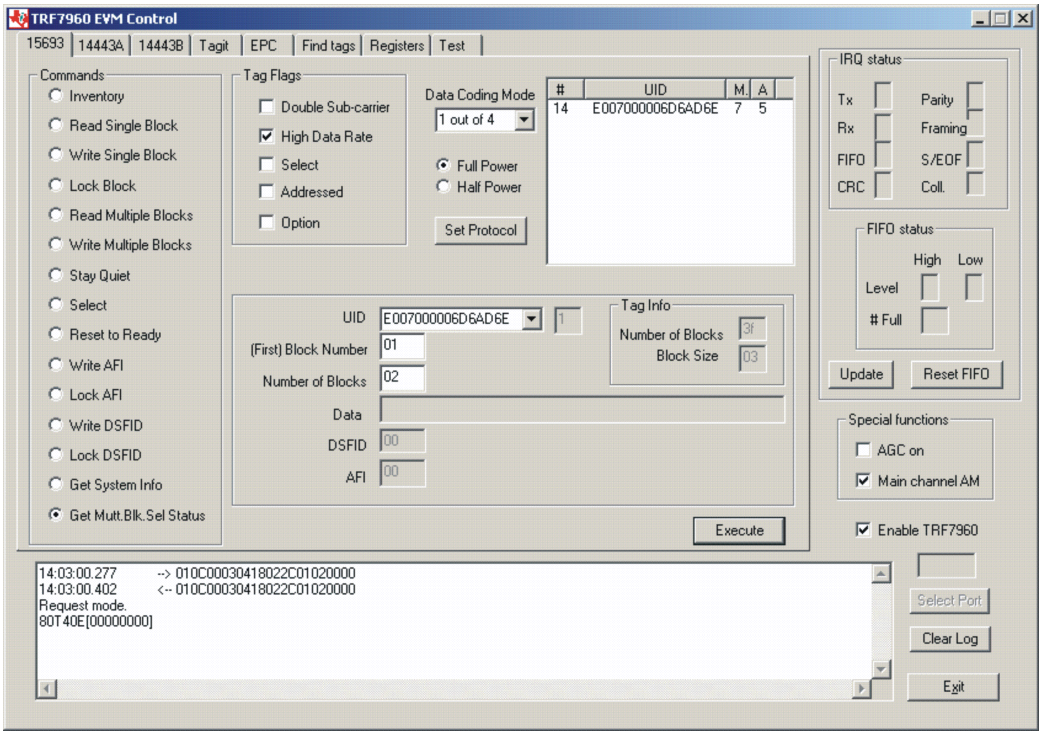
### 3.5.15 Get Multiple-Block Security Status (Get Mult_Blk Sel Status)

The *Get Multiple-Block Security Status (Get Mutt. Blk. Sel Status)* command gets a block security status byte for each block requested. This byte encodes the write protection of the block specified (e.g., unlocked, (user/factory) locked, etc.).

To get multiple block security status, the user should:

- Click the button for *Get Mult.Blk.Sel Status* in the *Commands* window.
- Click on any flags that must be set in the *Tag Flags* window.
- Optionally select a tag from the *UID* pulldown list in the *Tag Data* window (if only one tag is present, only one choice is available).
- Enter two hex digits corresponding to the starting block number in the *(First) Block Number* field in the *Tag Data* window. The blocks are numbered from *00* to *FF* (0 to 255).
- Enter two hex digits corresponding to the number of blocks to be written in the *Number of Blocks* field in the *Tag Data* window. The number of blocks in the request is one less than the number of blocks that the tag returns in its response.

  E.g., a value of *06* in the *Number of Blocks* field requests to read 7 blocks. A value of *00* requests to read a single block.
- *Execute* the command.



**Request Packet:**

**01 0C 00 03 04 18 02 2C 01 02 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0C | Packet length = 12 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Flags | 02 | Option flag = 0; High Data Rate flag = 1 |
| Get Multiple Block Security Status Cmd | 2C | |
| Block Number | 01 | (First) block number = 01 (block #2) |
| Number of Blocks | 02 | Number of blocks = 3. Note: Number of read blocks equals number plus one. In this example; reading 3 blocks beginning at block #2. |
| EOF | 00 00 | End of frame |

**Get Multiple Block Security Status (Tag Response)**

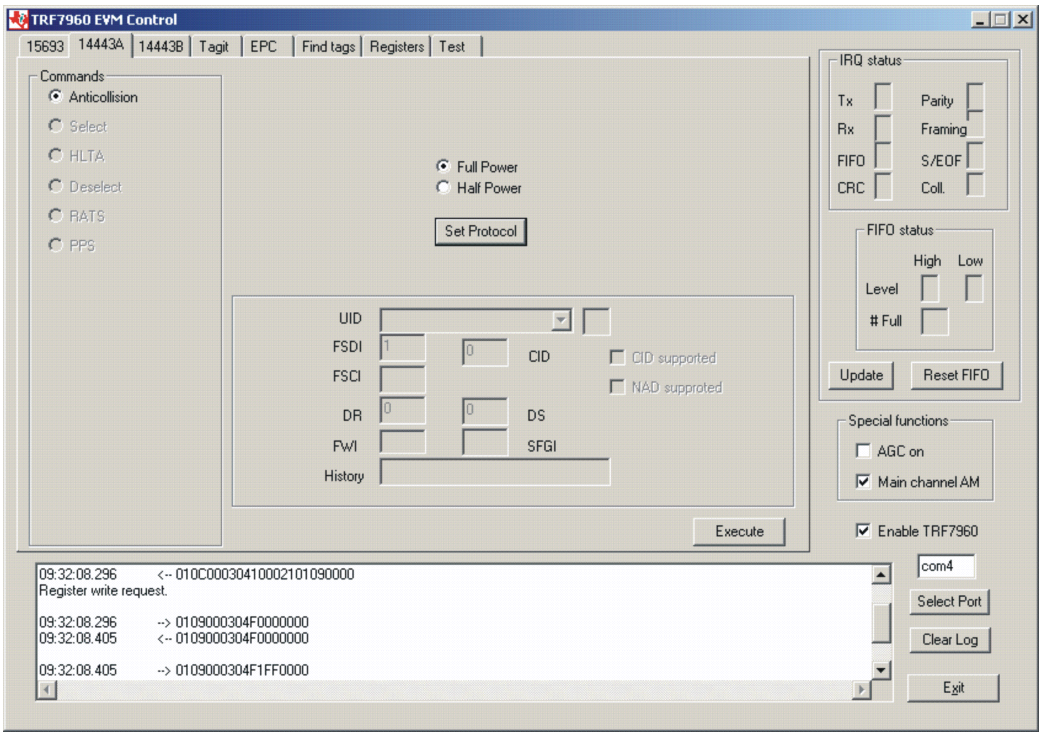
Request Mode

80T40E[00000000] Comment: (**80T** end of transmit , **40E** end of receive, [**00** no tag error,
**00** security status of block number 01 (block #2),
**00** security status of block number 02 (block #3),
**00** security status of block number 03 (block #4])

## 3.6 ISO/IEC 14443A Protocol

This section describes the ISO/IEC 14443A protocol. Program operation is a little different in this protocol compared to ISO 15593. Some commands must be run in sequence: e.g., an *Anticollision* command, when executed, activates a radio button for the *Select* command, etc.

An ISO14443A set protocol command sends three commands (register write, set AGC, and set receiver mode (AM / PM).




### First Command: Register Write

**01 0C 00 03 04 10 00 21 01 09 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0C | Packet length = 12 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 10 | Register write |
| Register 00 | 00 21 | In Register 00 (Chip Status Control register), Write 21 (RF output active, 5-V operation). |
| Register 01 | 01 09 | In Register 01 (ISO Control register), Write 09 (set protocol to ISO1443A, high bit rate, 212 kbps). |
| EOF | 00 00 | End of frame |

### Second Command: Set AGC

**01 09 00 03 04 F0 00 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | F0 | AGC toggle |
| AGC Off | 00 | AGC on = FF |
| EOF | 00 00 | End of frame |

### Third Command: Set Receiver Mode

**01 09 00 03 04 F1 FF 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | F1 | AM / PM toggle |
| AM | FF | FF = AM, while a 00 = PM |
| EOF | 00 00 | End of frame |

### 3.6.1 Anticollision (Execute Button)

The *Anticollision* command is linked with the *Select* command, in that it must be run first.

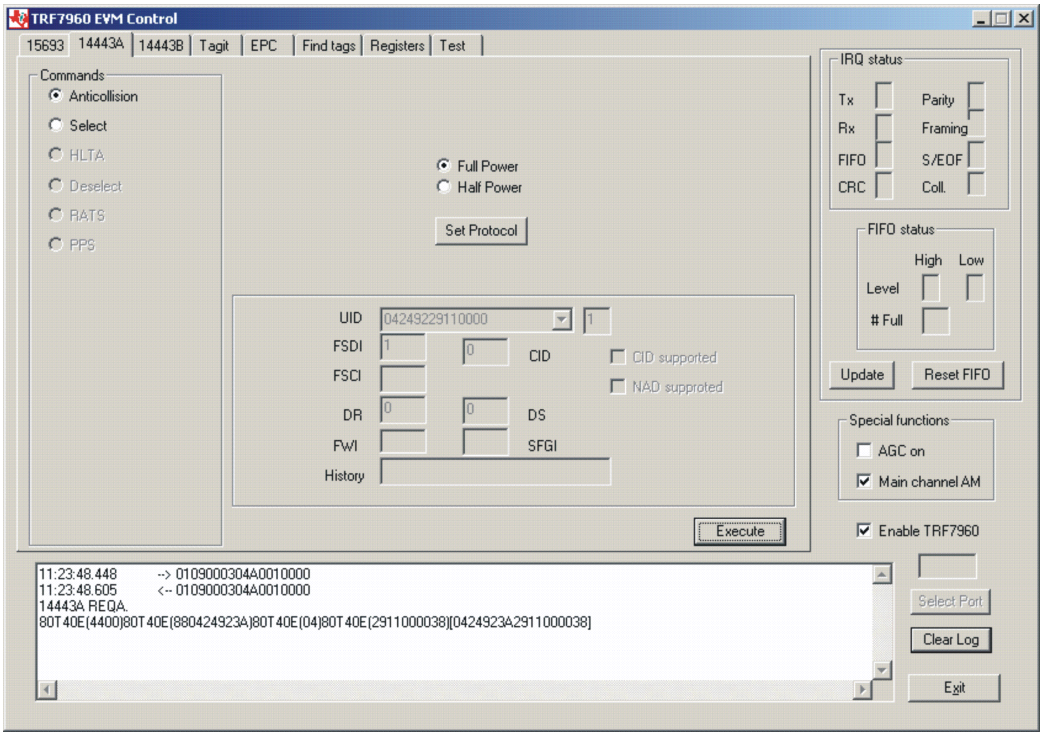
The request packet specifies the cascade level of the UID, the number of bits send to the tag(s) using *Anticollision/Select* frame and the actual data bits/bytes. The *Anticollision* request is transmitted in a bit-oriented anticollision frame.

The *Select* request is sent over the RF interface in a *Standard* frame. The *Anticollision* request may specify the number of bits in the range 0 through 39, i.e., [0, 39]. The *Select* request must always specify 40 bits to send. Even if the number of bits specified is less than 40, five bytes of data must follow. The complete UID must be collected from the tag before *Select* with 40 bits of UID can be attempted.

After a successful Anticollision/Select execution, the tag responds with ERROR_NONE in the Response Status byte field. The data field contains the sent data bits and the data bits of the UID that could be resolved up to any collision or up to the full UID.

To do an Anticollision/Select, the user should:

- Click the button for *Anticollision* in the *Commands* window.
- *Execute* the command.
- Click the button for *Select* in the *Commands* window.
- *Execute* the command.



**Request Packet:**

**01 09 00 03 04 A0 01 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | A0 | Tag type A, anticollision, REQA |
| REQA | 01 | 01 = REQA (REQuest type A)<br>00 = WUPA (Wake UP type A) |
| EOF | 00 00 | End of frame |

### Anticollision (Tag Response)

14443A REQA.

80T40E(4400)80T40E(880424923A)80T40E(04)80T40E(2911000038)[0424923A2911000038]

Shown are several tag responses with the following format:
IRQ Status Register **(**< Tag response with no CRC >**)**
IRQ Status Register **[**< Tag response with CRC >**]**
A tag response with "**( )**" indicates a response with no CRC, while
A tag response with "**[ ]**" indicates a response with CRC.

| | | |
|---|---|---|
| 80T40E(4400) | | ATQA (answer to request, type A), UID size single, no bit-frame anticollision |
| 80T40E(880424923A) | 88 | Cascade tag 88 |
| | 042492 | 3 bytes of UID, bytes UID0, UID1, UID2 |
| | 3A | BCC (block character check) |
| 80T40E(04) | | SAK (select acknowledge, type A), cascade bit set |
| 80T40E(2911000038) | 29110000 | 4 bytes of UID, bytes UID3, UID4, UID5, UID6 |
| | 38 | BCC (block character check) |
| [0424923A2911000038] | | Complete UID response + 4 BCC bytes |
| | | UID = 04249229110000—7 bytes (or 56 bits) |

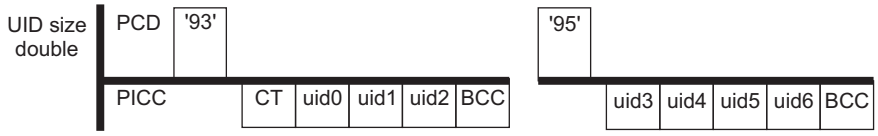
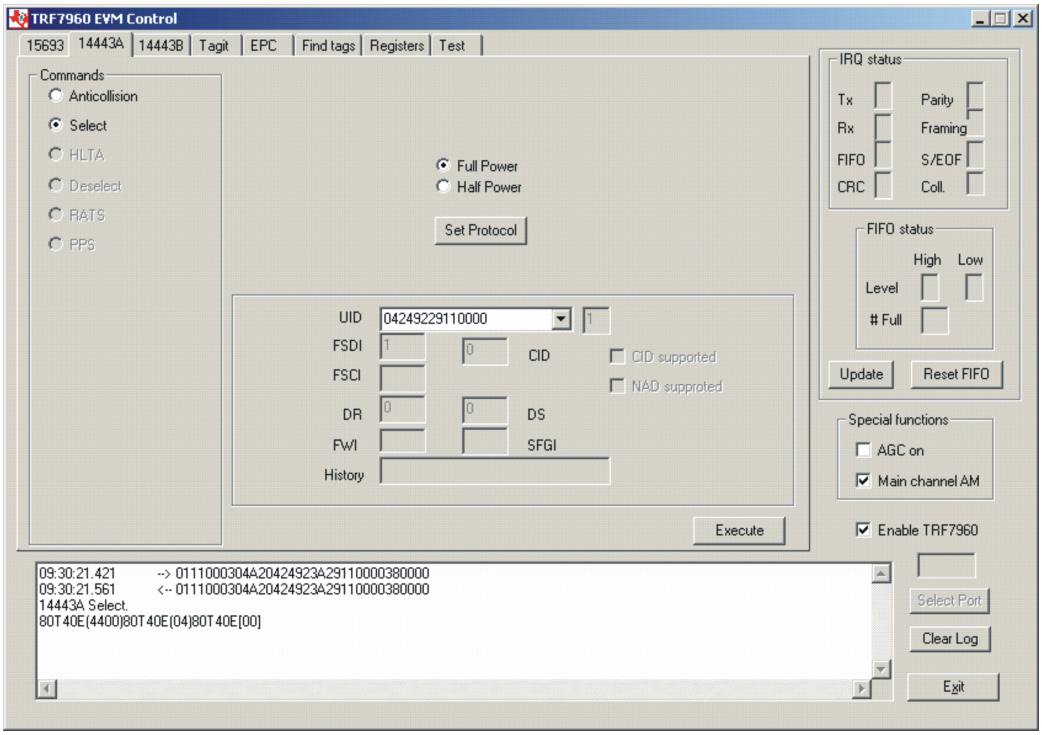



**Figure 3-1. Example Cascaded Byte**

### 3.6.2 Select




**Request Packet:**

**01 11 00 03 04 A2 04 24 92 3A 29 11 00 00 38 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 11 | Packet length = 17 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | A2 | Select |
| UID | 0424923A2911000038 | Complete Tag UID (04249229110000)<br>[3A and 38 are block character checks (BCC)] |
| EOF | 00 00 | End of frame |

**Select (Tag Response)**

14443A Select.

80T40E(4400)80T40E(04)80T40E[00]

Shown are several tag responses with the following format:

IRQ Status Register **(**< Tag response with no CRC >**)**

IRQ Status Register **[**< Tag response with CRC >**]**

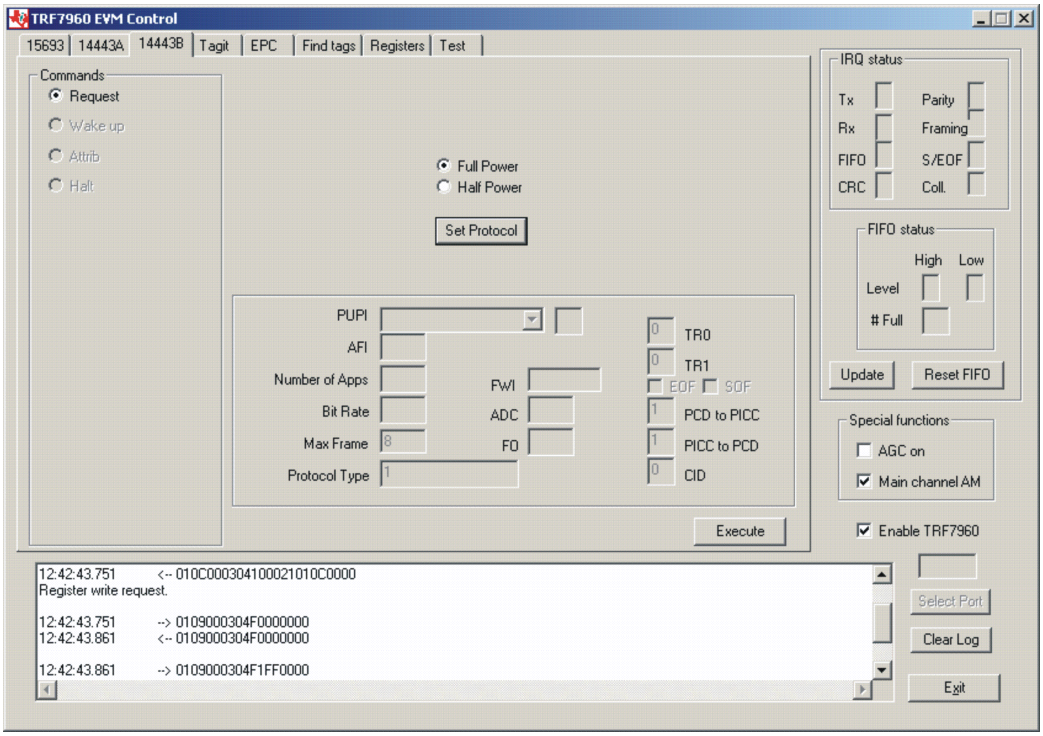
A tag response with "**( )**" indicates a response with no CRC, while
A tag response with "**[ ]**" indicates a response with CRC.

| | |
|---|---|
| 80T40E(4400) | ATQA (answer to request, type A), UID size single, no bit-frame anticollision |
| 80T40E(04) | SAK (select acknowledge, type A), cascade bit set |
| 80T40E[00] | UID received with no error (tag enters ACTIVE state) |

## 3.7 ISO/IEC 14443B Protocol

This section describes the ISO 14443B protocol. Similar to the ISO 14443A protocol, program operation is a little different compared to ISO 15693; some commands must be run in sequence.

An ISO 14443B set protocol command sends three commands (register write, set AGC, and set receiver mode (AM / PM).




**First Command: Register Write**
**01 0C 00 03 04 10 00 21 01 0C 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0C | Packet length = 12 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 10 | Register write |
| Register 00 | 00 21 | In Register 00 (Chip Status Control register), Write 21 (RF output active, 5-V operation) |
| Register 01 | 01 0C | In Register 01 (ISO Control register), Write 12 (set ISO14443B protocol, 106 kbps) |
| EOF | 00 00 | End of frame |

### Second Command: Set AGC
### 01 09 00 03 04 F0 00 00 00 (all bytes are continuous; spaces are added for clarity)

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | F0 | AGC toggle |
| AGC Off | 00 | AGC on = FF |
| EOF | 00 00 | End of frame |

### Third Command: Set Receiver Mode
### 01 09 00 03 04 F1 FF 00 00 (all bytes are continuous; spaces are added for clarity)

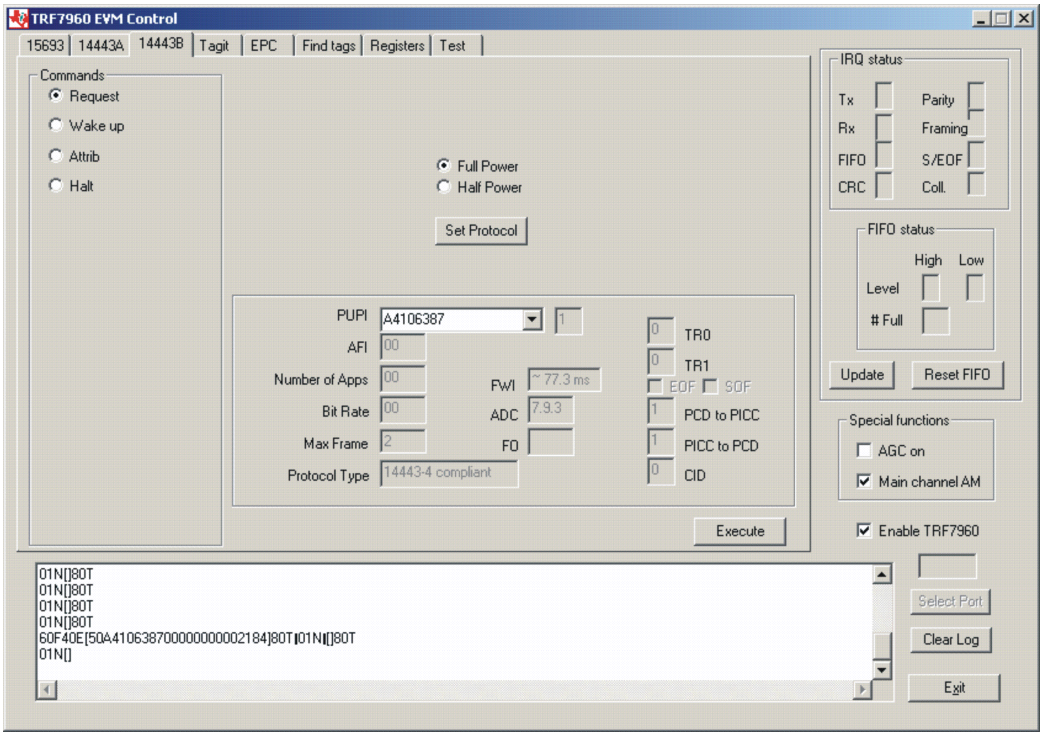
| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | F1 | AM / PM toggle |
| AM | FF | FF = AM, while a 00 = PM |
| EOF | 00 00 | End of frame |

### 3.7.1 Request Command (REQB Cmd Format)

The *Request* command determines if a tag is present.

To do a *Request* command, the user should:

- Click the button for *Request* in the *Commands* window.
- *Execute* the command.




**Request Packet:**

**01 09 00 03 04 B0 04 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | B0 | Tag type B, anticollision – REQB |
| Enable 16 Slots | 04 | |
| EOF | 00 00 | End of frame |

**Request (Tag Response)**

80T01N[]80T Comment: (slot # 0, **80T** end of transmit, **01N** no response interrupt, **[]** < no tag response >)

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

60F40E[50A41063870000000002184]80T

01N[]80T

01N[]

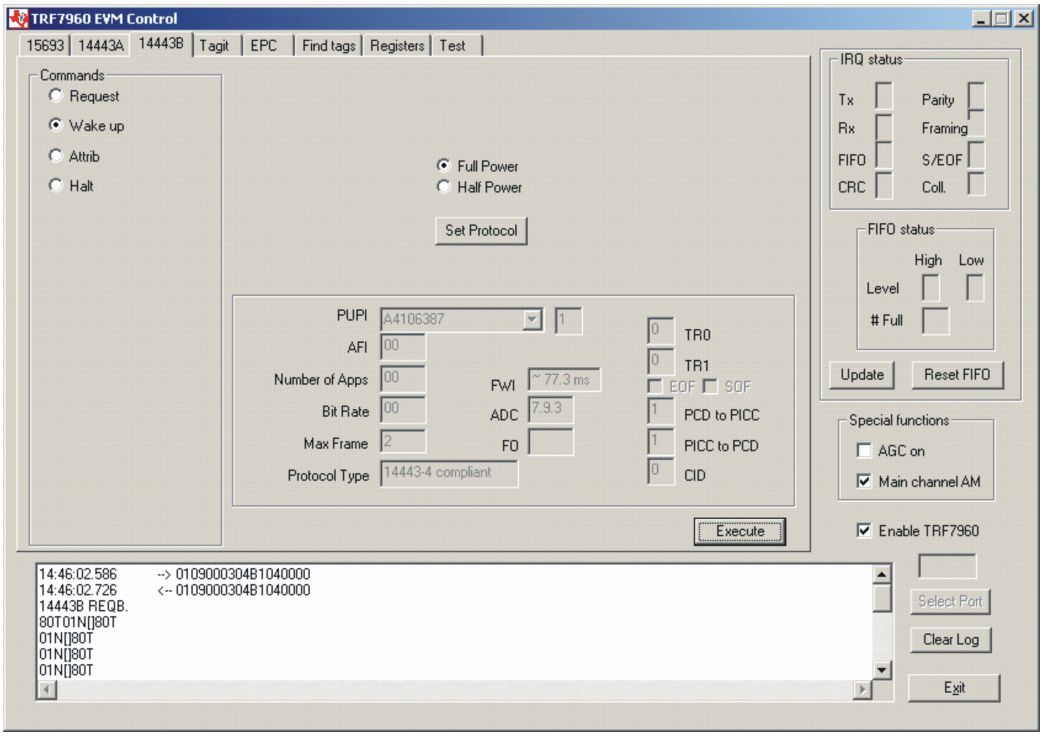
Examination of slot #13 is as follows:

| | |
|---|---|
| 50 | ATQB response header |
| A4106387 | PUPI (Pseudo-unique PICC identifier) |
| 00 00 00 00 | Application Data |
| 00 21 84 | Protocol information as follows: |

| | |
|---|---|
| 00 | Bit rate capability (PICC supports only 106 kbps in both directions) |
| 2 | 32 bytes (maximum frame size) |
| 1 | Protocol type (compliant with 14443-4) |
| 8 | FWI (frame waiting time integer) |
| 4 | ADC + FO (data coding options) |

### 3.7.2 Wake Up B

A Wake Up command is used to take a tag from the HALT state to its idle state.




**Request Packet:**

**01 09 00 03 04 B1 04 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | B1 | WUPB (wake up B) |
| Enable 16 slots | 04 | |
| EOF | 00 00 | End of frame |

**Wake Up (Tag Response)**

Response to Wake Up is as follows:

14443B REQB.

80T01N[]80T  Comment: (Slot # 0, **80T** end of transmit, **01N** no response interrupt, **[]** no tag response)

01N[]80T

01N[]80T

01N[]80T

01N[]80T

60F40E[50A4106387000000000002184]80T01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

01N[]80T

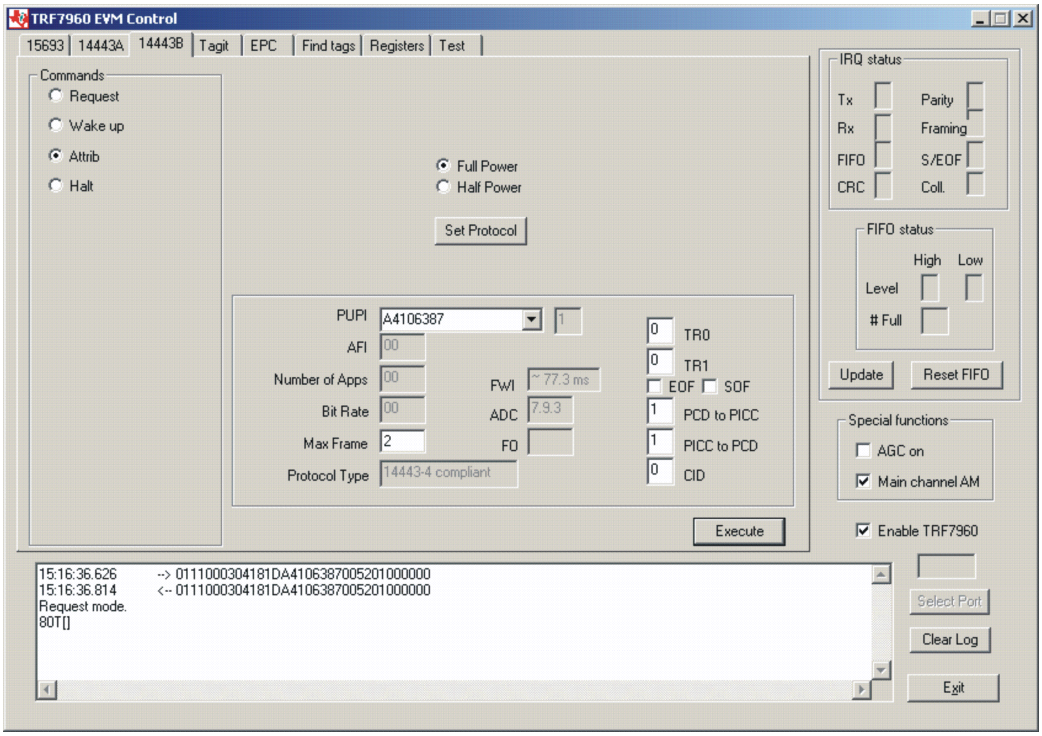
01N[]

Examination of slot #13 is as follows:

| | |
|---|---|
| 50 | ATQB response header |
| A4106387 | PUPI (Pseudo-unique PICC identifier) |
| 00 00 00 00 | Application Data |
| 00 21 84 | Protocol information as follows: |

- 00 Bit rate capability (PICC supports only 106 kbps in both directions)
- 2 32 bytes (maximum frame size)
- 1 Protocol type (compliant with 14443-4)
- 8 FWI (frame waiting time integer)
- 4 ADC + FO (data coding options)

### 3.7.3 ATTRIB (PICC or Tag Selection Cmd, Type B)




**Request Packet:**

**01 11 00 03 04 18 1D A4 10 63 87 00 52 01 00 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 11 | Packet length = 17 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request Mode |
| Constant Header | 1D | Always 1D |
| PUPI | A4106387 | Pseudo-unique PICC identifier |
| Param 1 | 00 | TR0 and TR1 (guard time) are defaults; SOF and EOF are required. |
| Param 2 | 52 | Data rate is 212 kbps; maximum frame size is 32 bytes. |
| Param 3 | 01 | PICC (or tag) compliant with 14443-4 |
| Param 4 | 00 | CID (card identifier) not supported |
| EOF | 00 00 | End of frame |

**ATTRIB (Tag Response)**

Request mode.

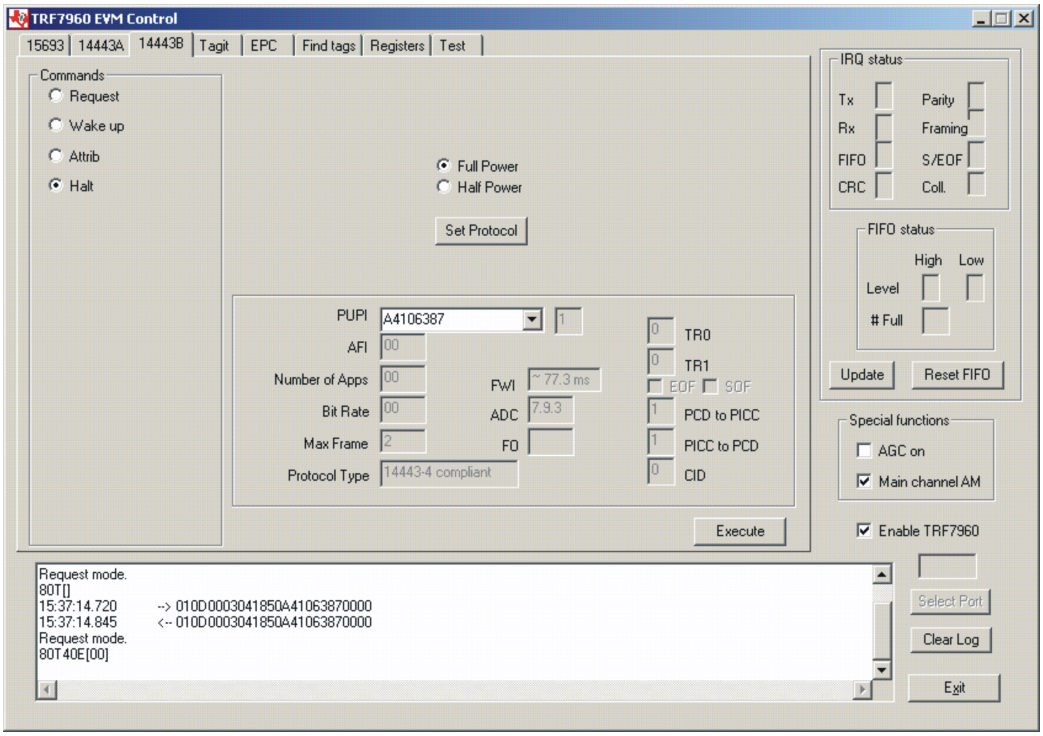
80T[] Comment: (**80T** end of transmit, **[]** no tag response)

### 3.7.4 HALTB Command

The HALTB cmd is used to set a PICC in a HALT state, which in turn stops the PICC from responding to a REQB command. After entering this state, the PICC ignores all commands except the WUPB (wake up B) command.

To do an *HALTB* command, the user should:

- Click the button for *HALTA* in the *Commands* window.
- *Execute* the command.




**Request Packet:**

**01 0D 00 03 04 18 50 A4 10 63 87 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0D | Packet length = 13 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| Response Header | 50 | Always 50 |
| PUPI | A4106387 | Pseudo-unique PICC identifier |
| EOF | 00 00 | End of frame |

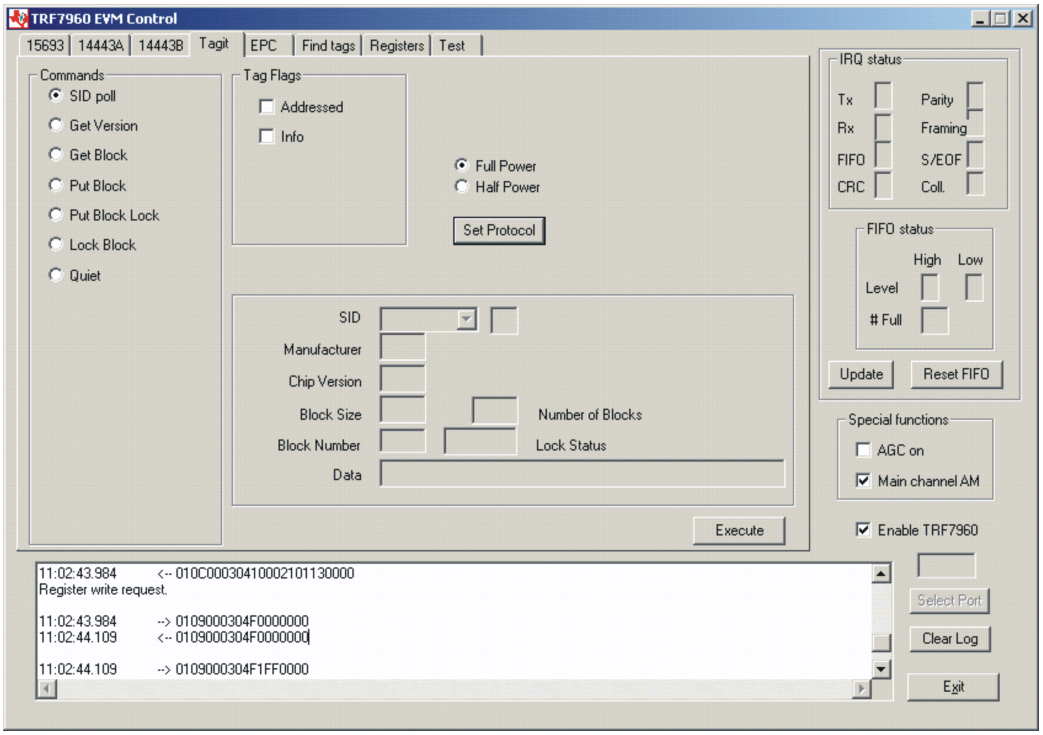
**HALTB (Tag Response)**

Request mode.

80T40E[00] Comment: (**80T** end of transmit, **40E** end of receive, **[00]** no tag error)

## 3.8 Tag-it Protocol

A Tag-It set protocol command sends three commands (register write, set AGC, and set receiver mode (AM/PM).




**First Command: Register Write**
**01 0C 00 03 04 10 00 21 01 13 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0C | Packet length = 12 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 10 | Register write |
| Register 00 | 00 21 | In Register 00 (Chip Status Control register), Write 21 (RF output active, 5-V operation) |
| Register 01 | 01 13 | In Register 01 (ISO Control register), Write 13 (set Tag-It protocol) |
| EOF | 00 00 | End of frame |

### Second Command: Set AGC
### 01 09 00 03 04 F0 00 00 00 (all bytes are continuous; spaces are added for clarity)

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | F0 | AGC toggle |
| AGC Off | 00 | AGC on = FF |
| EOF | 00 00 | End of frame |

### Third Command: Set Receiver Mode
### 01 09 00 03 04 F1 FF 00 00 (all bytes are continuous; spaces are added for clarity)

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 09 | Packet length = 9 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | F1 | AM / PM toggle |
| AM | FF | FF = AM, while a 00 = PM |
| EOF | 00 00 | End of frame |

### 3.8.1 Simultaneous ID (SID) Poll

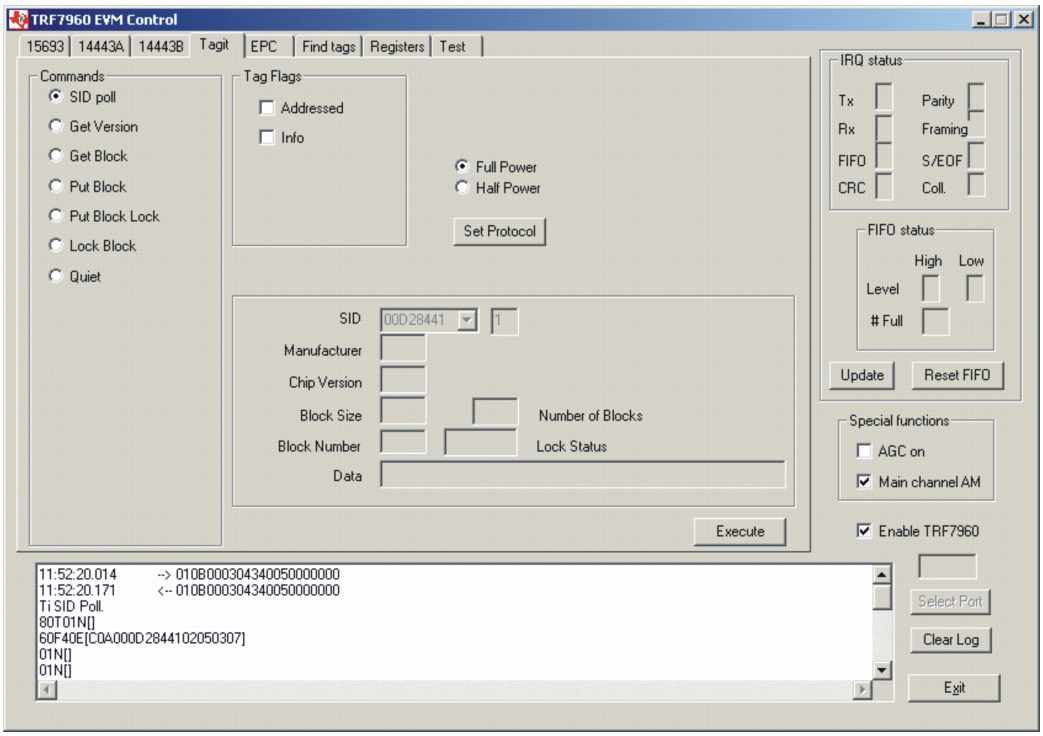
The *SID poll* request is used to acquire the simultaneous IDs of Tag-it transponders. This request decreases the likelihood of a data collision by forcing transponders to respond in 1 of 16 slots based on a portion of their SIDs. To perform a slotted sequence, the *Slot Marker/End-of-Frame* Request is used in conjunction with this request. Any collision that does occur can be further arbitrated using the anticollision mask in an algorithm outlined in the Tag-it Transponder Protocol Reference Manual.

To do an SID poll, the user should:

- Click the button for *SID poll* in the *Commands* window.
- Click on the *Info* flag, if desired, in the *Tag Flags* window.
- *Execute* the command.

**Information:**

The EVM transmitter remains **ON** in order to preserve the tag states changed by the request.




**Request Packet:**

**01 0B 00 03 04 34 00 50 00 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comment |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0B | Packet length = 11 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 34 | Ti SID poll |
| | 00 | Request from reader to tag |
| | 50 | SID poll request |
| | 00 | Mask length |
| EOF | 00 00 | End of frame |

### SID Poll Response

Reader/Tag response (0 through 15 slots) is as follows:

IRQ Status Register [<Tag response if any>]

Example:

Ti SID Poll

80T01N[] Comment: (slot # 0, **80T** end of transmit, **01N** no response interrupt, **[]** no tag response)

60F40E[C0A000D2844102050307] Comment: (slot # 1, **60F** receive data buffer 75% full, **40E** end of receive, [**C0A000D2844102050307**] tag response)

01N[]

01N[]

01N[]

01N[]

01N[]

01N[]

01N[]

01N[]

01N[]

01N[]

01N[]

01N[]

01N[] Comment: (slot # 15, **01N** no response interrupt, **[,40]** < no tag response >

### SID Tag Response

**[C0 A0 00D28441 02 05 03 07] (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| Response Code | C0 | Response from tag to reader |
| Command Code | A0 | SID poll |
| SID | 00D28441 | 4 bytes or 32 bits |
| Chip Mfg ID | 02 05 | (7 bits) = 02h (note: TI = 01b) + chip version (9 bits) = 05h<br>0000 0010 0000 0101 = 16 bits binary<br>0 2 0 5 = 0205 hex |
| Block size | 03 | No. + 1 = 4 (4 bytes or 32 bits) |
| No. of Blocks | 07 | No. + 1 = 8 |

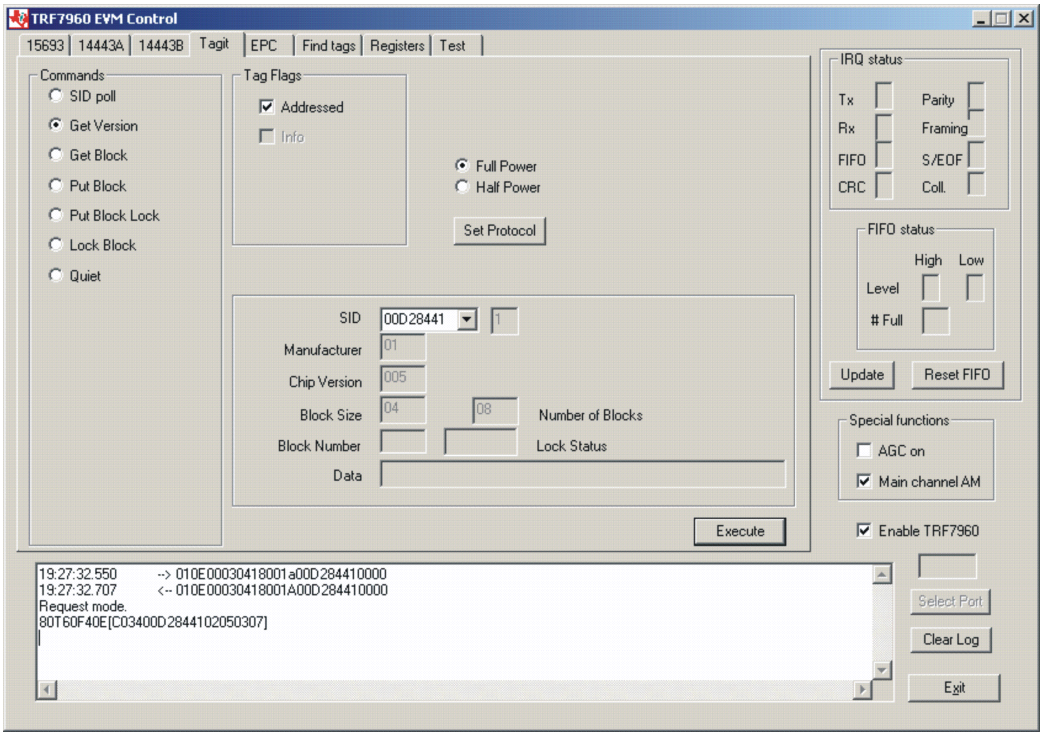
Note: Tag memory is 8 blocks each at 32 bits to equal a total of 256 bits (8 blocks × 32 bits = 256 bits).

### 3.8.2 Get Version

The *Get VERSION* request acquires information on the properties of a responding tag. These properties include IC version and manufacturer information as well as the number and size of memory blocks available.

To get the IC version, the user should:

- Click the button for *Get Version* in the *Commands* window.
- Click on the *Address* flag, if desired, in the *Tag Flags* window.
- *Execute* the command.




**Request Packet:**

**01 0E 00 03 04 18 00 1A 00 D2 84 41 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0E | Packet length = 14 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| | 00 | Request from reader to tag |
| | 1A | Address flag set |
| SID | 00 D2 84 41 | 4 bytes or 32 bits |
| EOF | 00 00 | End of frame |

### Get Version Response

Request mode

80T60F40E[C03400D2844102050307]  Comment: **80T** end of transmit, **60F** receive data buffer 75% full, **40E** end of receive, [**C03400D2844102050307**] tag response shown as follows:

**[C0 34 00D28441 02 05 03 07]**

| Field | Contents | Comments |
|---|---|---|
| Response Code | C0 | Response from tag to reader |
| Command Code | 34 | Get version cmd = 3<br>Address flag set = 4, not set = 0<br>1100 00 0011 0100<br>C 0 3 4 |
| SID | 00D28441 | 4 bytes or 32 bits |
| Chip Mfg. ID | 02 05 | (7 bits) = 02h (note: TI = 01b) + chip version (9 bits) = 05h<br>0000 0010 0000 0101 = 16 bits binary<br>0 2 0 5 = 0205 hex |
| Block Size | 03 | No. + 1 = 4 (4 bytes or 32 bits) |
| No. of Blocks | 07 | No. + 1 = 8 |

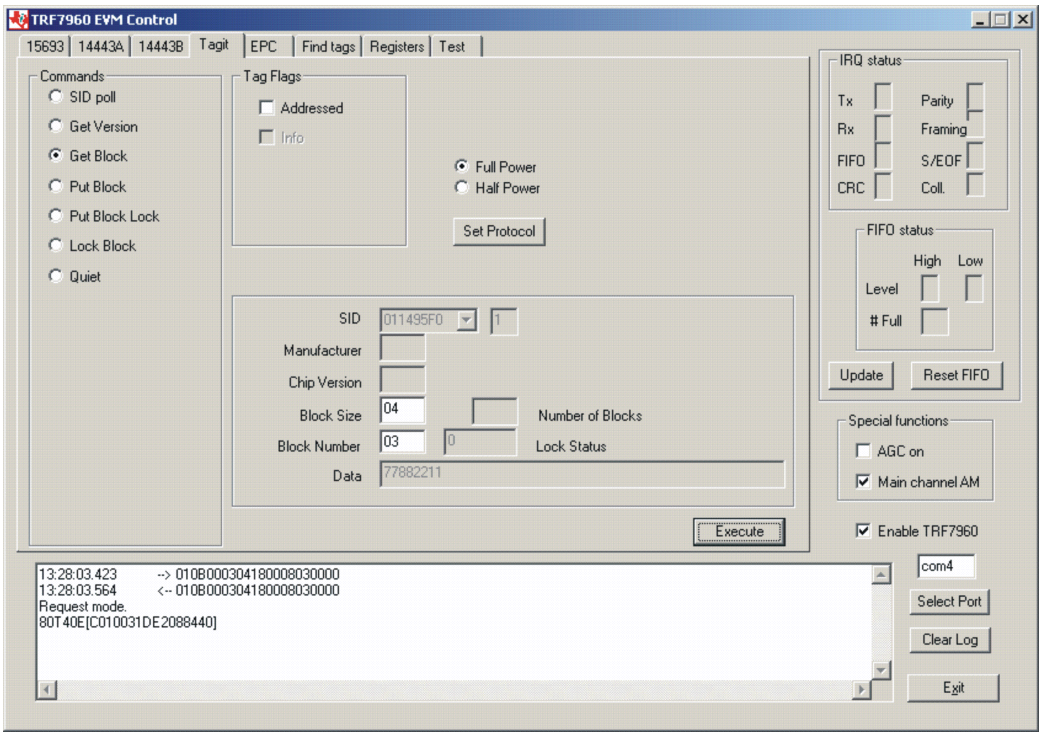
Note: Tag memory is 8 blocks, each containing 32 bits, to equal a total of 256 bits (8 blocks $\times$ 32 bits = 256 bits).

### 3.8.3 Get Block

The *Get Block* request gets the data from one memory block of the responding tag. In addition to this data, a block security status byte is returned. This byte indicates the write protection status of the block specified [e.g., unlocked, (user/factory) locked, etc.].

To get blocks, the user should:

- Click the button for *Get Block* in the *Commands* window.
- Click on the *Address* flag, if desired, in the *Tag Flags* window.
- Enter two hex digits for block size in the *Block Size* field of the *Tag Data* window.
- Enter two hex digits for block number in the *Block Number* field of the *Tag Data* window.
- *Execute* the command.




**Request Packet:**

**01 0B 00 03 04 18 00 08 03 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0B | Packet length = 11 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| | 00 | Request from reader to tag |
| Command Code | 08 | Get Block, nonaddressed = 08<br>addressed = 0A |
| Block Number | 03 | No. + 1 = 4 |
| EOF | 00 00 | End of frame |

### Get Block Response

Request mode

80T40E[C010031DE2088440] Comment: **80T** end of transmit, **40E** end of receive, [**C010031DE2088440**] tag response shown as follows:

| Field | Contents | Comments |
|---|---|---|
| Response Code | C0 | Response from tag to reader Figure 3-2 |
| Command Code | 10 | Get block command Figure 3-2 |
| Block number | 03 | No. + 1 = 4 Figure 3-2 |
| Block data | 1D E2 08 84 | Note: Bits are shifted. Figure 3-3 |
| | 4 | Shifted data byte Figure 3-3 |
| | 0 | Added byte to complete data payload Figure 3-3 |

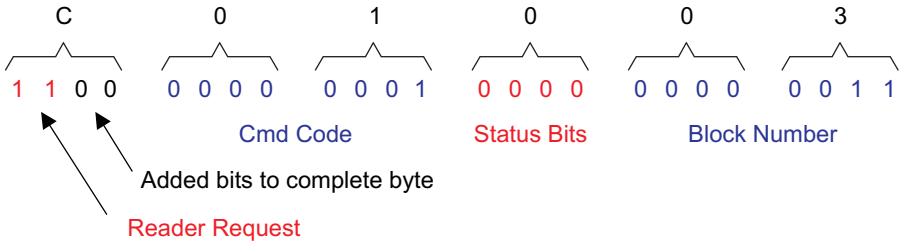



**Figure 3-2. Get Block Response Packet Structure (Part 1)**

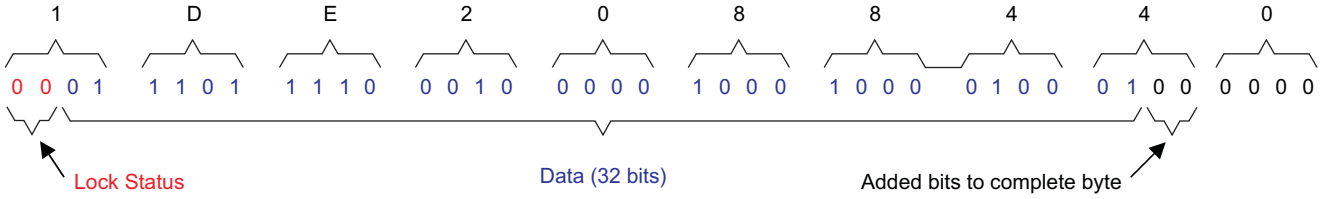


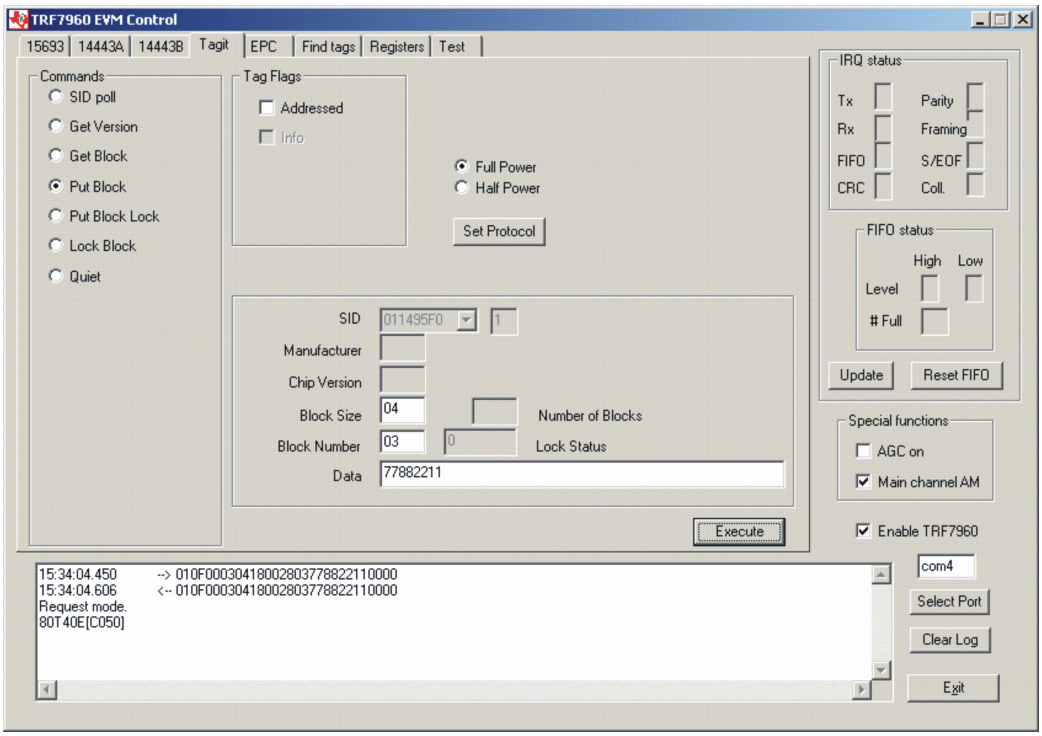

**Figure 3-3. Get Block Response Packet Structure (Part 2)**

### 3.8.4 Put Block

The *Put Block* request writes data to one memory block of the addressed tag(s). In order to successfully write data, the host must know the size of the memory block of the tag. This information is available through the *Get IC Version* Request or *SID Poll* sequence requesting version data. A corrupted response or lack of response from the TRF7960 does not necessarily indicate a failure to perform the write operation. Additionally, multiple tags may process a nonaddressed request.

To put a block (write to a block), the user should:

- Click the button for *Put Block* in the *Commands* window.
- Click on the *Address* flag, if desired, in the *Tag Flags* window.
- Enter two hex digits for block size in the *Block Size* field of the *Tag Data* window.
- Enter two hex digits for block number in the *Block Number* field of the *Tag Data* window.
- Enter the desired data in the *Data* field of the *Tag Data* window.
- *Execute* the command.




**Request Packet:**

**01 0F 00 03 04 18 00 28 03 77 88 22 11 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0F | Packet length = 15 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Command | 18 | Request mode |
| | 00 | Request from reader to tag Figure 3-4 |
| Command Code | 28 | Put block Figure 3-4 |
| Block Number | 03 | No. + 1 = 4 Figure 3-4 |
| Block Data | 77 88 22 11 | 32 bits |
| EOF | 00 00 | End of frame |

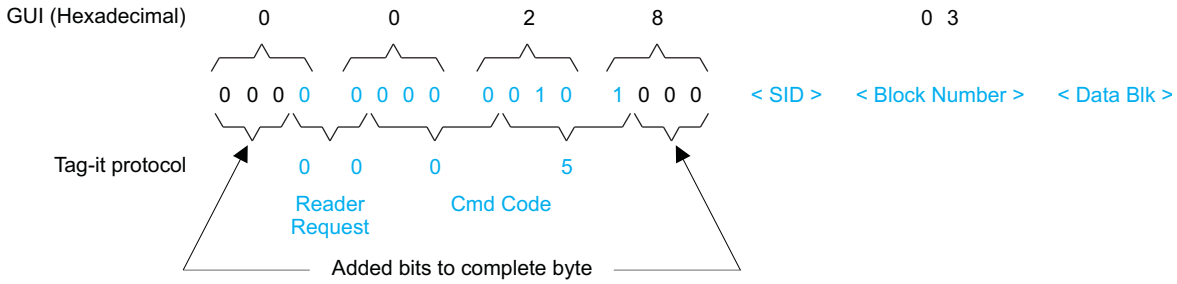



**Figure 3-4. Put Block Request Packet Structure**

### Put Block Response

Request mode

80T40E[C050] Comment: **80T** end of transmit, **40E** end of receive, [**C050**] tag response shown as follows:

| Field | Contents | Comments |
|---|---|---|
| Response Code | C0 | Response from tag to reader Figure 3-5 |
| Command Code | 50 | Put block command Figure 3-5 |

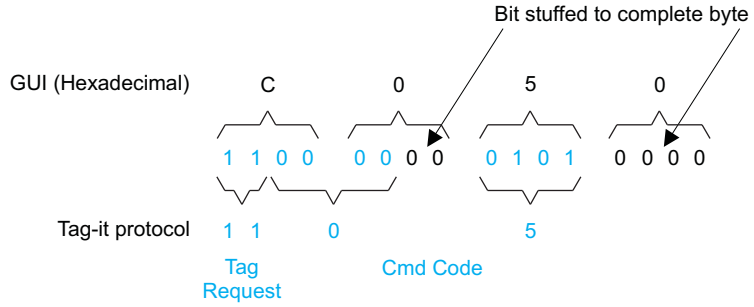



**Figure 3-5. Put Block Response Packet Structure**

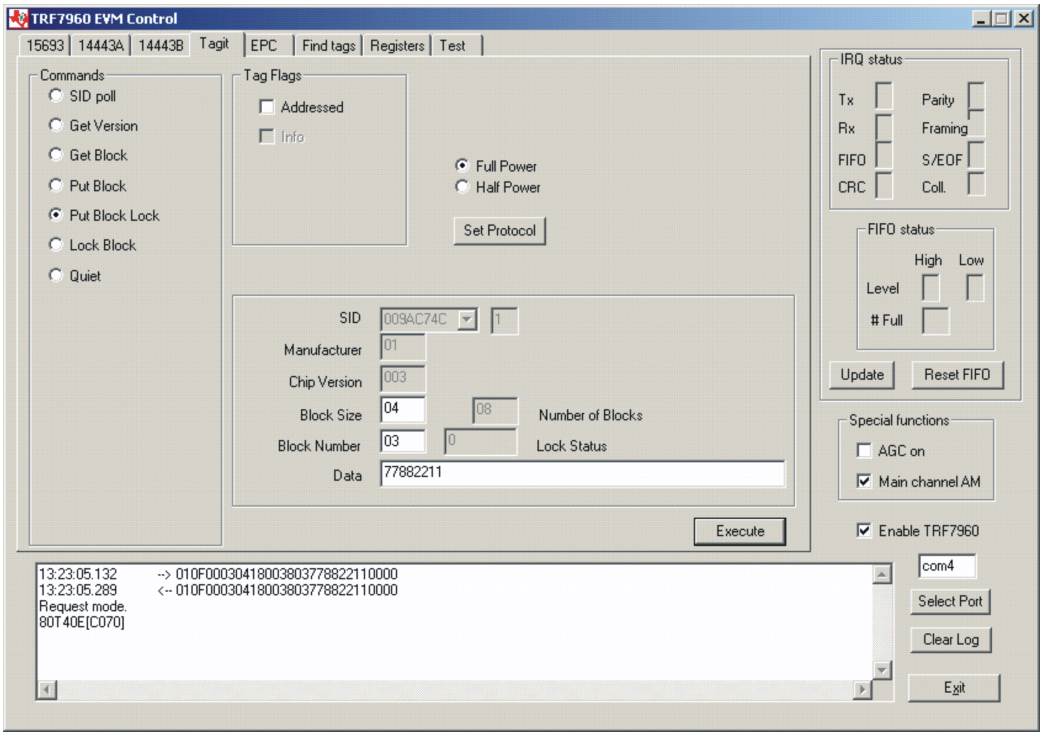
Note: The Tag-It protocol uses both binary and hexadecimal bytes, whereas the GUI uses hexadecimal bytes only.

### 3.8.5 Put Block Lock

The *Put Block Lock* request writes data to one memory block of the addressed tag(s) and locks that block from further write operations. In order to successfully write data, the host must know the size of the memory block of the tag. This information is available through the *Get IC Version* request or *SID Poll* sequence requesting version data. A corrupted response or lack of response does not necessarily indicate a failure to perform the write-lock operation. Additionally, multiple transponders may process a nonaddressed request.

To put a lock block (write to a block and then write protect it), the user should:

- Click the button for *Put Block Lock* in the *Commands* window.
- Click on the *Address* flag, if desired, in the *Tag Flags* window.
- Enter two hex digits for block size in the *Block Size* field of the *Tag Data* window.
- Enter two hex digits for block number in the *Block Number* field of the *Tag Data* window.
- Enter the desired data in the *Data* field of the *Tag Data* window.
- *Execute* the command.




**Request Packet:**

**01 0F 00 03 04 18 00 38 03 77 88 22 11 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0F | Packet length = 15 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Command | 18 | Request mode |
| | 00 | Request from reader to tag Figure 3-6 |
| Command Code | 38 | Put block lock Figure 3-6 |
| Block Number | 03 | No. + 1 = 4 Figure 3-6 |
| Block Data | 77 88 22 11 | 32 bits |
| EOF | 00 00 | End of frame |

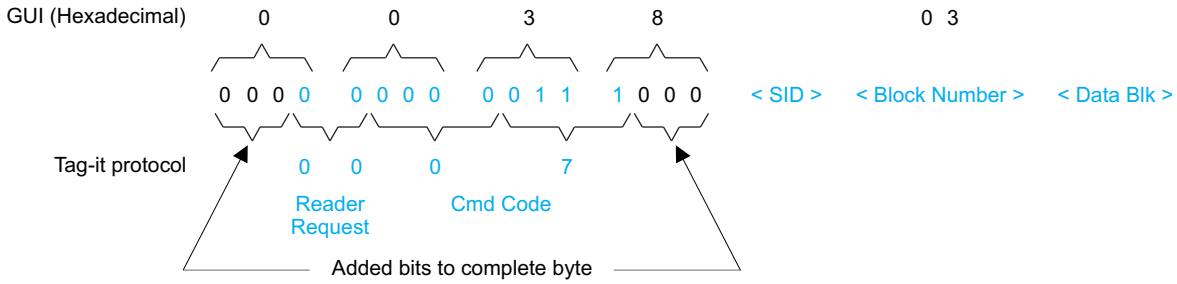



**Figure 3-6. Put Block Lock Request Packet Structure**

### Put Block Lock Response

Request mode

80T40E[C070] Comment: **80T** end of transmit, **40E** end of receive, [**C070**] tag response shown as follows:

| Field | Contents | Comments |
|---|---|---|
| Response Code | C0 | Response from tag to reader Figure 3-7 |
| Command Code | 70 | Put block lock command Figure 3-7 |

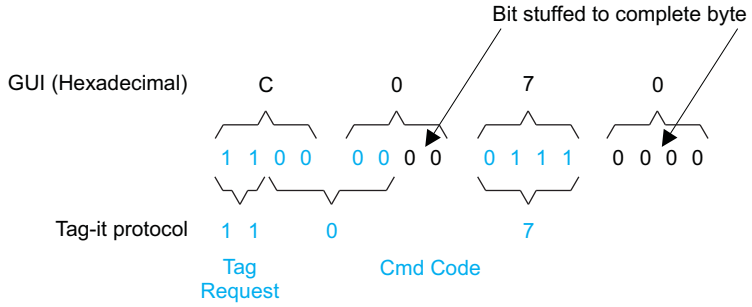



**Figure 3-7. Put Block Lock Response Packet Structure**

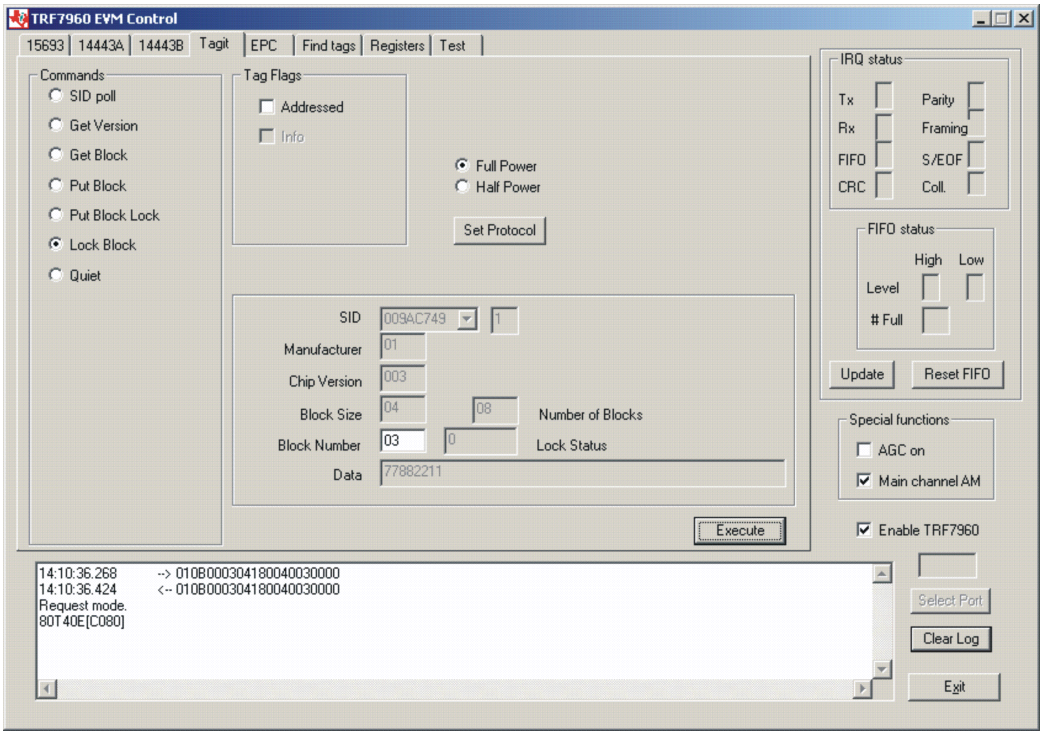
Note: The Tag-It protocol uses both binary and hexadecimal bytes, whereas the GUI uses hexadecimal bytes only.

### 3.8.6 Lock Block

The *Lock Block* request write-protects one memory block of the addressed tag(s). A corrupted response or lack of response does not necessarily indicate a failure to perform the lock operation. Additionally, multiple tags may process a nonaddressed request.

To lock a block (write protect a block), the user should:

- Click the button for *Lock Block* in the *Commands* window.
- Click on the *Address* flag, if desired, in the *Tag Flags* window.
- Enter two hex digits for block number in the *Block Number* field of the *Tag Data* window.
- *Execute* the command.




**Request Packet:**

**01 0B 00 03 04 18 00 40 03 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0B | Packet length = 11 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Command | 18 | Request mode |
| | 00 | Request from reader to tag Figure 3-8 |
| Command Code | 40 | Lock block Figure 3-8 |
| Block Number | 03 | No. + 1 = 4 Figure 3-8 |
| EOF | 00 00 | End of frame |

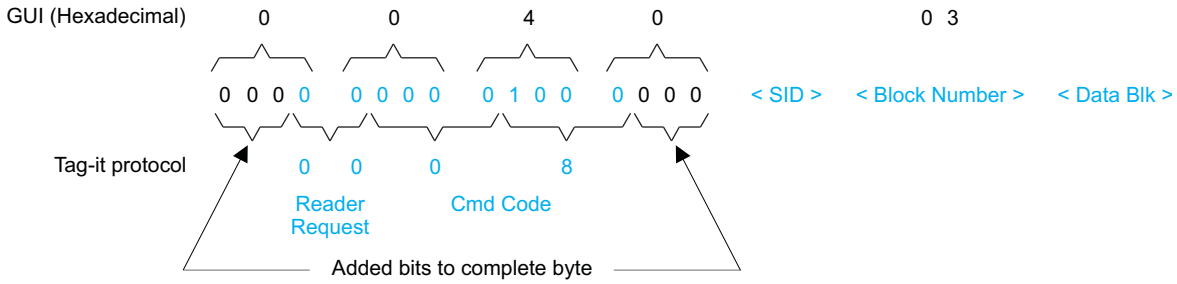



**Figure 3-8. Lock Block Request Packet Structure**

### Lock Block Response

Request mode

80T40E[C080] Comment: **80T** end of transmit, **40E** end of receive, [**C080**] tag response shown as follows:

| Field | Contents | Comments |
|---|---|---|
| Response Code | C0 | Response from tag to reader Figure 3-9 |
| Command Code | 80 | Put block lock command Figure 3-9 |

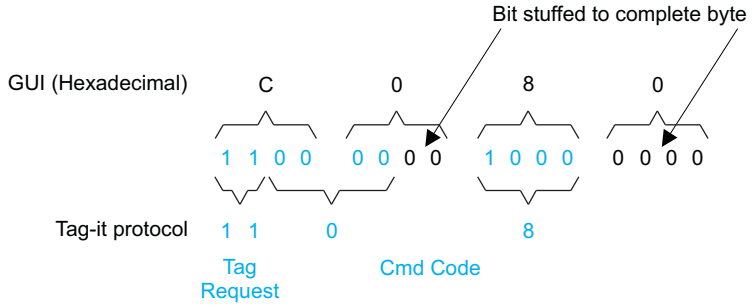



**Figure 3-9. Lock Block Response Packet Structure**

Note: The Tag-It protocol uses both binary and hexadecimal bytes, whereas the GUI uses hexadecimal bytes only.

### 3.8.7 Quiet

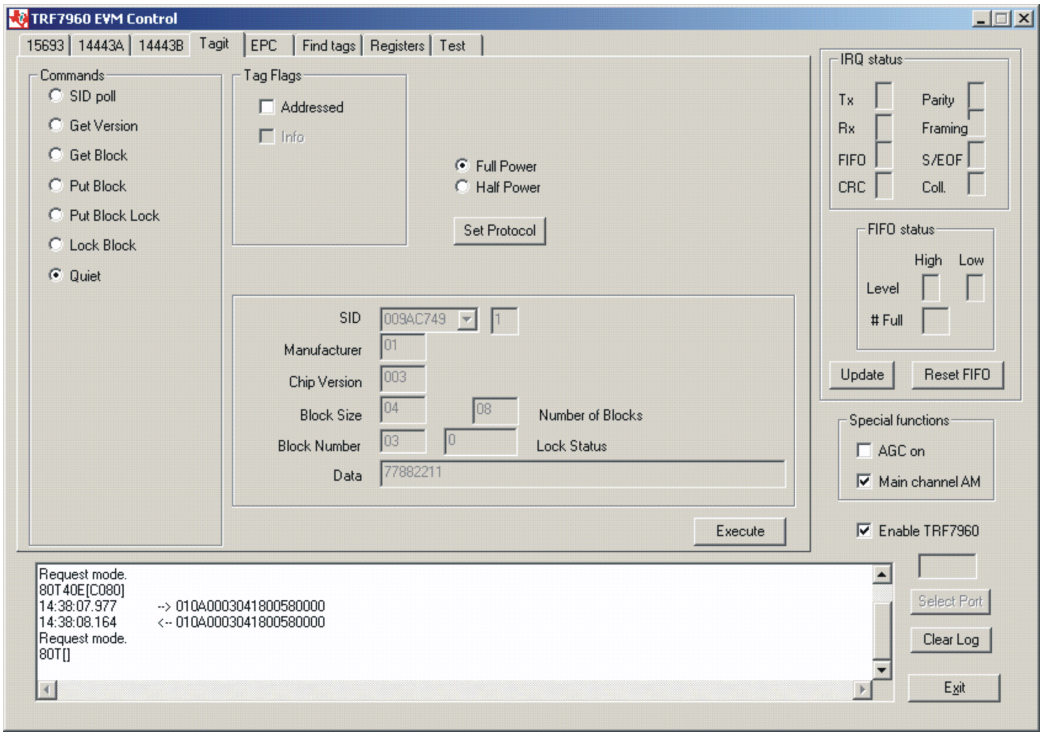
The *Quiet* request is used to silence a tag, preventing it from responding to any nonaddressed or *SID Poll* related requests. The tag does, however, respond to requests with matching SID. As there is no response to this request from the receiving tag, only request status and errors are reported.

**Information:**

The EVM transmitter remains **ON** in order to preserve the tag states changed by the request.

To silence a tag, the user should:

- Click the button for *Quiet* in the *Commands* window.
- Click on the *Address* flag, if desired, in the *Tag Flags* window.
- *Execute* the command.




**Request Packet:**

**01 0A 00 03 04 18 00 58 00 00 (all bytes are continuous; spaces are added for clarity)**

| Field | Contents | Comments |
|---|---|---|
| SOF | 01 | Start of frame |
| Packet Length | 0A | Packet length = 10 bytes |
| Constant | 00 | |
| Begin Data Payload | 03 04 | Start of data payload |
| Firmware Cmd | 18 | Request mode |
| | 00 | Request from reader to tag Figure 3-10 |
| Command Code | 58 | Quiet Figure 3-10 |
| EOF | 00 00 | End of frame |

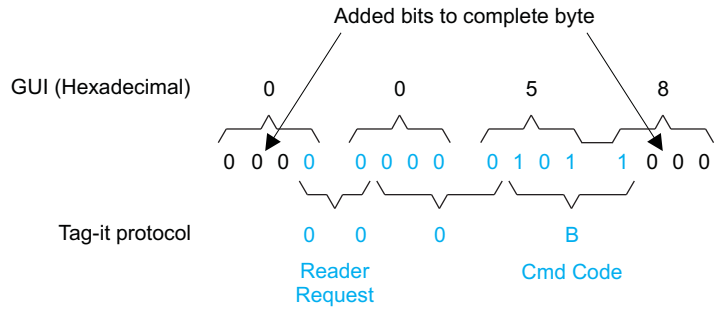



**Figure 3-10. Quiet Request Packet Structure**

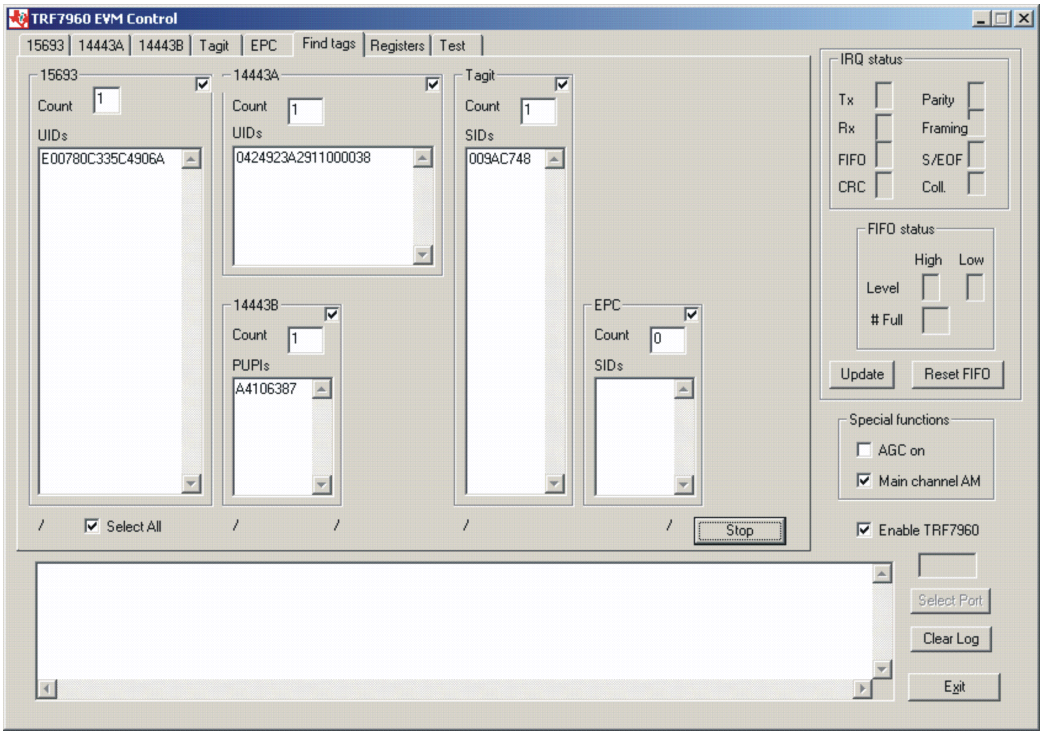
### Quiet (Tag Response)

Request mode

80T[] Comment: **80T** end of transmit, [] no tag response)

## 3.9 Find Tags

The *Find tags* window enables the query of the RF field for all supported tags. It continuously switches from one standard to another and sends an *Inventory* request and displays all the tag labels found within the read range of the reader. The user can choose which protocols to be searched by selecting the appropriate buttons that correspond to the protocol field. This reduces the time associated with cycling through the other standards that are of no interest to the user. If the *Select All* button is checked, all the supported protocols are included in the search operation.

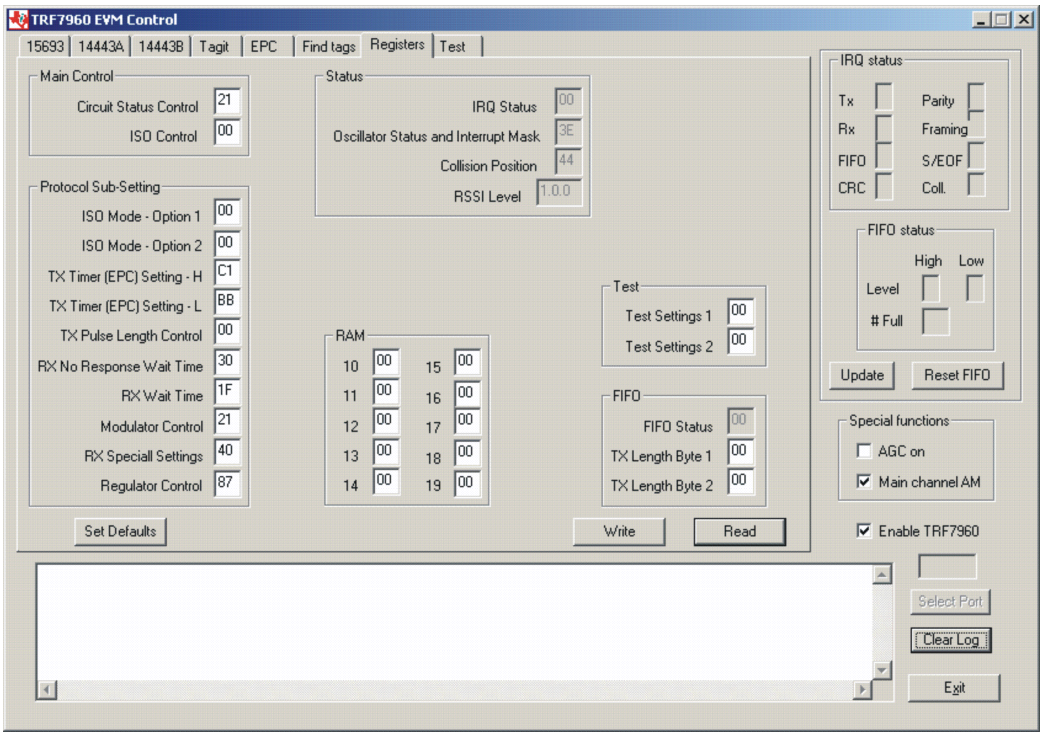



Once the *Run* button is clicked, the window shows all tags found within its reception area, regardless of protocol, if the *Select All* option is checked. Otherwise, it only finds tags of the protocols that are checked. This command runs until the *Stop* button is clicked (shared location with the *Run* button). An indicator for each of the supported standards is active when the particular protocol is running. This clockwise rotating cursor can be found located left of the *Run* button.

This command is recommended for demonstrations, as it requires no specific knowledge of commands/flags for each protocol.

## 3.10 Registers

The content of the registers can be read and written in the *Registers* window. Do not alter the register content unless you are familiar with the functions described in the TFR7960 specifications. If you change the content by mistake, press the *Set Defaults* button.




The register values are updated automatically every time the user enters the *Registers* tab or when the special functions are changed.

## 3.11 Test

If desired, the user can send manual commands by using *Test* tab. Only the *command + parameters* field must be typed in. All other fields in the protocol can be left out:

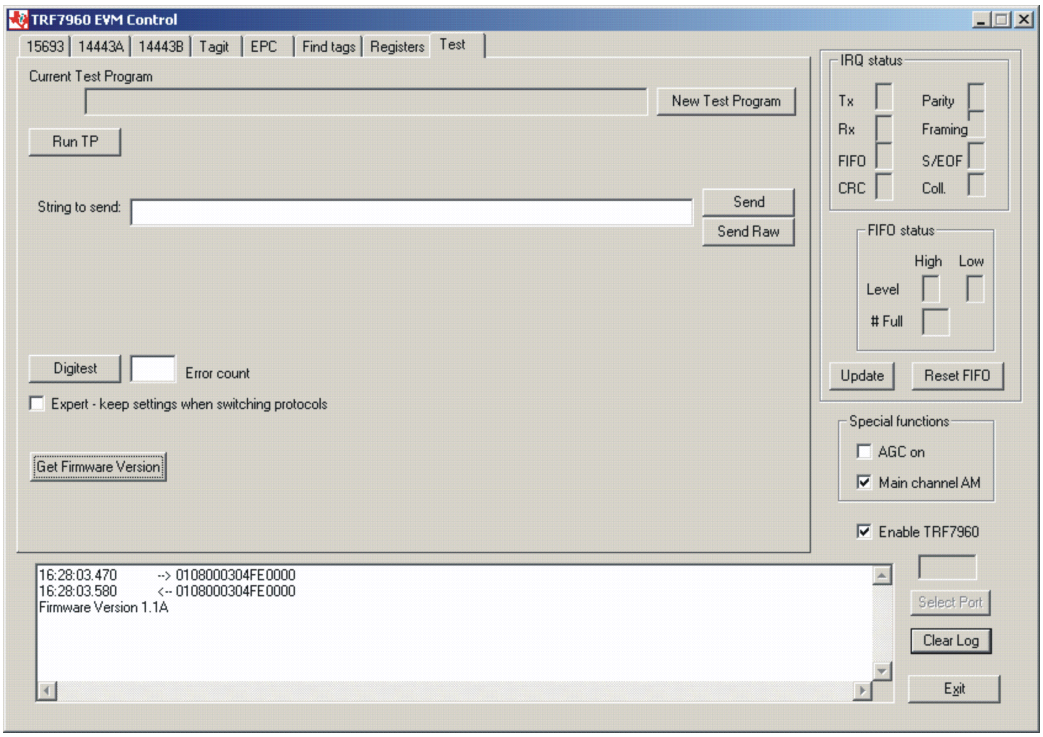
| SOF (0x01) | Number of bytes | 0x00 | 0x0304 | Command + parameters | EOF (0x0000) |
|---|---|---|---|---|---|

The communication starts with SOF (0x01). The second byte defines the number of bytes in the frame including SOF. The third byte should be kept at 0x00, fourth byte at 0x03 and the fifth byte at 0x04. The sixth byte is the command code, which is followed by parameters or data. The communication ends with 2 bytes of 0x00.

| Command | Parameters | Example |
|---|---|---|
| **0x03** TRF796x enable/disable | 0x00 – Reader enable<br>0xFF – Reader disable | 01 09 00 03 04 **03 FF** 0000 |
| **0x0F** Direct mode | | 01 08 00 03 04 **0F** 0000 |
| **0x10** Write single register | Address, data, address, data... | 01 0A 00 03 04 **10 15 67** 0000 |
| **0x11** Write continuous | Address, data, data... | 01 0C 00 03 04 **11 13 67 46 A4** 0000 |
| **0x12** Read single register | Address, address, ... | 01 0B 00 03 04 **12 01 0A 13** 0000 |
| **0x13** Read continuous | NR. of bytes to read, start address | 01 0A 00 03 04 13 05 03 0000 |
| **0x14** Inventory (ISO 15693) | FIFO data | 01 0B 00 03 04 **14 06 01 00** 0000 |
| **0x15** Direct command | Direct command code | 01 09 00 03 04 **15 0F** 0000 |
| **0x16** Write raw | Data or commands ... | 01 10 00 03 04 **16 91 3D 00 40 AA BB CC DD** 0000 |
| **0x18** Request command ISO 15693, Tag-it, 14443B Halt | Flags, command code, data,... (as specified in ISO and Tag-it) | 01 0B 00 03 04 **18 06 20 01** 0000 |
| **0x34** SID poll (Tag-it) | Flags, command code, mask (as specified in Tag-it) | 01 0B 00 03 04 **34 00 50 00** 0000 |
| **0x54** Begin round (EPC) | No. of slots | 01 09 00 03 04 **54 03** 0000 |
| **0x55** Close slot sequence (EPC) | | 01 08 00 03 04 **55** 0000 |
| **0xA0** REQA (14443A) | | 01 08 00 03 04 **A0** 0000 |
| **0xA2** Select (14443A) | CID | 01 0D 00 03 04 **A2 11 22 33 44 44** 0000 |
| **0xB0** REQB (14443B) | | 01 08 00 03 04 **B0** 0000 |
| **0xF0** AGC selection | 0x00 – AGC enable<br>0xFF – AGC disable | 01 09 00 03 04 **F0 FF** 0000 |
| **0xF1** AM/PM input selection | 0x00 – FM input<br>0xFF – AM input | 01 09 00 03 04 **F1 00** 0000 |
| **0xFE** Get Version | | 01 08 00 03 04 **FE** 0000 |

### 3.11.1 Expert Mode Selection

There is an added feature that allows the user to keep the user adjusted register settings without having the individual *set protocol* do it for them. Currently, a user wanting to test to a particular standard would go to the desired tab and then do a *set protocol*, which configures all the registers to a default value. Once this is done, the user can go to the *Test* tab, select the *Expert* check box and then go to the *Register* tab to make the necessary modifications. This allows the reader to keep the existing register settings even if the user must go back to the other protocol (15693, 14443A, etc.) tabs to do some of the preset commands.



# ISO/IEC 15693 Reference Material

## A.1 UID Format

The tags are uniquely identified by a 64-bit unique identifier (UID). This is used for addressing each tag uniquely and individually during the anticollision loop, and for one-to-one exchange between a reader and a tag.

The format of the UID is shown below:

| Bits 64 to 57 | Bits 56 to 49 | Bits 48 to 1 |
|---|---|---|
| E0 | Manufacturer code | IC serial number |

The UID is composed of:

- The 8 MSBs, which are *E0.*
- The 8-bit IC manufacturer code
- A unique serial number of 48 bits assigned by the IC manufacturer

## A.2 Tag Memory Organization

Tag memory is organized into blocks of bytes. Addressing is by block only. There is no individual byte addressing for read or write; the whole block is accessed. It is analogous to a spreadsheet with rows and columns, where addressing accesses a whole row at once.

The format of tag memory is shown as follows:

| Bits 16 to 14 | Bits 13 to 9 | Bits 8 to 1 |
|---|---|---|
| RFU | Block size in bytes | Number of blocks |

- Block size is expressed in 5 bits, allowing up to 32 bytes, i.e., 256 bits. It is one less than the actual number of bytes. E.g., a value of *1F* indicates 32 bytes; a value of *00* indicates 1 byte.
- Number of blocks is defined in 8 bits, allowing up to 256 blocks. It is one less than the actual number of blocks. E.g., a value of *FF* indicates 256 blocks; a value of *00* indicates 1 block.
- The 3 most-significant bits are reserved for future use and are set to zero.

**Information:**

This addressing scheme limits the total storage of the tag to 8K bytes.

**Note:**

The software GUI that you use may be storing data in ASCII, rather than hexadecimal. This cuts the storage capacity of the tag in half, because 8 bits are required for each ASCII character instead of 4 with hexadecimal. It may require a data stream capture instrument to differentiate.

## A.3 Flag Definitions

- *High Data Rate:* the default data rate is used for maximum detection range. If *High Data Rate* is selected in the *Tag Flags* window, communication with the tag is faster, but the range is reduced.
- *AFI is present:* The default setting for the AFI (Application Family Identifier – see Section A.6) is off. If *AFI is present* is selected in the *Tag Flags* window, AFI is enabled in commands and responses.
- *One Slot:* the definition of *slot*, as used in the software, is the number of tags that may be received at a time. The default is 16. If only *One Slot* is selected in the *Tag Flags* window, the algorithm detects a flag sooner, but stops after detecting the first tag. Other tags in the reception range of the reader are ignored.

**Request Flags Bits 1 to 4**
**(Ref.: ISO 15693-3:2000(E), Section 7.3.1 Table 3, Page 9)**

| Bit | Flag Name | Value | Description |
|---|---|---|---|
| b1 | Subcarrier flag | 0 | A single subcarrier is used by the tag. |
| | | 1 | Two subcarriers are used by the tag. |
| b2 | Data rate flag | 0 | Low data rate |
| | | 1 | High data rate |
| b3 | Inventory flag | 0 | Flags 5 to 8 meaning in following tables (points to table 4 in ISO 15693-3 protocol) |
| | | 1 | Flags 5 to 8 meaning in following tables (points to table 5 in ISO 15693-3 protocol) |
| b4 | Protocol extension flag | 0 | No protocol format extension |
| | | 1 | Protocol format is extended. Reserved for future use. |

**Request Flags Bits 5 to 8 when inventory flag IS NOT set**
**(Ref.: ISO 15693-3:2000(E), Section 7.3.1 Table 4, Page 10)**

| Bit | Flag Name | Value | Description |
|---|---|---|---|
| b5 | Select flag | 0 | Request executed by any tag according to the setting of *Address* flag. |
| | | 1 | Request executed only by tag in selected state. The *Address* flag is set to 0 and the UID field is not included in the request. |
| b6 | Address_flag | 0 | Request is not addressed. UID field is not included. It can be executed by any tag. |
| | | 1 | Request is addressed. UID field is included. It is executed only by the tag whose UID matches the UID specified in the request. |
| b7 | Option_flag | 0 | Meaning is defined by the command description. It is set to 0 if not otherwise defined by the command. |
| | | 1 | Meaning is defined by the command description. |
| b8 | RFU | 0 | Reserved for future use |

**Request Flags Bits 5 to 8 when inventory flag IS set**
**(Ref.: ISO 15693-3:2000(E), Section 7.3.1 Table 5, Page 10)**

| Bit | Flag Name | Value | Description |
|---|---|---|---|
| b5 | AFI_flag | 0 | AFI field is not present. |
| | | 1 | AFI field is present. |
| b6 | Nb_slots_flag | 0 | 16 slots |
| | | 1 | 1 slot |
| b7 | Option_flag | 0 | Meaning is defined by the request description. It is set to 0 if not otherwise defined by the request. |
| | | 1 | Meaning is defined by the request description. |
| b8 | RFU | 0 | Reserved for future use |

**Response Flags**
**(Ref.: ISO 15693-3:2000(E), Section 7.4.1 Table 6, Page 11)**

| Bit | Flag Name | Value | Description |
|---|---|---|---|
| b1 | Error flag | 0 | No error |
| | | 1 | Error detected. Error code is in the *Error* field of response. |
| b2 | RFU | 0 | Reserved for future use |
| b3 | RFU | 0 | Reserved for future use |
| b4 | Extension flag | 0 | No protocol format extension. |
| | | 1 | Protocol format is extended. Reserved for future use. |
| b5 | RFU | 0 | Reserved for future use |
| b6 | RFU | 0 | Reserved for future use |
| b7 | RFU | 0 | Reserved for future use |
| b8 | RFU | 0 | Reserved for future use |

## A.4 Error Codes
## (Ref.: ISO 15693-3:2000(E), Section 7.4.2 Table 7, Page 12)

| Error Code | Meaning |
|---|---|
| 01 | The request is not supported, i.e., the request code is not recognized. |
| 02 | The request is not recognized, for example: a format error occurred. |
| 03 | The request option is not supported. |
| 0F | Error with no information given or a specific error code is not supported. |
| 10 | The specified block is not available (does not exist). |
| 11 | The specified block is already locked and thus cannot be locked again. |
| 12 | The specified block is locked and its content cannot be changed. |
| 13 | The specified block was not successfully programmed. |
| 14 | The specified block was not successfully locked. |
| A0–DF | Custom request error codes. |
| All others | Reserved for future use |

## A.5 ISO15693 Commands That Must Be Supported by Third-Party Readers for Texas Instruments Endorsement

| ISO15693 COMMANDS | TI TAG TYPES | | | |
|---|---|---|---|---|
| | Request Code | Standard (256-Bit) | Pro (256-Bit) | Plus (2K-Bit) |
| **MANDATORY COMMANDS (ISO15693)** | | | | |
| Inventory | 0x01 | √ | √ | √ |
| Stay quiet | 0x02 | √ | √ | √ |
| **OPTIONAL COMMANDS (ISO15693)** | | | | |
| Read single block | 0x20 | √ | √ | √ |
| Write single block | 0x21 | √ | √ | √ |
| Lock block | 0x22 | √ | √ | √ |
| Read multiple blocks | 0x23 | | | √ |
| Select | 0x25 | | | √ |
| Reset to ready | 0x26 | | | √ |
| Write AFI | 0x27 | | | √ |
| Lock AFI | 0x28 | | | √ |
| Write DSFID | 0x29 | | | √ |
| Lock DSFID | 0x2A | | | √ |
| Get system information | 0x2B | | | √ |
| Get multiple-block security status | 0x2C | | | √ |
| **TEXAS INSTRUMENTS CUSTOM COMMANDS** | | | | |
| Write two blocks | 0xA2 | | | √ |
| Lock two blocks | 0xA3 | | | √ |
| Kill | 0xA4 | | √ | |
| Write single block password | 0xA5 | | √ | |

## A.6 Application Family Identifier (AFI) Definitions

| AFI Most Significant Nibble | AFI Least Significant Nibble | Meaning Tags Respond From | Examples/Note |
|---|---|---|---|
| 0 | 0 | All families and subfamilies | No applicable reselection |
| X | 0 | All subfamilies of family X | Wide applicable preselection |
| X | Y | Only the Yth subfamily of family X | |
| 0 | Y | Proprietary subfamily Y only | |
| 1 | 0, Y | Transport | Mass transit, bus, airline |
| 2 | 0, Y | Financial | IEP, banking, retail |
| 3 | 0, Y | Identification | Access control |
| 4 | 0, Y | Telecommunication | Public telephony, GSM |
| 5 | 0, Y | Medical | |
| 6 | 0, Y | Multimedia | Internet services |
| 7 | 0, Y | Gaming | |
| 8 | 0, Y | Data storage | Portable files |
| 9 | 0, Y | Item management | |
| A | 0, Y | Express parcels | |
| B | 0, Y | Postal services | |
| C | 0, Y | Airline bags | |
| D | 0, Y | RFU | Reserved for future use |
| E | 0, Y | RFU | Reserved for future use |
| F | 0, Y | RFU | Reserved for future use |

## B.1 Response Flags

| Bit | Value | Meaning |
|---|---|---|
| 0 | 0 | No error |
| | 1 | Error |
| 1 | 0 | Reserved |
| 2 | 0 | Nonaddressed |
| | 1 | Addressed |
| 3 | 0 | Format type |
| 4 | 0 | Unused |
| 5 | | |
| 6 | | |
| 7 | | |

## B.2 Status Flag (Response Frame)

| Bit | Function |
|---|---|
| 0 | Exception |
| 1 | More |
| 2 | Emulation |
| 3 | Auto Repeat |
| 4 | BCC |
| 5 | Reserved |
| 6 | |
| 7 | |

## B.3 Control Flags (Request Frame)

| Bit | Function |
|---|---|
| 0 | Reserved |
| 1 | More |
| 2 | Emulation |
| 3 | Auto Repeat |
| 4 | BCC |
| 5 | Reserved |
| 6 | |
| 7 | |

## EVALUATION BOARD/KIT IMPORTANT NOTICE

Texas Instruments (TI) provides the enclosed product(s) under the following conditions:

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. Persons handling the product(s) must have electronics training and observe good engineering practice standards. As such, the goods being provided are not intended to be complete in terms of required design-, marketing-, and/or manufacturing-related protective considerations, including product safety and environmental measures typically found in end products that incorporate such semiconductor components or circuit boards. This evaluation board/kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.

Should this evaluation board/kit not meet the specifications indicated in the User's Guide, the board/kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies TI from all claims arising from the handling or use of the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge.

EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

TI currently deals with a variety of customers for products, and therefore our arrangement with the user **is not exclusive.**

TI assumes **no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein.**

Please read the User's Guide and, specifically, the Warnings and Restrictions notice in the User's Guide prior to handling the product. This notice contains important safety information about temperatures and voltages. For additional information on TI's environmental and/or safety programs, please contact the TI application engineer or visit www.ti.com/esh.

No license is granted under any patent right or other intellectual property right of TI covering or relating to any machine, process, or combination in which such TI products or services might be or are used.

## FCC Warning

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

## EVM WARNINGS AND RESTRICTIONS

It is important to operate this EVM within the input voltage range of 5 V.

Exceeding the specified input range may cause unexpected operation and/or irreversible damage to the EVM. If there are questions concerning the input range, please contact a TI field representative prior to connecting the input power.

Applying loads outside of the specified output range may result in unintended operation and/or possible permanent damage to the EVM. Please consult the EVM User's Guide prior to connecting any load to the EVM output. If there is uncertainty as to the load specification, please contact a TI field representative.

During normal operation, some circuit components may have case temperatures greater than 40°C. The EVM is designed to operate properly with certain components above 40°C as long as the input and output ranges are maintained. These components include but are not limited to linear regulators, switching transistors, pass transistors, and current sense resistors. These types of devices can be identified using the EVM schematic located in the EVM User's Guide. When placing measurement probes near these devices during operation, please be aware that these devices may be very warm to the touch.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| RFID | www.ti-rfid.com | Telephony | www.ti.com/telephony |
| Low Power Wireless | www.ti.com/lpw | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265