Perfection in Automation
www.br-automation.com

# Technical Description
## 5E9010.29

| | |
|---|---|
| Date: | 20.07.2015 |
| Short text: | Transponder-Reader 13,56MHz 22mm USB |

# I Version information

| Version | Date | Comment | Author |
|---|---|---|---|
| 1.0 | 01.04.2010 | First edition | M. Hochländer |
| 1.1 | 15.04.2010 | Second edition (Port settings included) | M. Hochländer |
| 1.2 | 21.04.2010 | Third edition (Note of samples in AS included) | M. Hochländer |
| 1.3 | 04.11.2010 | Fourth edition (Note in Chapter 4 included) | M. Hochländer |
| 1.4 | 08.03.2011 | Fifth edition (Stay Quiet Command updated) | M. Hochländer |
| 1.5 | 22.12.2011 | Sixth edition (Samples updated) | M. Hochländer |
| 1.6 | 21.11.2014 | Insert information by " Read multiple blocks" | Anna Sigl |
| 1.7 | 24.02.2015 | " Read multiple blocks" --> Option Flag | Anna Sigl |
| 1.8 | 11.06.2015 | Correction commando syntax | Anna Sigl |
| 1.9 | 20.07.2015 | Correction rated current | Anna Sigl |

**Table 1: Versions**

# II Distributors

| Name | Company, Department | Amount | Comment |
|---|---|---|---|
| Günter Schuster | Bernecker + Rainer, cHMI Technical Manager | 1 | |
| Michael Hochländer | Bernecker + Rainer, cHMI Project Development | 1 | |
| | | | |
| | | | |

**Table 2: Distributors**

# III Organization of safety notices

The latest version of the User's Manual can be downloaded as pdf file from the B&R homepage www.br-automation.com

Safety guidelines in this document are organized as follows:

| Safety notice | Description |
|---|---|
| Danger! | Disregarding the safety regulations and guidelines can be life-threatening. |
| Warning! | Disregarding the safety regulations and guidelines can result in severe injury or major damage to material. |
| Caution! | Disregarding the safety regulations and guidelines can result in injury or damage to material. |
| Information: | Important information for preventing errors. |

**Table 3: Safety notices**

# IV Table of contents

# 1 General information

## Information:

**B&R does its best to keep technical descriptions as current as possible. The latest version of the technical description can be downloaded in PDF format from the B&R homepage www.br-automation.com**

## 1.1 Model numbers

### 1.1.1 5E9010.29

| Model number | Description |
|---|---|
| 5E9010.29 | **Transponder-Reader 13,56MHz 22mm USB** |

**Table 4: Model number 5E9010.29**

# 2 Transponder-Reader

## 2.1 General information

With the transponderreader/writer 5E9010.29 you can read or write to a transponder which is compatible to the standard ISO15693.
The memory size depends of the used transponder.
B&R offers a transponder with ICODE SLI – Chip and a size of 1K.
Data are stored in an EEPROM with a memory capacity of 1024 bit and is organized in 32 blocks of 4 bytes each (1 block = 32 bits). The higher 28 blocks contain user data and the lowest 4 blocks contain the unique identifier, the write access conditions and special data like AFI and DSFID.

> ### Information
>
> **Further informations you can find in the datasheet of the used chip.**

## 2.2 5E9010.29

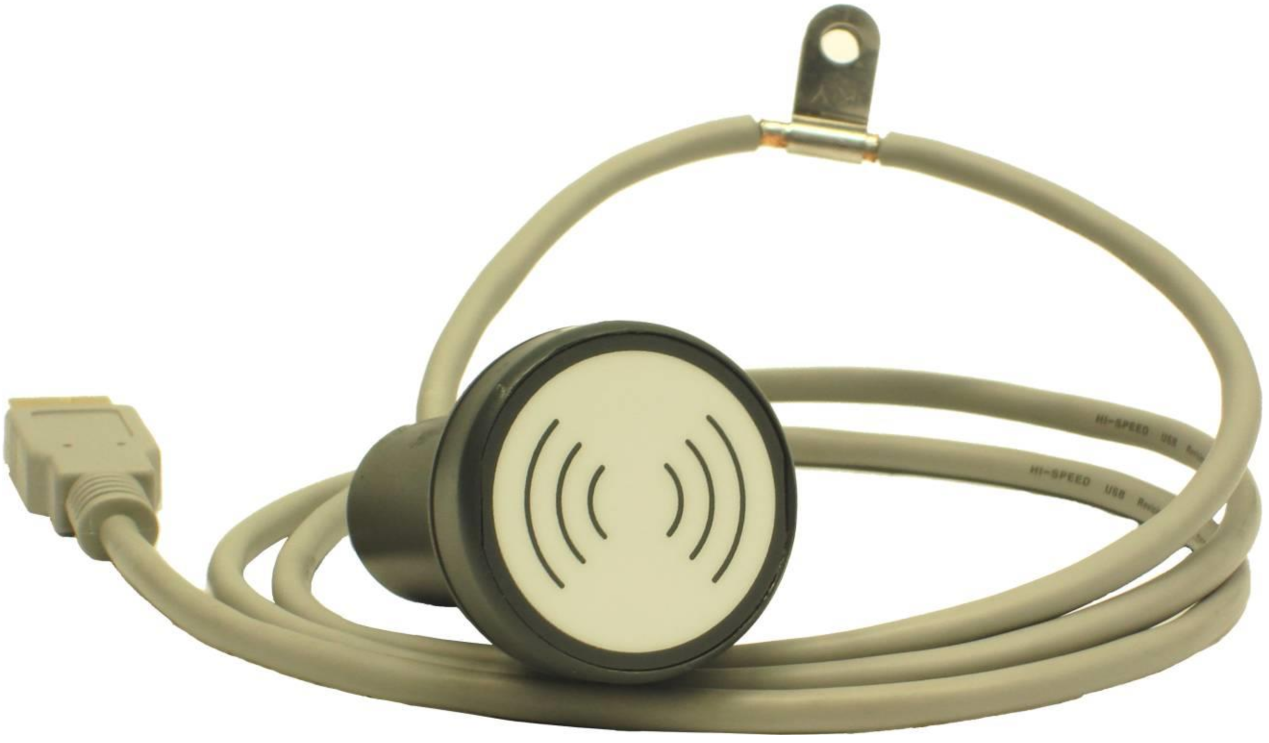
| Catalog photo - Automation Panel 5E9010.29 |
|---|
|  |

**Figure 1: Catalog photo**

## 2.3 Views 5E9010.29

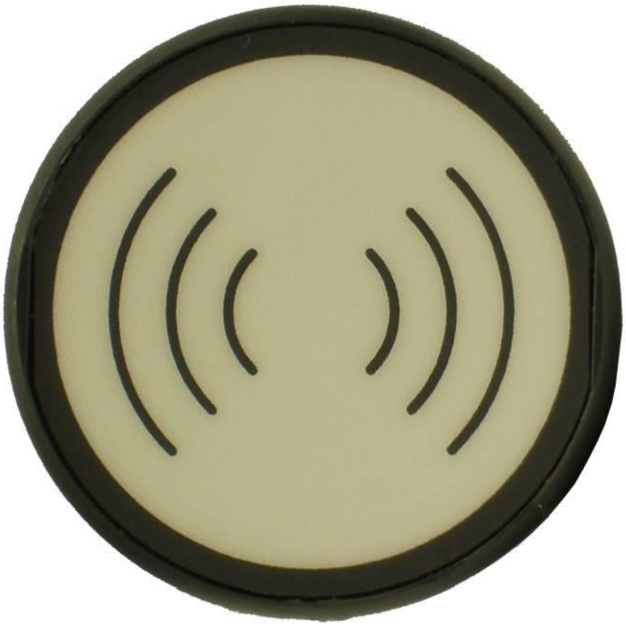
| Front view - Automation Panel 5E9010.29 |
|---|
|  |

**Figure 2: Front view**

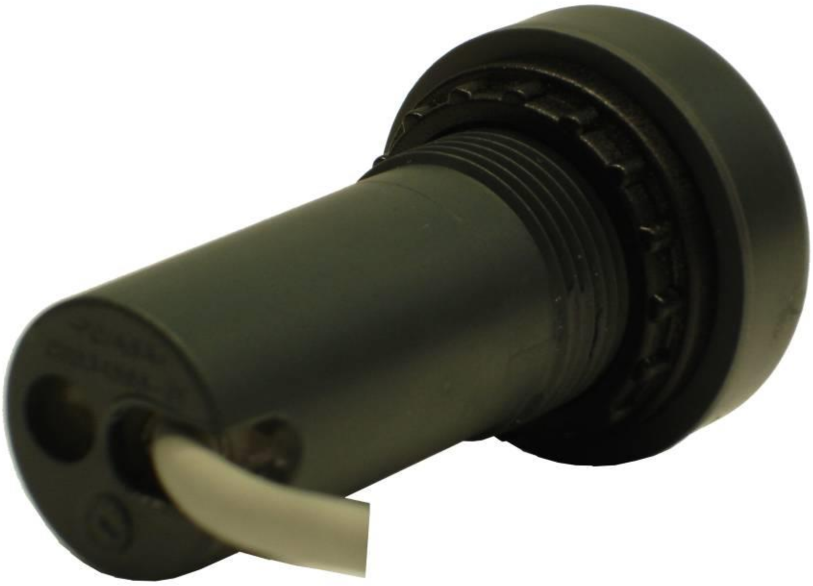
| Rear view - Automation Panel 5E9010.29 |
|---|
|  |

**Figure 3: Rear view**

## 2.4 Interfaces

### 2.4.1 USB

| 4-pin connector | | |
|---|---|---|
| **Comment** | **Description** | |
| Type | USB 2.0 | |
| Transfer speed | Low speed (1.5 MBit/s), full speed (12 MBit/s) | |
| Connection | Type A | |
| | | |

**Table 5: USB**

## 2.5 Stickers

### 2.5.1 Serial number sticker

#### 2.5.1.1 General information

Each B&R device is given a unique serial number sticker with a barcode that allows the device to be clearly identified.

#### 2.5.1.2 Design / dimensions

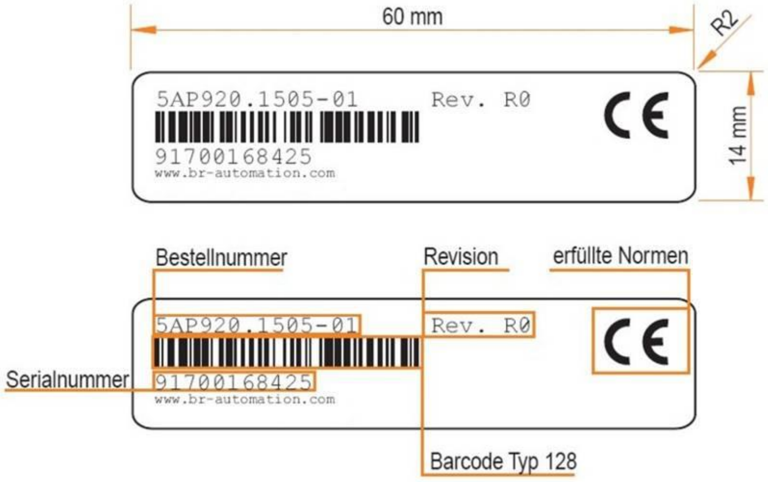
| Serial number sticker |
|---|
|  |

**Figure 4: Design/dimensions - Serial number sticker**

## 2.6 Dimensions

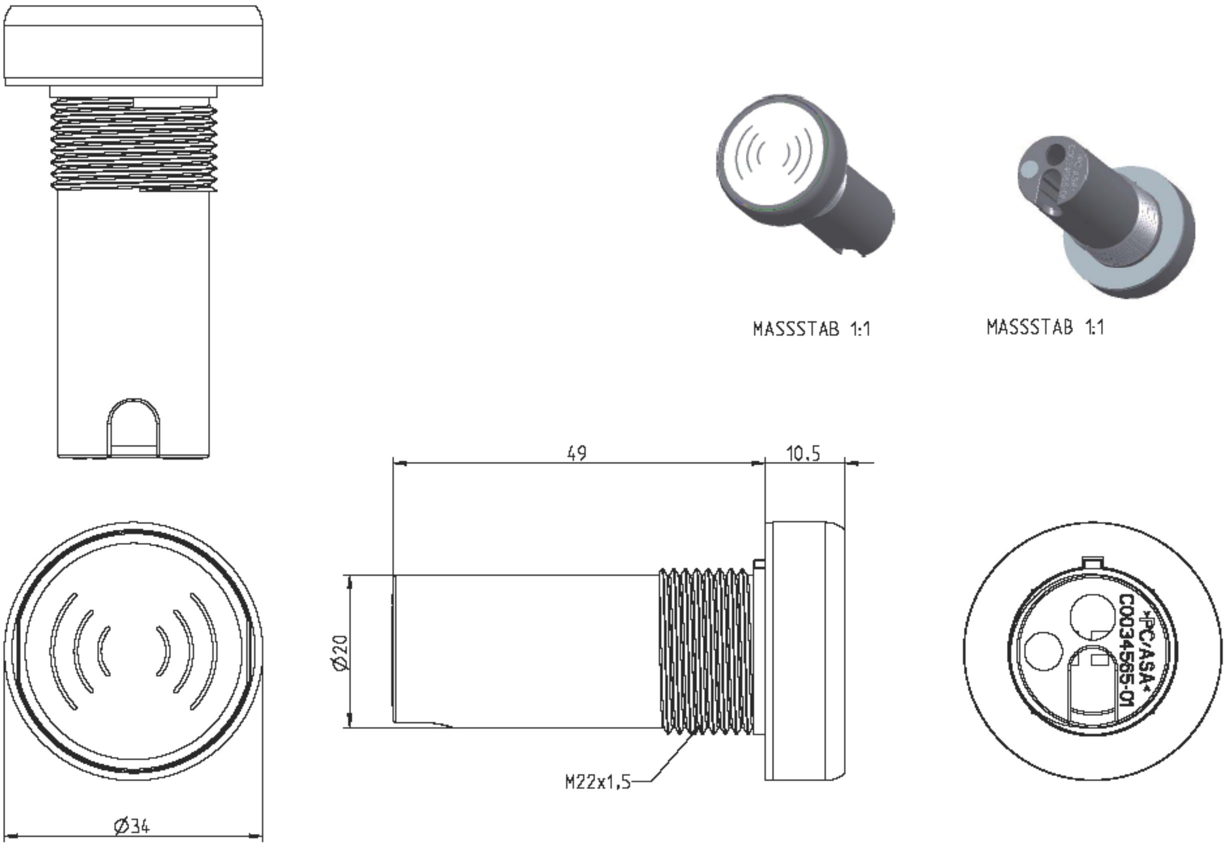
| Dimensions – 5E9010.29 |
|---|
|  |

**Figure 5: Dimensions**

## 2.7 Foil design

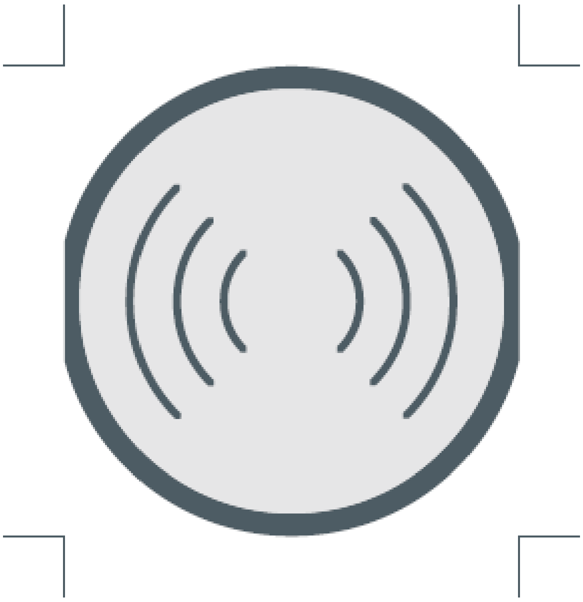
| Foil design – 5E9010.29 |
|---|
|  |

**Figure 6: Foil design**

## 2.8 Technical data

| **Features** | **5E9010.29** |
|---|---|
| USB interface | |
| Type | USB 2.0 |
| Transfer speed | Low speed (1.5 MBit/s), full speed (12 MBit/s) |
| Connection | Type A |
| **Electrical characteristics** | |
| Power supply | |
| Rated voltage | 5 VDC ± 25% |
| Rated current | Maximum 150 mA |
| Electrical isolation | No |
| **Mechanical characteristics** | |
| Outer dimensions | |
| Width | Approx. 34 mm |
| Height | Approx. 34 mm |
| Depth | Approx. 49,5 mm |
| Front / Housing | |
| Frame | PC/ASA (LuranS KR2867C-WU) |
| Membrane | Polyester |
| Design | Customer-specific |
| Gasket | Flat gasket around front |
| Weight | Approx. 0,06 kg |
| **Environmental characteristics** | |
| Ambient temperature | |
| Operation | 0°C .. +50°C |
| Storage | -25°C .. +60°C |
| Transport | -25°C .. +60°C |
| Relative humidity | |
| Operation / Storage / Transport | 5% to 90%, non-condensing |
| Vibration | |
| Operation (continuous) | 2 - 9 Hz: 1.75 mm amplitude / 9 - 200 Hz: 0.5 g |
| Operation (occasional) | 2 - 9 Hz: 3.5 mm amplitude / 9 - 200 Hz: 1 g |
| Storage | 2 - 8 Hz: 7.5 mm amplitude / 8 - 200 Hz: 2 g / 200 - 500 Hz: 4 g |
| Transport | 2 - 8 Hz: 7.5 mm amplitude / 8 - 200 Hz: 2 g / 200 - 500 Hz: 4 g |
| Shock | |
| Operation | 15 g, 11 ms |
| Storage | 30 g, 15 ms |
| Transport | 30 g, 15 ms |
| Protection type | IP65 / NEMA 250 type 4X, dust and sprayed water protection |
| Altitude | Max. 3000 m[1)] |

**Table 6: Technical data**

Derating the maximum ambient temperature - typically 1°C per 1000 meters (from 500 meters above sea level).

# Information

**USB 2.0 is supported up to a cable length of 5 meters**

# 3 Operating the transponder with several OS.

A driver from FTDI is necessary for operation (not for AutomationStudio). This creates a virtual COM port that the transponder uses for communication.
This driver can be downloaded for free from FTDI's support page.
http://www.ftdichip.com/Drivers/VCP.htm
The used chip is FT232R.
The connection is established with e.g. HyperTerminal.

**Port Settings:**
Bits per second: 115200
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

# 4 Protocol, Commands, Appendix

## Information:

**The following command examples may include settings that do not work with all transponders available.**
**For the supported settings please refer to the respective datasheet of the transponder.**

## 4.1 Host to Reader Protocol

The communication from host to reader is organized into frames. Each frame consists of 6 fields:

| | | | | | |
|---|---|---|---|---|---|
| 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.
**All commands are encoded in ASCII and must be terminated with “\n” after EOF.**

## 4.2 Set Protocol

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 00 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),<br>write 21 (RF output active, + 5VDC operation) |
| Register 01 | 01 00 | In register 01 (ISO control register),<br>write 00 (set protocol to ISO 15693 low bit rate, 6.62 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, AGC off = 00 |
| 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 | AM = FF, 00 = PM |
| EOF | 00 00 | End of frame |

## 4.3 Commands

### 4.3.1 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.
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.

**Request Packet:**
**01 0B 00 03 04 14 04 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 | 04 | Inventory flag = 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:
[<Tag response if any>, RSSI register value]
RSSI = Received Signal Strength Indicator
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 main channel, which is AM, is used as the primary one, and PM is the auxiliary channel.

Example:

[,40] Comment: (slot # 0, **[,40]** < no tag response >, RSSI register status)
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
[2CF7FE11000007E0,63] Comment: (slot # 12, **[2CF7FE11000007E0,63]** < tag UID in reverse-byte order>, RSSI register status, 63 means value 6 for main channel, 3 for auxiliary channel)
[,40]
[,40]
[,40]

### 4.3.2 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 (Option flag = 1). This byte shows the write-protection of the block specified [e.g., unlocked, (user/factory) locked, etc.].

**Request Packet:**
**01 0B 00 03 04 18 00 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 | 00 | No flags |
| 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)**

[0011111111] Comment: ( [**00** no tag error, **11 11 11 11** tag block data, 32 bits])

### 4.3.3 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 the reader does not necessarily indicate a failure to perform the write operation. Additionally, multiple transponders may process a nonaddressed request.

**Request Packet:**
**01 0F 00 03 04 18 00 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 | 00 | No flags |
| 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)**

[00] Comment: ( [**00**] no tag error)

### 4.3.4 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 reader 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.

**Request Packet:**
**01 0B 00 03 04 18 00 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 | 00 | No flags |
| 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)**

□ Comment: ( □ no tag response)

### 4.3.5 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 (Option flag = 1). This byte shows the write-protection of the block specified [e.g., unlocked, (user/factory) locked, etc.].

**Request Packet:**
**01 0C 00 03 04 18 00 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 | 40 | Option 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)**

[0011000011220000223300033] Comment: ( [**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])

### 4.3.6 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.

Executes **Write Single Block** multiple times.
**01 0F 00 03 04 18 00 21 02** 11 11 11 11 **00 00 Block 02 write; (block #3)**
**01 0F 00 03 04 18 00 21 03** 00 00 00 00 **00 00 Block 03 write; (block #4)**
**01 0F 00 03 04 18 00 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 | 00 | No flags |
| 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)**

[00] Comment: ( [**00**] no tag error)

### 4.3.7 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.

**Request Packet:**
**01 12 00 03 04 18 20 02 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 | Address flag is set |
| Stay Quiet Cmd | 02 | |
| UID | 8C AC D6 06 00 00 07 E0 | UID (reverse byte ordered). Normal UID byte order is E0 07 00 00 06 D6 AC 8C. |
| EOF | 00 00 | End of frame |

**Stay Quiet (Tag Response)**

[] Comment: ( **[]** no tag response)

### 4.3.8 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.

**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 | Address flag is set |
| Select Cmd | 25 | |
| UID | 8C AC D6 06 00 00 07 E0 | UID (reverse byte ordered). Normal UID byte order is E0 07 00 00 06 D6 AC 8C. |
| EOF | 00 00 | End of frame |

**Select (Tag Response)**

[] Comment: ( **[]** no tag response)

### 4.3.9 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.

**Request Packet:**
**01 0A 00 03 04 18 00 26 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 |
| Reset to Ready Cmd | 26 | |
| EOF | 00 00 | End of frame |

**Reset to Ready (Tag Response)**

[00] Comment: ( [**00**] no tag error)

### 4.3.10 Write AFI (Application Family Identifier)

The *Write AFI* command records a new value to the AFI register (see Page 36 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.

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. This is depending on the manufacturer of the tag, such as Texas Instruments.

**Request Packet:**
**01 0B 00 03 04 18 40 27 05 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 |
| Write AFI Cmd | 27 | |
| AFI | 05 | Application family identifier, 05 = medical application |
| EOF | 00 00 | End of frame |

**Write AFI (Tag Response)**

[00] Comment: ( [**00**] no tag error)

### 4.3.11 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.

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. This is depending on the manufacturer of the tag, such as Texas Instruments.

**Request Packet:**
**01 0A 00 03 04 18 40 28 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 | 40 | Option flag = 1 |
| Lock AFI Cmd | 28 | |
| EOF | 00 00 | End of frame |

**Lock AFI (Tag Response)**

[] Comment: ( **[]** no tag error)

### 4.3.12 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.

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. This is depending on the manufacturer of the tag, such as Texas Instruments.

**Request Packet:**
**01 0B 00 03 04 18 40 29 18 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 |
| Write DSFID Cmd | 29 | |
| DSFID value | 18 | Data Storage Format ID |
| EOF | 00 00 | End of frame |

**Write DSFID (Tag Response)**

[00] Comment: ( [**00**] no tag error)

### 4.3.13 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.

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. This is depending on the manufacturer of the tag, such as Texas Instruments.

**Request Packet:**
**01 0A 00 03 04 18 40 2A 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 | 40 | Option flag = 1 |
| Lock DSFID Cmd | 2A | |
| EOF | 00 00 | End of frame |

**Lock DSFID (Tag Response)**

[] Comment: ( **[]** no tag response)

### 4.3.14 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).

**Request Packet:**
**01 0A 00 03 04 18 00 2B 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 |
| 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:

[000F6EADD606000007E000003F0388] Comment: [**00 0F 6EADD606000007E0 00 00 3F 03 88**] tag response shown as follows:

| Field | Contents | Comments |
|---|---|---|
| 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 E0 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 definded by manufacturer |

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

The *Get Multiple-Block Security Status (Get Mult. 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.).

**Request Packet:**
**01 0C 00 03 04 18 00 2C 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 | 18 | Request Mode |
| Flags | 00 | No flags |
| Get Multiple Block Security Status Cmd | 2C | Read Multiple Blocks Cmd |
| 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)**

[00000000] Comment: ( [**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])

## 4.4 Appendix

### 4.4.1 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

### 4.4.2 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.

### 4.4.3 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 Page 36) 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 | High data rate |
| | | 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. |

### 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 |

### 4.4.4 ISO15693 Commands

| **ISO15693 COMMANDS** | |
|---|---|
| | **Request Code** |
| **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 |
| **B&R Custom Commands** | |
| AGC toggle | 0xF0 |
| AM/PM toggle | 0xF1 |

### 4.4.5 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 |

### 4.4.6 Response Flags

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

### 4.4.7 Status Flag (Response Frame)

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

### 4.4.8 Control Flags (Request Frame)

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

### 4.4.9 Information Flags

| Bit | Flag Name | Value | Description |
|---|---|---|---|
| b1 | DSFID | 0 | DSFID is not supported. DSFID field is not present. |
| | | 1 | DSFID is supported. DSFID field is present. |
| b2 | AFI | 0 | AFI is not supported. AFI field is not present. |
| | | 1 | AFI is supported. AFI field is present. |
| b3 | VICC memory size | 0 | Information on VICC memory size is not supported. Memory size field is not present. |
| | | 1 | Information on VICC memory size is supported. Memory size field is present. |
| b4 | IC reference | 0 | Information on IC reference is not supported. IC reference field is not present. |
| | | 1 | Information on IC reference is supported. IC reference field is present. |
| b5 | RFU | 0 | Reserved for future use |
| b6 | RFU | 0 | |
| b7 | RFU | 0 | |
| b8 | RFU | 0 | |

### 4.4.10 Chip Status Control Register (00h)

| Controls the power mode, RF on / off, AGC, AM / PM | | | |
|---|---|---|---|
| Register default is 0x01. It is preset at EN = L or POR = H | | | |
| **Bit** | **Bit Name** | **Function** | **Comments** |
| B7 | stby | 1 = standby mode<br>0 = active mode | Standby mode keeps regulators and oscillator running en_rec = L, en_tx = L |
| B6 | direct | 1 = received subcarrier signal (decoders bypassed)<br>0 = received decoded signal from selected decoder | The modulation control is direct through MOD input. The receiver subcarrier signal is on I/0_6. |
| B5 | rf_on | 1 = RF output active<br>0 = RF output not active | When B5 = 1, it activates the RF field and receivers. |
| B4 | rf_pwr | 1 = half output power<br>0 = full output power | 1 = RF driver at 8 Ω<br>0 = RF driver at 4 Ω |
| B3 | pm_on | 1 = selects PM input signal<br>0 = selects AM input signal | Multiplexes the RX1_IN1 and RX2_IN2 inputs |
| B2 | agc_on | 1 = AGC on<br>0 = AGC off | AGC selection |
| B1 | rec_on | 1 = Receiver enable for external field measurement | Receiver and oscillator are enabled, intended for external field measurement. |
| B0 | vrs5_3 | 1 = 5V operation (Vin)<br>0 = 3V operation | Selects the $V_{DD\_RF}$ range; 5 V (4.3 V–5 V), or<br>3 V (2.7 V–3.4 V) |

## 5

## 6

### 6.1.1 ISO Control Register (01h)

| | | | |
|---|---|---|---|
| Controls the ISO selection | | | |
| Register default is 0x02, which is ISO15693 high bit rate, one subcarrier, 1 out of 4. It is preset at EN = L or POR = H. | | | |
| **Bit** | **Bit Name** | **Function** | **Comments** |
| B7 | rx_crc_n | Receiving without CRC | 1 = no RX CRC<br>0 = RX CRC |
| B6 | dir_mode | Direct mode type | 0 = output is subcarrier data.<br>1 = output is bit stream (I/O_6) and bit clock (I/O_5) from decoder selected by ISO bits |
| B5 | rfid | RFID mode | Should always be set to 0 |
| B4 | iso_4 | RFID mode | See Page 40 RFID MODE Selections |
| B3 | iso_3 | | |
| B2 | iso_2 | | |
| B1 | iso_1 | | |
| B0 | iso_0 | | |

### 6.1.2

### 6.1.3 RFID MODE Selections

| Iso_4 | Iso_3 | Iso_2 | Iso_1 | Iso_0 | Protocol | Remarks |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | ISO15693 low bit rate 6.62 kbps one subcarrier 1 out of 4 | Default for reader |
| 0 | 0 | 0 | 0 | 1 | ISO15693 low bit rate 6.62 kbps one subcarrier 1 out of 256 | |
| 0 | 0 | 0 | 1 | 0 | ISO15693 high bit rate 26.48 kbps one subcarrier 1 out of 4 | |
| 0 | 0 | 0 | 1 | 1 | ISO15693 high bit rate 26.48 kbps one subcarrier 1 out of 256 | |
| 0 | 0 | 1 | 0 | 0 | ISO15693 low bit rate 6.67 kbps double subcarrier 1 out of 4 | |
| 0 | 0 | 1 | 0 | 1 | ISO15693 low bit rate 6.67 kbps double subcarrier 1 out of 25 | |
| 0 | 0 | 1 | 1 | 0 | ISO15693 high bit rate 26.69 kbps double subcarrier 1 out of 4 | |
| 0 | 0 | 1 | 1 | 1 | ISO15693 high bit rate 26.69 kbps double subcarrier 1 out of 256 | |
| 0 | 1 | 0 | 0 | 0 | ISO14443A bit rate 106 kbps | RX bit rate when TX bit rate is different than RX (reg03) |
| 0 | 1 | 0 | 0 | 1 | ISO14443A high bit rate 212 kbps | |
| 0 | 1 | 0 | 1 | 0 | ISO14443A high bit rate 424 kbps | |
| 0 | 1 | 0 | 1 | 1 | ISO14443A high bit rate 848 kbps | |
| 0 | 1 | 1 | 0 | 0 | ISO14443B bit rate 106 kbps | RX bit rate when TX bit rate is different than RX (reg03) |
| 0 | 1 | 1 | 0 | 1 | ISO14443B high bit rate 212 kbps | |
| 0 | 1 | 1 | 1 | 0 | ISO14443B high bit rate 424 kbps | |
| 0 | 1 | 1 | 1 | 1 | ISO14443B high bit rate 848 kbps | |
| 1 | 0 | 0 | 1 | 1 | Tag-it | |

# 7 Operating the Transponder with AutomationStudio

## 7.1 Requirements

For operating the transponder with AutomationStudio at least version **AS3.0.80** and **AR3.0** is required.

In AutomationStudio and AutomationRuntime the use of the transponderreader/writer is supported by the libraries **AsUSB** and **DVFrame**.

## 7.2 Library AsUSB

With this library and the included function blocks **UsbNodeListGet** and **UsbNodeGet** you can search a transponderreader/writer on the target.
The search takes place with **VendorID** and **ProductID**.
With the **ifName** of the output structure from the function block **UsbNodeGet** and using the library **DVFrame** in series, you can communicate with the transponderreader/writer.

### Information:

**Further informations and some samples in ANSI-C you can find in the online help of AutomationStudio.**

## 7.3 Library DVFrame

With this library you can communicate with the transponderreader/writer.

With the function blocks of the library **DVFrame** it is allowed to send or to receive data as a frame.
The frame driver controls the hardware without changing the data in the frame.
So you can realize fast and easy application specific communications.

### Information:

**Further informations and some samples in ANSI-C you can find in the online help of AutomationStudio.**

FRM_xopen

To open a transponderreader/writer-device the usbDevice.ifName is transfered in the parameter “device”.
No optional parameters in the paramterstring “mode” are necessary.

```
/* initialize open structure */
FrameXOpenStruct.device = (UDINT) StringDevice;
FrameXOpenStruct.mode = (UDINT)0;
FrameXOpenStruct.config = (UDINT)0;
FrameXOpenStruct.enable = 1;
FRM_xopen(&FrameXOpenStruct); /* open an interface */
```

FRM_read

In the USB-driver an identifier of the frame-end is implemented. The identifier is **0x0a (LF)**
With the function block **FRM_read** always a complete frame inclusive the identifier **0x0a (LF)** or the maximum length of a frame (256 bit) will be received.
If frames are greater than 256 bit and therefore the length of the first received frame is equal to 256 you have to repeat the function block **FRM_read** as long as the received length is <256 bit.
The driver can handle a maximum of 256 frames with a length of 256bit.
If you transmit more commands to the transponderreader/writer and more then 256x256 bytes are received, then no more data can be handled. The data will be erased and the user gets no error message.

# 8 Contents of delivery

| Number of pieces | Description |
|---|---|
| 1 | Transponder-Reader 5E9010.29 |
| 1 | Cable-clip |

**Table 7: Contents of delivery**

# 9 Figure index

# 10 Table index

# 11 Index