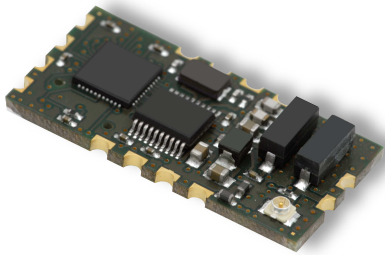


## Protocol Guide ISO15\_V2 Family

of metraTec HF RFID readers and modules based on ISO 15693



Date: July 2020

Version: 1.1

For firmware version: 3.10

Customer Edition

## Table of Contents

Introduction .....	6
1. Devices .....	6
2. Target audience .....	6
3. Further Documents .....	7
Typographic Conventions .....	8
1. Host - Device - Communication and Framing .....	10
1.1. Hardware .....	10
1.2. Framing .....	10
1.3. Instruction composition .....	10
1.4. Helpful Tools .....	11
2. Reader Control Instructions .....	12
2.1. Reset ( <b>RST</b> ) .....	13
2.2. Activate Bootloader ( <b>BTL</b> ) .....	14
2.3. Revision ( <b>REV</b> ) .....	14
2.4. Read Firmware ( <b>RFW</b> ) .....	15
2.5. Read Hardware ( <b>RHW</b> ) .....	15
2.6. Read Bootloader Revision ( <b>RBL</b> ) .....	16
2.7. Read Hardware Revision ( <b>RHR</b> ) .....	16
2.8. Read Serial Number ( <b>RSN</b> ) .....	17
2.9. End Of Frame Mode ( <b>EOF</b> ) .....	17
2.10. No End of Frame Mode ( <b>NEF</b> ) .....	18
2.11. Cyclic Redundancy Check Control ( <b>CRC</b> ) .....	19
2.12. Cyclic Redundancy Check On ( <b>CON</b> ) .....	20
2.13. Cyclic Redundancy Check Off ( <b>COF</b> ) .....	20
2.14. Heartbeat ( <b>HBT</b> ) .....	21
2.14.1. Get heartbeat parameter state ( <b>SHW</b> ) .....	22
2.14.2. HBT OFF ( <b>OFF</b> ) .....	22
2.14.3. HBT ON ( <b>INT</b> ) .....	22
2.15. Start Up Commands ( <b>SUC</b> ) .....	23

2.15.1. Reset Command Sequence .....	23
2.15.2. Set Command Sequence .....	24
2.15.3. Enable (ON) .....	24
2.15.4. Disable (OFF) .....	24
2.15.5. Reset (RST) .....	24
2.15.6. Execute (EXE) .....	25
2.15.7. Print out (SHW) .....	25
2.15.8. Fully Deleted (DEL) .....	25
2.16. Read Start Up Commands (RSC) .....	26
2.17. Edge Commands (EGC) .....	26
2.18. Continuous Repeat Prefix (CNR) .....	29
2.19. Break (BRK) .....	30
2.20. Standby (STB) .....	30
2.21. Wake Up (WAK) .....	31
2.22. Read Input Pin (RIP) .....	31
2.23. Write Output Pin (WOP) .....	32
2.24. Set Antenna Port (SAP) .....	33
2.24.1. Activate specific Antenna using MAN .....	34
2.24.2. Activate specific Antenna not using MAN .....	34
2.24.3. Automatic switching mode .....	35
2.24.4. Show current state .....	35
2.24.5. Automatic antenna reporting .....	35
2.25. Echo (ECH) .....	36
2.26. Verbosity Level (VBL) .....	37
2.27. Mode (MOD) .....	37
2.28. Settings (SET) .....	38
2.28.1. High on Tag values (HOT) .....	38
2.28.2. High on Tag flags (HOT) .....	39
2.28.3. Configure Input High Commands (IHC) .....	40
2.28.4. Set Input High Commands (IHC) .....	40

2.28.5. Power Level ( <b>PWR</b> ) .....	40
2.29. Set RF Interface ( <b>SRI</b> ) .....	41
2.29.1. RX1 input .....	41
2.29.2. RX2 input .....	42
2.29.3. Single Subcarrier .....	42
2.29.4. Double Subcarrier .....	43
2.29.5. Disable RF .....	43
2.29.6. Config T3 mode .....	43
2.29.7. Disable for some Time .....	44
2.29.8. Config power level .....	44
2.29.9. Config power saving .....	45
2.29.10. Get current RF state .....	45
2.30. Set Timings ( <b>STT</b> ) .....	46
2.30.1. Set Timings T1MAX, T1MIN, T2, T3 and TWMAX .....	46
2.30.2. Set Timing TWMIN .....	48
2.30.3. Set Timing TWWND .....	48
3. Tag Manipulation Instructions .....	50
3.1. Inventory ( <b>INV</b> ) .....	50
3.1.1. Simple Inventory .....	50
3.1.2. Application family identifier ( <b>AFI</b> ) .....	51
3.1.3. Single Slot Inventory ( <b>SSL</b> ) .....	51
3.1.4. Only New Tag ( <b>ONT</b> ) .....	51
3.1.5. Masking ( <b>MSK</b> ) .....	52
3.2. Reading Request ( <b>REQ</b> ) .....	53
3.3. Writing Request ( <b>WRQ</b> ) .....	56
3.4. Direct Reading Request ( <b>DRQ</b> ) .....	57
3.5. Direct Writing Request ( <b>DWQ</b> ) .....	58
4. Precommands .....	59
4.1. Command Answer Prefixes ('P') .....	59
4.2. Answer Counter ('C') .....	59

4.3. Using 'P' and 'C' together .....	60
4.4. Single Prefix ('SP') .....	61
5. Error Codes .....	62
A. Quick Start Guide and Examples .....	65
A.1. Typical Reader Initialization Sequence .....	65
A.2. Reading the Tag ID of a tag .....	65
A.3. Reading Tag IDs continuously .....	66
A.4. Example for writing and reading to and from ISO 15693 tags .....	67
A.5. Configuring reader to automatically start reading tag IDs when powered .....	68
B. CRC Calculation .....	69

# Introduction

## 1. Devices

This document describes the metraTec firmware protocol for the following metraTec RFID readers:

QR15\_V2

Dwarf15, HW Revision 03xx

DeskID\_ISO, HW Revision 03xx

QuasarMX, HW Revision 03xx

UM15, HW Revision 03xx, Hardware name is DeskID\_ISO

All of these work according to ISO 15693 with ISO15693 tags.



### Note

UM15 are identical to DeskID\_ISO both in hardware and firmware. They only lack the DeskID\_ISOs package. The documentation will therefor use only DeskID\_ISO but never UM15 to describe this products.

A description of the other protocols can be found on the website, at <https://www.metratec.com/en/support/downloads>.

This guide does not cover the protocol for other ISO15 readers but they usually share a subset of equal commands while differing in others. Other devices will have different hardware or special usecases causing them to differ. This includes older hardware revisions of Dwarf15, DeskID\_ISO and QuasarMX plus the QR15.

The same is true for older firmware versions. You can expect commands to not be removed if possible but to get added parameters or getting outdated by new, stronger or more precise functions.

A special case is the QuasarLR. It is parallel developed sharing firmware revision counting. The user can expect a lot of functions to be shared and added at the same time. This is mostly true for comfort functions (lets say heartbeating or power saving) as the hardware still differs.

## 2. Target audience

The target audience for this document are programmers, who need to communicate with the reader and want to write their own software for this task using the programming language of their choice. An alternative to this low level protocol is to use our free .NET DLL on MS Windows systems. As this Programming Guide is the reference of all commands the reader supports it is by necessity rather long and complex in parts. If you just want to get started and would like to see how easy the readers are to use in typical applications please start by checking out the Quick Start Guide and Examples in the Appendix.



## Note

Code examples are written in C99.

Instructions (as well as this document) are divided into two main groups:

- Reader Instructions, divided into
  - Reader Control Instructions
  - Reader Configuration Instructions
- Tag Manipulation Instructions

All instructions have error codes that are described in Chapter 5, *Error Codes*.

### 3. Further Documents

For an even deeper understanding of the operating principle it might be useful to read the datasheets and norms regarding your tag IC, esp. ISO 15693(-3) as well as the respective tag IC datasheet.



## Typographic Conventions

Special typographic conventions and highlightings are used in metraTec protocol guides and other documents to streamline content that would otherwise be hard to express (e.g. syntax descriptions) and in order to provide a consistent look across metraTec documentation.

The following table summarizes typographic conventions and their descriptions:

Convention	Description
<b>COMMAND</b>	A command name, i.e. the <i>literal</i> name of a command in a metraTec protocol. For instance, <b>RST</b> would correspond to the literal characters of a command that could be sent to a metraTec device.
Literal	Highlights a <i>literal value</i> directly representing the literal characters that have to be used (e.g. in a protocol). For instance UCO would correspond to the literal characters as they could be returned by a metraTec device.
<i>Token</i>	This convention highlights a replaceable (abstract) <i>Token</i> that in contrast to a literal token is a placeholder for some other value that must be substituted by the user. The abstract <i>Token</i> is usually documented in more detail.
<LITERAL>	Represents a literal character that cannot be printed as such or needs to be highlighted specifically and is therefore formatted as an abstract identifier. Examples include "<CR>" — representing the carriage return character (ASCII 13) — and "<SPACE>" — representing one or more space characters (ASCII 32). This special formatting is used both in syntax descriptions, command and response examples.
{ <i>Construct</i> }	This convention highlights that a <i>Construct</i> is required. It is most commonly used in syntax descriptions to highlight that a parameter <i>must</i> be specified in the position that this construct is used.
[ <i>Construct</i> ]	Highlights that a <i>Construct</i> is optional. It is most commonly used in syntax descriptions to highlight that a parameter <i>may</i> be specified in the position that this construct is used.
<i>Construct</i> ...	Highlights that a <i>Construct</i> may be repeated many times.
... <i>Name</i> ...	The horizontal ellipsis "..." may be used in syntax descriptions to represent arbitrary characters. The arbitrary character field may be given a <i>Name</i> in order to document it in more detail.
<i>Alternative</i> <sub>1</sub>   <i>Alternative</i> <sub>2</sub> ...   <i>Alternative</i> <sub>n</sub>	Highlights that in the position of this construct one of <i>n</i> alternatives may be used.
<i>Literal Line 1</i> <i>Literal Line 2</i> <i>Literal Line 3</i>	A literal block of text. It is often used to document example protocol exchanges or code examples. In the former case, literal placeholders like "<CR>" may be included in the code block to express that lines are separated by carriage return. In the latter case, programming language source



Convention	Description
	code may be syntax highlighted. These literal blocks of code may also contain callout graphics or line annotations to document each line of text.
» <i>Command</i>	In examples of a command-response exchange, the literal examples of the <i>Command</i> and <i>Response</i> may be highlighted differently. Otherwise these literal blocks of text are formatted the same as described above.
« <i>Response</i>	
 <b>Note</b> Paragraph	A paragraph set off from the text to highlight noteworthy information.
 <b>Warning</b> Paragraph	A paragraph set off from the text to highlight information necessary to prevent harm to electronic devices or persons.

# Chapter 1. Host - Device - Communication and Framing

## 1.1. Hardware

Devices all use a UART internally. Some (like QR types) use them directly. Others convert to USB (like DeskID types) or Ethernet (like Quasar / Pulsar types). The UART uses 115200 baud, no parity bit, 1 stop bit for both directions. Flow control is not supported.

## 1.2. Framing

The host - to - reader communication based on ASCII strings. Each string is terminated with a carriage-return (0x0D) character, *not* the null byte like in C strings, and will be transmitted with the most significant byte first. Every command gets an answer. 0x0D is called carriage return, **<CR>** or {CR} in this document.

The communication from the reader to the host system (i.e. the responses or events) can be multiple lines long. The answer can be terminated by a line-feed character (0x0A, referred to as **<LF>** or {LF}). This is controlled by **EOF** command. This is useful because reader - to - host communication can be multiple lines long and there are some events. Events can happen without a causing command. Examples are heartbeat events (controlled by **HBT** command).



### Note

The line-feed end of events or responses is deactivated by default.

## 1.3. Instruction composition

An instruction is composed of a command, usually 3 characters long, and may be followed by a command depending number of parameters. Their meaning can depend on the command and also on previous parameters. The parameter use is described in the command description. Trailing spaces are allowed (though not recommended), any other whitespace, be it added or replacing an expected **<SPACE>** is forbidden.

```
{ Command } [ <SPACE> Parameter ... ] <CR>
```

```
RFW<CR>
```

```
char RFW[ 4 ] = { 'R', 'F', 'W', 13 };
```

Example 1. **RFW** command without parameter

The first value which will be sent in the above examples is 0x52 ('R'), followed by 0x46, 0x57, 0x0D.

```
INV SSL<CR>
```

```
char Inv[8] = "INV SSL\r";
```

Example 2. **INV** command with parameter *SSL*

## 1.4. Helpful Tools

For debugging purposes it is very helpful to use a program to “sniff” the communication between the host and the reader. Depending on the type of communication and hardware you use, this can be:

- If you communicate via a (real or virtual) COM-Port: a Com-Port Monitor (several free version available on the net)
- If you use Ethernet or other TCP/IP-based communication, like WiFi: a packet sniffing tool, e.g. wireshark/ethereal, which is available for almost every platform
- If you use a direct UART connection or something at a similar low level: a hardware logic analyzer
- To send ASCII data via a serial connection or even Ethernet, you can use the free me-traTerm terminal software, available on our website.

## Chapter 2. Reader Control Instructions

This list gives an overview of all the existing instructions that directly influence the reader itself. All commands that are connected to the tag can be found in the Tag manipulation instructions.

Command	Name	Description
<b>RST</b>	Reset	Resets the reader
<b>BTL</b>	Activate Bootloader	Activates bootloader
<b>REV</b>	Revision	Returns information on reader (partially deprecated)
<b>RFW</b>	Read Firmware	Returns firmware name and version
<b>RHW</b>	Read Hardware	Returns hardware name and version
<b>RBL</b>	Read Bootloader Revision	Returns bootloader name and version
<b>RHR</b>	Read Hardware Revision	This command is deprecated.
<b>RSN</b>	Read Serial Number	Reads the Serial Number of the reader
<b>EOF</b>	End Of Frame Mode	Controls the use of End of Frame delimiter <LF>
<b>NEF</b>	No End of Frame Mode	This command is deprecated.
<b>CRC</b>	Cyclic Redundancy Check Control	Controls the CRC usage on the Host-Reader communication interface.
<b>CON</b>	Cyclic Redundancy Check On	This command is deprecated.
<b>COF</b>	Cyclic Redundancy Check Off	This command is deprecated.
<b>HBT</b>	Heartbeat	Sends a "HBT" every given number of seconds
<b>SUC</b>	Start Up Commands	Commands to execute on startup
<b>RSC</b>	Read Start Up Commands	This command is deprecated.
<b>EGC</b>	Edge Commands	Controls the edge triggered command feature
<b>CNR</b>	Continuous Repeat Prefix	A meta command used to repeat a tag command indefinitely.
<b>BRK</b>	Break	Command to end <b>CNR</b> mode
<b>STB</b>	Standby	Sends the reader into standby / sleep mode to save power
<b>WAK</b>	Wake Up	Ends standby / sleep mode
<b>RIP</b>	Read Input Pin	Reads the state of an input pin
<b>WOP</b>	Write Output Pin	Writes the state of an output pin
<b>SAP</b>	Set Antenna Port	Sets the antenna port that is active on a metaTec multiplexer
<b>ECH</b>	Echo	Echos up to 16 characters
<b>VBL</b>	Verbosity Level	Sets the amount of details the reader communicates to the user
<b>MOD</b>	Mode	MOD is not supported
<b>SET</b>	Settings	Command that allows configuring some reader settings
<b>SRI</b>	Set RF Interface	Configures RF interface

Command	Name	Description
<b>STT</b>	Set Timings	Sets important timings for ISO 15693 communication.

Table 1. Overview of Reader Control Instructions

## 2.1. Reset (RST)

The **RST** command resets the reader. The reset command has no parameters. After sending the command and receiving the answer OK! the reader will behave mostly like after (re-)powering. The bootloader starts, the basic configuration (UART etc.) is done, the HF power is still turned off and the reader has to be initialized again. This includes Startup commands from **SUC** command.



### Note

There might be some minor hardware differences though. For an absolute identical startup you need the power reset.



### Note

Reset is executed even with a communication CRC disabled (**CRC OFF**) but CRC added, in standby state (**STB**) or running **CRC** mode. This allows to safely reset anytime.

The startup process (from the time the OK! is received until new commands to the reader are accepted and processed) takes up to 200ms.

### Instruction

**RST** <CR>

### Examples

```
RST<CR>
```

Example 3. Reset reader to default values / behaviour

### Return Values in Case of Success

OK! <CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.2. Activate Bootloader (BTL)

This command resets the device and activates the bootloader.

### Instruction

**BTL** <CR>

### Examples

```
BTL<CR>
```

*Example 4. Activates bootloader*

### Return Values in Case of Success

OK! <CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.3. Revision (REV)

This command is mostly partially deprecated. For reading the firmware name (which is identical to the product name) and version use **RFW**. HW\_arch is not readable by any other command but there is usually no need for this value.

### Instruction

**REV** <CR>

### Examples

```
REV<CR>
```

*Example 5. Read reader information (deprecated)*

### Return Values in Case of Success

PRODUCT\_NAME[16 bytes]HW\_arch[4bytes]FW\_revision[4bytes]<CR>

### Return Values in Case of Failure

"NOS <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.4. Read Firmware (RFW)

This command returns the name of the firmware - identical to the product name - and the version of the firmware.

### Instruction

**RFW** <CR>

### Examples

```
RFW<CR>
```

*Example 6. Read firmware name and revision from reader*

### Return Values in Case of Success

```
FIRMWARE_NAME FW_revision[4bytes]<CR>
```

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.5. Read Hardware (RHW)

This command returns the name of the hardware and the version of the hardware - both identical to the name and version printed on the circuit board.

### Instruction

**RHW** <CR>

### Examples

```
RHW<CR>
```

*Example 7. Read hardware name and revision from reader*

### Return Values in Case of Success

```
PRODUCT_NAME HW_revision[4bytes]<CR>
```

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.6. Read Bootloader Revision (RBL)

This command returns the name and revision of the bootloader running on the device.

### Instruction

**RBL** <CR>

### Examples

```
RBL<CR>
```

*Example 8. Read bootloader name and version*

### Return Values in Case of Success

*BOOTLOADER\_NAME* *MAYOR\_REV*[4bytes]*MINOR\_REV*[4bytes]<CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.7. Read Hardware Revision (RHR)

This command is deprecated. Use **RHW** to get hardware revision (plus name) instead. It returns 4 characters, the first 2 are major revision, the second are subrevision.

### Instruction

**RHR** <CR>

### Examples

```
RHR<CR>
```

*Example 9. Read hardware revision of reader (deprecated)*

### Return Values in Case of Success

*MMSS*<CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"



## 2.8. Read Serial Number (RSN)

The RSN command returns the serial number of the reader.

### Instruction

**RSN** <CR>

### Examples

```
RSN<CR>
```

*Example 10. Read serial number of reader*

### Return Values in Case of Success

JJJJMMDDHHMMSSxx<CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.9. End Of Frame Mode (EOF)

With enabled End of frame delimiter any instruction answer or event information will be ended by a line feed (<LF>, 0x0A) character. This is additional to the <CR> ending every line. This allows the user to build a simpler parser since it is clear when not to expect any further message from the reader and what lines are part of a single answer.

In case a command was executed using the **CNR** prefix for repetitive / continuous execution every complete answer of a single iteration will also be appended with the additional line feed.



### Note

Please keep in mind that asynchronous errors that reset the reader (like brownout or watchdog) will lead to the error code being reported after the reader has been reset (and the EOF delimiter deactivated as is the default setting). Thus the error code will not be terminated by the line feed! This is even true with **SUC** usage as the commands are executed after the check of reset causes.

Using no parameter is identical to **EOF ON** to be compatible with older device versions and types. This use is deprecated and therefore not recommended to use.

### Instruction

**EOF** [<SPACE> { ON | OFF | SHW }] <CR>

## Parameters

Name	Type
End of frame state	Optional Enumeration (ON, OFF or SHW)

## Examples

```
EOF ON<CR>
```

Example 11. Turn on End of Frame delimiter <LF>

```
EOF OFF<CR>
```

Example 12. Turn off End of Frame delimiter <LF>

```
EOF SHW<CR>
```

Example 13. Show End of Frame delimiter state answering 'ON' or 'OFF'

## Return Values in Case of Success

"OK! <CR>", "ON <CR>" or "OFF <CR>"

## Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.10. No End of Frame Mode (NEF)

This command is deprecated. **NEF** command will turn off the End of Frame delimiter. It is identical to **EOF OFF**.

### Instruction

```
NEF <CR>
```

### Examples

```
NEF<CR>
```

Example 14. Turn off End of Frame delimiter

## Return Values in Case of Success

OK! <CR>

## Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.11. Cyclic Redundancy Check Control (CRC)

This command controls the Cyclic Redundancy Check (CRC) of the host-to-reader and vice versa communication. This is used to detect transmission errors between the reader and the host. In general enabling this feature is not necessary except in scenarios where you have lots of noise on the communication bus (e.g. when using USB communication in the vicinity of electric motors) or if you encounter any other problems with communication errors.

If this feature is activated (default is OFF), the reader firmware expects a CRC16 (4 hex numbers) between the command to the reader and the respective `<CR>`. Between the command and the CRC there is a space character which is included in the CRC calculation. All lines from the reader to host will also be extended accordingly.

The CRC uses the 0x8408 polynomial, starting value is 0xFFFF (just like ISO15693). Appendix B, *CRC Calculation* shows a function in C/C++ to calculate the correct CRC16.



### Note

This has nothing to do with the reader to tag communication CRC mandated by the ISO 15693 even though it uses the same CRC algorithm.

With the `SHW` the user gets the state of the CRC check printed back as `ON` or `OFF`. With `ON` and `OFF` CRC is enabled or disabled.



### Note

`CRC` will work with a CRC added even with CRC check disabled so the user can get or set the state even if the state is unknown. The same is true for `RST` and `BTL` to make a save reset or bootloader start possible.

## Instruction

```
CRC <SPACE> { ON | OFF | SHW } <CR>
```

## Parameters

Name	Type
CRC state	Enumeration (ON, OFF or SHW)

## Examples

```
CRC ON<CR>
```

*Example 15. Turn on CRC checking*

```
CRC OFF<CR>
```

*Example 16. Turn off CRC checking*

```
CRC SHW B6A8<CR>
```

*Example 17. Show CRC checking state answerering 'ON' or 'OFF', asking with CRC (always working even if CRC is off)*

## Return Values in Case of Success

"OK! <CR>", "OK! 9356 <CR>", "ON <CR>" or "OFF <CR>"

## Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.12. Cyclic Redundancy Check On (CON)

This command is deprecated. Disables the CRC just like **CRC ON**. If successful the command returns OK! with the according CRC of "OK! ".



### Note

CRC will work with a CRC added even with CRC check disabled so the user can set the state even if the state is unknown.

### Instruction

**CON** <CR>

### Examples

```
CON<CR>
```

*Example 18. Turn on reader to computer CRC checking*

## Return Values in Case of Success

OK! 9356 <CR>

## Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.13. Cyclic Redundancy Check Off (COF)

This command is deprecated. Disables the CRC just like **CRC OFF**. If successful the command returns OK! with the according CRC of "OK! ".



## Note

CRC will work with a CRC added even with CRC check disabled so the user can set the state even if the state is unknown.

### Instruction

**COF** [ <SPACE> 4F5E ] <CR>

### Examples

```
COF 4F5E<CR>
```

Example 19. Turn off reader to computer CRC checking (please note the mandatory CRC)

### Return Values in Case of Success

OK! <CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.14. Heartbeat (HBT)

The heartbeat command enables or disables the heartbeat. If enabled the device will send a HBT every given number of seconds. A HBT without any parameter or with SHW will give you the state meaning either OFF or the number of seconds.

```
>> HBT SHW<CR>
```

```
<< OFF<CR>
```

```
>> HBT<CR>
```

```
<< OFF<CR>
```

```
>> HBT 1<CR>
```

```
<< OK!<CR>
```

one second delay

```
<< HBT<CR>
```

one second delay

```
<< HBT<CR>
```

```

>> HBT 20<CR>
<< OK!<CR>

20 seconds delay

<< HBT<CR>

20 seconds delay

<< HBT<CR>

>> HBT SHW<CR>
<< 20<CR>

>> HBT OFF<CR>
<< OK!<CR>

```

Example 20. **HBT** command and answer

### 2.14.1. Get heartbeat parameter state (SHW)

#### Instruction

**HBT** [ <SPACE> SHW ] <CR>

#### Parameters

Name	Type	Description
SHW	Optional Key-word	The parameter is the SHW keyword. It's use is optional.

### 2.14.2. HBT OFF (OFF)

OFF will switch the heartbeat off.

#### Instruction

**HBT** <SPACE> OFF <CR>

#### Parameters

Name	Type	Description
OFF	Keyword	The parameter is the OFF keyword

### 2.14.3. HBT ON (INT)

Any number (1-300) will switch the heartbeat on with for the specified time in seconds as time distance.

## Instruction

**HBT** <SPACE> {Heartbeat time} <CR>

## Parameters

Name	Type
Heartbeat time	Decimal Integer ( $1 \leq x \leq 300$ )

## Return Values in Case of Success

OK! <CR>

## Return Values in Case of Failure

"NOR <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.15. Start Up Commands (SUC)

Any time the reader firmware (re-)starts - either by using the **RST** command or by enabling the power supply - the reader will set all parameters to their default settings. In case you want another setup after a reader reset, the **SUC** command allows setting up a set of commands to be executed at start up of the device - e.g. to (re-)apply any settings you want automatically, start CNR mode or activate edge controlled commands via **EGC** command. **SUC** commands are persistently stored in FLASH memory. Upon start up of the device the commands are loaded and executed nearly as if they had been given to the reader at that time. Responses to the commands are suppressed and commands have no CRC checking even if the commands contain a **CRC ON**. Due to this, please make sure to check the spelling of the commands you are setting as any error messages are also suppressed. For testing the user may use **SUC EXE**.

**ON** and **OFF** enable and disable the execution at startup without changing the command string. **EXE** executes the currently set of commands just like directly executing them. This includes normal answers so it can be used for testing. **SHW** prints out the current settings. **RST** resets the data to default settings (**OFF** and default string). **DEL** deletes the used data area. This results in truly freed memory so it is save for downgrades (and also for upgrades but this usually have no problems).

Multiple commands are separated by ";" (semicolon). **RST** and **BTL** are not accepted. If the continuous prefix **CNR** is used the command will be executed continuously and will therefor show results. As usual, continuously executing commands can be terminated by the **BRK** command.

### 2.15.1. Reset Command Sequence

**SUC** without further parameters is deprecated. Use the more clear **SUC RST** for identical results.

## Instruction

**SUC** <CR>

## Examples

```
SUC<CR>
```

Example 21. SUC instruction for resetting the command sequence previously defined

### 2.15.2. Set Command Sequence

**SUC** followed by a semicolon separated sequence of regular commands sets these as the command sequence to be carried out at startup.

#### Instruction

```
SUC <SPACE> {...} <CR>
```

## Examples

```
SUC SRI SS 100;CNR INV<CR>
```

Example 22. SUC instruction for automatic start of continuous ID reading

### 2.15.3. Enable (ON)

**SUC** followed by the keyword **ON** will reenables startup command execution previously disabled. If there are no data the command will give the WMO error.

#### Instruction

```
SUC <SPACE> ON <CR>
```

## Examples

```
SUC ON<CR>
```

Example 23. SUC instruction for reenabling startup commands

### 2.15.4. Disable (OFF)

**SUC** followed by the keyword **OFF** will disable startup command execution but leaves the commands in memory so they can be reactivated by **SUC ON**. Keep in mind this will use up a flash write cycle (both ON and OFF) and should therefore not be used frequently.

#### Instruction

```
SUC <SPACE> OFF <CR>
```

## Examples

```
SUC OFF<CR>
```

Example 24. SUC instruction for turning off startup commands

### 2.15.5. Reset (RST)

**SUC** followed by the keyword **RST** will disable startup command execution and delete the command.



## Instruction

**SUC** <SPACE> RST <CR>

## Examples

```
SUC RST<CR>
```

Example 25. SUC instruction for resetting startup commands

### 2.15.6. Execute (EXE)

**SUC** followed by the keyword **EXE** will execute the currently set commands as if the device is restarted. If execution is disabled the command will give the **WMO** error.

## Instruction

**SUC** <SPACE> RST <CR>

## Examples

```
SUC OFF<CR>
```

Example 26. SUC instruction for turning off startup commands

### 2.15.7. Print out (SHW)

This command will return the sequence of startup commands set via **SUC**. The answer to the command is a first line stating whether SUC mode is turned on or off and then one command of the command sequence is reported per line instead of using the formatting with ';' that was used when setting the command sequence. As a last line of the answer the command returns **OK!**. This replaces the older **RSC** command.

## Instruction

**SUC** <SPACE> SHW <CR>

## Examples

```
SUC SHW<CR>
```

Example 27. SUC instruction for printing startup commands

### 2.15.8. Fully Deleted (DEL)

**SUC** followed by the keyword **DEL** will delete the SUC memory. This can be used to be save the SUC will not cause update related problems, especially with downgrades. The update needs to be started without a reset in between. The Firmware Update Tool 2 does not support this natively. Therefore the user needs to use the **BTL** command to start the bootloader before starting the update. **SUC SHW** will give invalid data. If the device is reset the data will be set to default value (just like by **SUC RST**).

## Instruction

**SUC** <SPACE> DEL <CR>

## Examples

```
SUC OFF<CR>
```

Example 28. SUC instruction for turning off startup commands

### Return Values in Case of Success

OK! <CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.16. Read Start Up Commands (RSC)

This command is deprecated and replaceable by **SUC SHW**.

### Instruction

RSC <CR>

### Examples

```
RSC<CR>
```

Example 29. Deprecated: Read startup command sequence previously set via SUC

### Return Values in Case of Success

OK! <CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.17. Edge Commands (EGC)



### Note

The **EGC** command is not supported by DeskID\_ISO due to missing GPIOs.

EGC replaces and improves the deprecated **SET IHC**. It allows defining a set of commands that is to be executed following an edge of an input pin - e.g. when a light barrier triggers. Sup-

ported input pins are 0 and 1 (Input 0/1 on QuasarMX, GPIO 0/1 on DwarfG2 and DwarfG2\_XR) and for each pin a different set of commands can be saved.

Triggers are set by the flags

- NONE
- OFF (deprecated, equals NONE)
- FALL
- ON (deprecated, equals FALL)
- RISE
- BOTH

Triggers are not save persistent. Default state after reset is `NONE`. If edge commands should be used from the start use `SUC` to set the trigger(s) up.

Some flags are used to control the EGC function.

- SHW
- EXE
- RST
- DEL

`SHW` returns the trigger (`NONE`, `RISE`, `FALL`, `BOTH`) in the first line. It is followed by an unknown number of lines containing the commands. This can be anything from none to multiple lines. After all command lines the answer is finalized by `OK!`.

`EXE` executes the current set commands. `EXE` is mostly intended for debugging and development but can also be used as a kind of multi command shortcut if not needed otherwise. It returns the commands answers plus a finalizing `OK!`.

`RST` reset the EGC command and trigger to default ("`ECH EDGE 0`").

`DEL` deletes the EGC memory area. This is intended for downgrades as the memory layout will most likely not match so this would cause problems. This command will cause problems with triggers, execution and showing because the memory is blank. This will be fixed by a reset back to default (including downgrades or upgrades which then give the new / old defaults).

If no flag is used the data given is saved as commands to execute when triggered by input or `EXE` parameter. Individual commands are separated by `;`. The `;`'s are replaced by `<CR>` internally so this equals multiple commands. The commands are persistent so at restart only a `EGC 0 BOTH` (or whatever trigger is wanted) should to be called. This may be called by the `SUC` command to have a persistent edge control if wanted.



## Warning

EGC triggers can collide for bidirectional IOs on Dwarf15 and QR15\_V2. Outputs are on the same pins so output commands (e.g. `WOP`, `SET HOT`, `SAP`) can collide and may change behaviour. They should not be used. Trigger `NONE` will stop the usage so the host can use IOs for other functions again without fear of conflicts.

```
>> EGC 0 REV<CR>
```

```
<< OK!<CR>
```

```
>> EGC 0 SHW<CR>
```

```
<< NONE<CR>REV<CR>
```

```
>> EGC 0 RISE<CR>
```

```
<< OK!<CR>
```

```
>> EGC SHW<CR>
```

```
<< RISE<CR>REV<CR>
```

Drive Pin 0 from low to high

```
<< QUASAR_MX      01000314<CR>
```

Drive Pin 0 from high to low

Nothing happens

```
>> EGC 0 NONE<CR>
```

```
<< OK!<CR>
```

Drive Pin 0 from low to high

Nothing happens

Drive Pin 0 from high to low

Nothing happens

*Example 30. EGC command and answer*

## Instruction

**EGC** <SPACE> {Pin} <SPACE> {...Commands OR control flags OR trigger flags...}  
<CR>

## Parameters

Name	Type
Pin	Decimal Integer ( $0 \leq x \leq 1$ )

## Examples

```
EGC 0 SRI SS 100;CNR INV<CR>
```

*Example 31. Set Edge command for pin 0*

```
EGC 1 BRK<CR>
```

Example 32. Set Edge command for pin 1

```
EGC 0 BOTH<CR>
```

Example 33. Set Edge trigger to both edges for pin 0

```
EGC 0 SHW<CR>
```

Example 34. Show current settings for pin 0

```
EGC 0 EXE<CR>
```

Example 35. Execute current commands for pin 0

## Return Values in Case of Success

OK! <CR>

## Return Values in Case of Failure

"NOR <CR>", "EDX <CR>", "NOS <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.18. Continuous Repeat Prefix (CNR)

**CNR** starts **Tag Instructions** in a continuous mode (CNR mode). This will lead to the respective command being repeated indefinitely or until either the **BRK** command is sent, the **RST** command or **BTL** command is sent or, with **BAR** parameter used, until a tag is found / answers. This is a very powerful mechanism for unassisted operations where the reader is initialized at the beginning (e.g. via **SUC** command) and then repeats the command over and over. Examples for useful continuous operations are reading tag IDs, reading data from tags or even writing and locking data on tags continuously, e.g. in a printer.

### Instruction

```
CNR <SPACE> {...} <CR>
```

### Examples

```
CNR INV SSL<CR>
```

Example 36. Continuously search for single tags in the field

```
CNR REQ 022003 CRC<CR>
```

Example 37. Continuously try reading block 03 from any (single) tag in the field

## Return Values in Case of Success

Command Responses <CR>

## Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.19. Break (BRK)

If the reader is in **CNR** mode it can be stopped by sending **BRK**. If the device is not in CNR mode it answers **NCM** (Not in CNR mode). Otherwise the current CNR iteration is finalized before the command is executed so the answer can never interfere the current operations answer.

### Instruction

**BRK** <CR>

### Examples

```
BRK<CR>
```

Example 38. Stop CNR mode

## Return Values in Case of Success

BRA <CR>

## Return Values in Case of Failure

"NCM <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.20. Standby (STB)

The standby command sets the reader into power save mode. The RF power is turned off. This means all tags in the field will also be powered down. If successful it returns GN8 ("Good Night"). The reader will not accept any commands except **RST** command and **BTL** command until a **WAK** command is received. Standby has no parameters. Standby saves the antenna state. After waking up it will be active or inactive like before going into standby mode.

### Instruction

**STB** <CR>

## Examples

```
STB<CR>
```

Example 39. Send reader to standby mode

## Return Values in Case of Success

GN8 <CR>

## Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.21. Wake Up (WAK)

The wake up command ends the power save mode. The reader will restore its last state prior to entering standby mode. If successful it returns GMO ("Good Morning"). Wake up has no parameters.

### Instruction

WAK <CR>

## Examples

```
WAK<CR>
```

Example 40. Wake reader from standby mode

## Return Values in Case of Success

GMO <CR>

## Return Values in Case of Failure

"DNS <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.22. Read Input Pin (RIP)

This command is used to read the current state of an input pin. It takes one parameter, which is the zero-based number of the input pin to be read. The possible parameter range depends

on the number of inputs the hardware has. QuasarMX and QuasarLR accept 0 and 1 as pin numbers. Dwarf15 and QR15 accept 0 to 7 as input pin numbers. The DeskID ISO and UM15 do not have input pins.

If successful, it returns either HI or LOW depending on whether the input pin is high or low.



## Warning

In case of the QR15 and the Dwarf15 the input pins can also be used as output pins (General Purpose Inputs / Outputs - GPIOs). When calling **RIP** the direction the pin is being used in is set to being an input pin (just like default).

## Instruction

**RIP** <SPACE> {Pin\_No} <CR>

## Parameters

Name	Type	Description
Pin_No	Hexadecimal Integer ( $0_{16} \leq x \leq 7_{16}$ )	Number of pin to read

## Examples

```
RIP 0<CR>
```

*Example 41. Read status of input pin 0*

## Return Values in Case of Success

HI <CR>  
Pin is in high state

LOW <CR>  
Pin is in low state

## Return Values in Case of Failure

"NOR <CR>", "EHX <CR>", "NOS <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.23. Write Output Pin (WOP)

This command is used to set the state of an output pin either to high or to low. It takes two parameters. The first parameter is the zero-based hexadecimal number of the output pin to



be written to. The second parameter is either "HI" or "LOW" to set the according pin to high or low respectively. The possible parameter range depends on the number of output pins the hardware has. QuasarMX and QuasarLR accept 0 to 3 as output pin numbers. Dwarf15 and QR15 accept 0 to 7 as output pin numbers. The DeskID ISO and UM15 do not have output pins.



## Warning

In case of the QR15 and the Dwarf15 the output pins can also be used as input pins (General Purpose Inputs / Outputs - GPIOs). When calling **WOP** the direction the pin is being used in is changed to being an output pin. Please make sure that the hardware connected to the pin will not exceed the pin's maximum limits in output mode before calling **WOP** as this can destroy the reader.

## Instruction

**WOP** <SPACE> {Pin\_No} <SPACE> { HI | LOW } <CR>

## Parameters

Name	Type	Description
Pin_No	Decimal Integer ( $0 \leq x \leq 7$ )	Number of pin to write
Pin Setting	Enumeration (HI or LOW)	New state of pin

## Examples

```
WOP 0 HI<CR>
```

*Example 42. Set output pin 0 to high state*

## Return Values in Case of Success

OK! <CR>

## Return Values in Case of Failure

"NOR <CR>", "NOS <CR>", "EHX <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.24. Set Antenna Port (SAP)

This command is used to set multiple outputs of a reader at once so that a metraTec multiplexer connected to the reader will directly activate the correct antenna port. It replaces a sequence of **WOP** commands that would allow setting the individual outputs sequentially. As

the DeskID\_ISO / UM15 has no outputs they will respond with NOS. This is also the case for the embedded products which either have a dedicated antenna or don't support multiplexer operation due to voltage levels.



### Note

Please remember that this command will set four outputs of the reader at once. In case you are only using some of the outputs for controlling a multiplexer (e.g. a 4- or 8-port multiplexer) still all 4 pins are set every time. Upper bits are set to zero. This is also true for the AUT mode.

## 2.24.1. Activate specific Antenna using MAN

In case you want to activate a specific antenna you just need to supply its number - the first antenna being antenna 0.

### Instruction

**SAP** <SPACE> **MAN** <SPACE> {Antenna Port} <CR>

### Parameters

Name	Type	Description
MAN	Keyword	Flag to manual activate specific port
Antenna Port	Decimal Integer (0 ≤ x ≤ 15)	Number of antenna port to activate

### Examples

```
SAP MAN 10<CR>
```

Example 43. Set 16 port multiplexer to activate antenna 10

## 2.24.2. Activate specific Antenna not using MAN

In case you want to activate a specific antenna you just need to supply its number - the first antenna being antenna 0.



### Note

This variant is deprecated and should not be used. SAP X is replaced by SAP MAN X

### Instruction

**SAP** <SPACE> {Antenna Port} <CR>

### Parameters

Name	Type	Description
Antenna Port	Decimal Integer (0 ≤ x ≤ 15)	Number of antenna port to activate

## Examples

```
SAP 10<CR>
```

Example 44. Set 16 port multiplexer to activate antenna 10

### 2.24.3. Automatic switching mode

In case you want to automatically switch between multiple antennas (e.g. trying to find all tags in a search area that can only be searched using multiple antennas) you can use this automatic switching mode which requires you to specify the number of antenna ports participating. The reader will switch between all antennas, automatically changing antenna after every command that addresses tags (Tag Manipulation Instruction).

#### Instruction

```
SAP <SPACE> AUT <SPACE> {No. Antennas} <CR>
```

#### Parameters

Name	Type	Description
No. Antennas	Decimal Integer ( $0 \leq x \leq 16$ )	Number of antennas connected. Zero is not strictly forbidden and equals MAN 0 due to implementation. Don't use AUT 0.

## Examples

```
SAP AUT 5<CR>
```

Example 45. Set the reader to automatically switch between the first five antennas (antennas 0-4) with every reading instruction

### 2.24.4. Show current state

This parameter show the current state. It shows the mode first (MAN or AUT) followed by the current antenna state. If the host changed the IOs by other means (like WOP, RIP, HOT) the IOs might have another state than used last. If AUT mode is started it will then show the number of antennas. Last (also optional) token is ARP showing the automatic antenna reporting is enabled.

#### Instruction

```
SAP <SPACE> SHW <CR>
```

## Examples

```
SAP SHW<CR>
```

Example 46. Get current antenna settings

### 2.24.5. Automatic antenna reporting

This parameter

## Instruction

**SAP** <SPACE> ARP <SPACE> { ON | OFF } <CR>

## Parameters

Name	Type
New state	Enumeration (ON or OFF)

## Examples

```
SAP ARP ON<CR>
```

Example 47. Enable automatic antenna report

## Return Values in Case of Success

OK! <CR>

## Return Values in Case of Failure

"NOR <CR>", "EDX <CR>", "NOS <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.25. Echo (ECH)

This command takes whatever the host gives as parameter and echoes that parameter. The echo is just normally added with <CR> and, if activated, <LF>. Any value may be echoed except <CR> which marks the command end. Lower case data will be echoed upper case.

```
>> ECH HELLO<CR>
```

```
<< HELLO<CR>
```

```
>> ECH hello<SPACE><SPACE>2<CR>
```

```
<< HELLO<SPACE><SPACE>2<CR>
```

Example 48. **ECH** command and answer

## Instruction

**ECH** <SPACE> {...String to echo...} <CR>

## Return Values in Case of Success

ECHO\* <CR>

## Return Values in Case of Failure

"WDL <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.26. Verbosity Level (VBL)

This command allows the user to adjust the amount of communication coming from the reader. For VBL set to zero a minimum amount of data is sent - e.g. the answer to an inventory request is empty (no answer at all) in case no tag is found and contains only the tag ID(s) without the number of tags found in case there are tags. In case of the default setting of one the answers correspond to what is shown in this documentation. Setting the verbosity level to two will also output debugging info to a certain extent. Without a parameter it reports

### Instruction

**VBL** [<SPACE> { 0 | 1 | 2 | SHW }] <CR>

### Parameters

Name	Type
Mode	Optional Enumeration (0, 1, 2 or SHW)

### Examples

```
VBL 2<CR>
```

Example 49. Set verbosity level to 2 (default value)

## Return Values in Case of Success

"OK! <CR>", "0 <CR>", "1 <CR>" or "2 <CR>"

## Return Values in Case of Failure

"EDX <CR>", "NOR <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.27. Mode (MOD)

The **MOD** is not supported on this device as only ISO15693 is supported. It is the default value and not changeable. The mode command selects the anti-collision and transmission protocol to be used for tag communication. Any setting will answer NOS.

## Instruction

**MOD** <SPACE> { 156 | 14A | 14B } <CR>

## Parameters

Name	Type
ISO Standard	Enumeration (156, 14A or 14B)

## Examples

```
MOD 156<CR>
```

*Example 50. Setting the reader configuration to read ISO 15693 tags (this is already set as the default)*

## Return Values in Case of Success

OK! <CR>

## Return Values in Case of Failure

"NOS <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.28. Settings (SET)

Command that in combination with the parameters described in the following subchapters allows configuring reader behaviour.

### 2.28.1. High on Tag values (HOT)

This parameter is only usable on readers that have output pins. It sets an output to high and low for a specified time if a valid tag communication happened. A valid tag communication can be a tag found in an inventory round (as a result of an **INV** command) or any tag answer to a command from the **REQ** command family that passes the CRC integrity test. It is identical to the IVF XX answer with XX not zero. Please note that a correct error message from the tag will also initiate this response. The output pin being temporarily set to high state is GPIO 7 for the Dwarf15 and QR15 and output 0 for the QuasarMX. The time set is in ms. This enables the "High on tag"-mode



## Warning

In case of the QR15 and the Dwarf15 the output pins can also be used as input pins (General Purpose Inputs / Outputs - GPIOs). When calling **SET HOT** and every time a tag is found the direction the pin is being used in is changed to being an output pin. Please make sure that the hardware connected to the pin is compatible to the pin's maximum limits in output mode before calling **SET HOT** as this can otherwise destroy the reader or connected hardware. It is also mutual exclusive to SET IHC / EGC, RIP, WOP, SAP or other IO usage on the same pin. This may cause unexpected or damaging behaviour.

## Instruction

```
SET <SPACE> HOT <SPACE> {High time} <SPACE> {Low time} <CR>
```

## Parameters

Name	Type
High time	Decimal Integer ( $0 \leq x \leq 255$ )
Low time	Decimal Integer ( $0 \leq x \leq 255$ )

## Examples

```
SET HOT 10 150<CR>
```

Example 51. Set the reader to turn the output on for 10ms and then off for 150ms in case a tag is found

```
SET HOT 100 50<CR>
```

Example 52. Set the reader to turn the output on for 100ms and then off for 50ms in case a tag is found

## 2.28.2. High on Tag flags (HOT)

OFF flag will disable the access to the IO (default state) and SHW flag returns the times if given or OFF if it is disabled. Setting the times to zero is not identical to OFF.

## Instruction

```
SET <SPACE> HOT <SPACE> { OFF | SHW } <CR>
```

## Parameters

Name	Type
Flag	Enumeration (OFF or SHW)

## Examples

```
SET HOT SHW<CR>
```

Example 53. Gets the current SET HOT state

```
SET HOT OFF<CR>
```

Example 54. Disables 'High on tag' mode

### 2.28.3. Configure Input High Commands (IHC)



#### Note

This subcommand is deprecated and replaced by **EGC**. Any information there can be applied to SET IHC, too.

#### Instruction

```
SET <SPACE> IHC <SPACE> {Pin} <SPACE> {...} <CR>
```

#### Parameters

Name	Type
Pin	Decimal Integer ( $0 \leq x \leq 1$ )

### 2.28.4. Set Input High Commands (IHC)

#### Instruction

```
SET <SPACE> IHC <SPACE> {Pin} <SPACE> {...} <CR>
```

#### Parameters

Name	Type
Pin	Decimal Integer ( $0 \leq x \leq 1$ )

#### Examples

```
SET IHC 1 INV<CR>
```

Example 55. Set the reader to search for tags once input pin 1 goes high

### 2.28.5. Power Level (PWR)



#### Note

SET PWR is deprecated. It is replaced by **SRI PWR**

#### Instruction

```
SET <SPACE> PWR <SPACE> { 100 | 200 | SHW } <CR>
```

#### Parameters

Name	Type
Power level	Enumeration (100, 200 or SHW)



## Examples

```
SET PWR 100<CR>
```

Example 56. Set power level to 100mW (e.g. for 2nd generation DeskID\_ISO)

```
SET PWR SHW<CR>
```

Example 57. Get power level

## Return Values in Case of Success

Answer <CR>

**SET** commands usually answer with

OK!

except for SHW which gives the current values of the parameters.

OK! <CR>

## Return Values in Case of Failure

"UPA <CR>", "NOS <CR>", "NOR <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.29. Set RF Interface (SRI)

The Set RF Interface Command is used to configure the RF interface of the reader and to turn the RF power on and off. As the ISO norm allows for different tag behaviour, the command allows adjusting the number of subcarriers (single or double) and the modulation depth. Unless the datasheet of the tag IC explicitly states something else (note: this is rare) the choice of single subcarrier and 100% modulation depth is the correct choice (see following subchapters). One of this needs to be set before communicating with tags. If not set the NRF error is given when trying to communicate with tags.



### Note

Before interacting with tags the RF needs to be enabled.

### 2.29.1. RX1 input

The QuasarMX, QR15, UM15, DeskID ISO and Dwarf15 starting with the second generation hardware (firmware versions  $\geq 3.0$ ) have two ports for receiving the RF signal from the tag. These two ports are always connected to the same antenna but differ slightly in their behaviour. In case you are using an external antenna it can make sense to compare reading performance on both ports to see which works better. The **SRI RX1** command allows selecting the first port (which is also the default setting).

## Instruction

**SRI** <SPACE> RX1 <CR>

## Examples

```
SRI RX1<CR>
```

*Example 58. Set reader to use the first receiving port (which is the default)*

### 2.29.2. RX2 input

The QuasarMX, QR15, UM15, DeskID ISO and Dwarf15 starting with the second generation hardware (firmware versions >= 3.0) have two ports for receiving the RF signal from the tag. These two ports are always connected to the same antenna but differ slightly in their behaviour. In case you are using an external antenna it can make sense to compare reading performance on both ports to see which works better. The **SRI** RX2 command allows selecting the second port.

## Instruction

**SRI** <SPACE> RX2 <CR>

## Examples

```
SRI RX2<CR>
```

*Example 59. Set reader to use the second receiving port*

### 2.29.3. Single Subcarrier

The ISO 15693 allows the tag IC manufacturers to specify whether the tags use single or double subcarrier communication with the reader. The vast majority of tag ICs use single subcarrier modulation so unless you are having trouble communicating with your tag or the datasheet specifies double subcarrier, single subcarrier (SS) is the setting of choice. Also, a vast majority of all tags use 100% ASK modulation depth - in rare cases a modulation depth of 10% can also be found.

## Instruction

**SRI** <SPACE> SS <SPACE> { 10 | 100 } <CR>

## Parameters

Name	Type
Modulation depth	Enumeration (10 or 100)

## Examples

```
SRI SS 100<CR>
```

*Example 60. Activate RF for interacting with tags expecting single subcarrier, 100% ASK modulation (probably 99% of all current tags)*

## 2.29.4. Double Subcarrier

The ISO 15693 allows the tag IC manufacturers to specify whether the tags use single or double subcarrier communication with the reader. As the vast majority of tag ICs use single subcarrier modulation do not use DS unless you are having trouble communicating with your tag or the datasheet specifies double subcarrier modulation. Also, a vast majority of all tags use 100% ASK modulation depth - in rare cases a modulation depth of 10% can also be found.

### Instruction

```
SRI <SPACE> DS <SPACE> { 10 | 100 } <CR>
```

### Parameters

Name	Type
Modulation	Enumeration (10 or 100)

### Examples

```
SRI DS 100<CR>
```

*Example 61. Activate RF for interacting with tags expecting double subcarrier, 100% ASK modulation (unusual, please check tag IC datasheet)*

## 2.29.5. Disable RF

In case you want to manually turn off the RF field you can use the OFF parameter with the SRI command.

### Instruction

```
SRI <SPACE> OFF <CR>
```

### Examples

```
SRI OFF<CR>
```

*Example 62. Deactivate RF manually*

## 2.29.6. Config T3 mode

ISO15693 states the real T3 (wait time) after EOF may differ depending on 100% or 10% modulation (ASK10 / ASK100 (also OOK)). For ASK10 the nominal response time Tnrt has to be added. This only applies to the inventory slot EOF so the T3Mode only changes **INV** command. The T3Mode is default set to AUTO so Tnrt is added depending on the set modulation depth. The host can set this to long (LNG) or short (SHT) manually if the tag has a different behaviour (like always using the short variant). Tnrt can be set in **STT** command.

### Instruction

```
SRI <SPACE> T3M <SPACE> { SHT | LNG | AUTO } <CR>
```

## Parameters

Name	Type
Mode	Enumeration (SHT, LNG or AUTO)

## Examples

```
SRI T3M SHT<CR>
```

*Example 63. Set T3M to short mode*

### 2.29.7. Disable for some Time

In case you want to automatically turn off the RF field for a specified amount of time (e.g. to depower a tag) you can use the `TIM` parameter with the time to turn the field off in milliseconds as a second parameter. The settings the RF interface had before the `TIM` are automatically restored once the field is turned back on. Calling without active RF will cause `NRF` error. This includes disabled power due to power saving mode. Other `SRI` commands may set the RF interface back before the time is up.

## Instruction

```
SRI <SPACE> TIM <SPACE> {Time} <CR>
```

## Parameters

Name	Type
Time	Decimal Integer ( $1 \leq x \leq 2000$ )

## Examples

```
SRI TIM 100<CR>
```

*Example 64. Deactivate RF for 100 ms*

### 2.29.8. Config power level

The device can work with 2 power levels. They are 200 and 100 Milliwatts.

## Instruction

```
SRI <SPACE> PWR <SPACE> { 100 | 200 } <CR>
```

## Parameters

Name	Type
Power value	Enumeration (100 or 200)

## Examples

```
SRI PWR 100<CR>
```

*Example 65. Set RF Transmitter output power to 100 mW*

### 2.29.9. Config power saving

Devices can have 3 different settings for power saving. OFF equals no power saving so the RF TX is run all the time if SRI enables it. OFF does not accept timing parameter! Burst mode disables TX after the disable time is over without any tag manipulation command. In slowing down mode the TX is disabled after every tag manipulation command for at least the disable time. In both burst and slowing down mode the TX is enabled "power up time"  $\mu$ s before the tag manipulation is executed if TX was deactivated (so in burst mode no time is lost if multiple commands are executed in short time). For slowing down mode the power down time is max 1000 (not 300k). Both saving modes have default timings.

#### Instruction

```
SRI <SPACE> SAV <SPACE> { OFF | BST | SLW } [<SPACE> {Disable time in milliseconds} [<SPACE> {Power up time in microseconds}]] <CR>
```

#### Parameters

Name	Type
Power saving state	Enumeration (OFF, BST or SLW)
Disable time in milliseconds	Optional Decimal Integer ( $0 \leq x \leq 300000$ )
Power up time in microseconds	Optional Decimal Integer ( $0 \leq x \leq 1000000$ )

#### Examples

```
SRI SAV BST<CR>
```

Example 66. Set power saving to burst with default timing

```
SRI SAV SLW 300<CR>
```

Example 67. Set power saving to slowed mode with default power up and 300ms power down

### 2.29.10. Get current RF state

This gets all informations set by SRI back. The answer has 5 lines. Every line contains the data as they are given (without the SRI) with optional values printed out. First line contains the SRI state (OFF, SS 10, SS 100, DS 10, DS 100). Second line contains the receiver line (RX1 / RX2). The third line contains the power saving state. The 4th line contains the power setting. The 5th line contains the T3M mode.

#### Instruction

```
SRI <SPACE> SHW <CR>
```

#### Examples

```
SRI SHW<CR>
```

Example 68. Get SRI settings

## Return Values in Case of Success

OK! <CR>

## Return Values in Case of Failure

"EDX <CR>", "NOS <CR>", "NRF <CR>", "NOR <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

### 2.30. Set Timings (STT)

The Set Timings Command is used to write parameters used to control request or inventory timings.



#### Note

This command should only be used with deeper knowledge of ISO 15693 and only in case something needs to be fine tuned. Under all normal circumstances the default values should work and timings shouldn't be changed. Be sure to have the tag IC datasheet available before modifying these values. In case your changes have unexpected / undesirable consequences resetting the reader will restore the default settings. All values (except TWMIN) can be changed to be carrier clock wise instead of  $\mu\text{s}$  by adding TCK parameter before the value. This is usual more precise.

#### 2.30.1. Set Timings T1MAX, T1MIN, T2, T3 and TWMAX

- T1MAX: T1 can be used equivalently. This is the maximum answer time from the end of sending any reading request (including inventory requests) to the beginning of the answer. Answers coming after this time will be ignored. The value is set in microseconds. Default value is 350 $\mu\text{s}$ . Maximum value is 38600 $\mu\text{s}$ .
- T1MIN: This is the minimum answer time for any reading request (including inventory request) to the beginning of the answer. Answers coming before this time will be ignored. The value is set in microseconds. Default value is 280 $\mu\text{s}$ . Maximum value is 38600 $\mu\text{s}$ . Values of 350 $\mu\text{s}$  are fine for some tags, this will give better results, especially in noisy environments.
- T2MIN: T2 can be used equivalently. This is the minimum time after any answer (including an inventory answer) to the beginning of the next request. Commands coming before this time will be delayed. The value is set in microseconds. Default value is 350 $\mu\text{s}$ . Maximum value is 38600 $\mu\text{s}$ .
- T3MIN: T3 can be used equivalently. This is the minimum time after any request (including inventory requests) to the beginning of the next request. Commands coming before this time will be delayed. The value is set in microseconds. Default value is 400 $\mu\text{s}$ . Maximum value is 38600 $\mu\text{s}$ . A T3 value that is set too high has no negative side effects except reducing speed.



## Note

The T3 value may be different not only by tag by communication mode. See TNRT for this, too. needs to be changed for inventory requests on tags requiring 10% ASK modulation compared to 100% ASK mode tags! Add the nominal response time (T nrt ) to the value used with 100% ASK if the value is given in the tag IC data sheet. If not try values manually. Typical values are around 4000µs. Maximum value is 38600µs.

- TNRT: This time is added to T3 in specific conditions settable by SRI T3M. It defaults to the answer time of an inventory answer which is 12 bytes + SoF + EoF. This equals 12 \* 512 (equals 1 bytes) + 2048 (equals EOF of SS which is a bit longer than EOF of DS) + 2048 (equals SOF of SS which is a bit longer than SOF of DS). This sums up to 10.240 Ticks. SRI T3M is defaulted to AUTO meaning it is added to communication if the ASK is 10.



## Note

This equals parts of the nominal response time as defined in SO15693-3:2000(E), 8.4 (Timing definitions). This time is both tag and answer dependent so we used a default inventory answer length as the best approximation. If another value is needed for your tag or its usage please change the value accordingly.

- TWMAX: This is a timer for write-alike requests (WRQ, DWQ). With option flag set this is the time from the end of the initial request to the end of frame triggering the answer. Without option flag it is the timeout value. Any answer after this time will be ignored. The value is set in microseconds. Default value is 20000µs. A shorter time might be ok for your tag (or larger time). Maximum value is 38600µs.
- TWMIN: This is the start time of write-alike answers. The answer is expected to be send between this and this + TWWND + n \* (4096 Ticks), n is a natural number >= 0. The window repeat time (4096 ticks) is given by ISO15693 and is not changeable.
- TWWND: This is the window size. It is given in ticks as it is short. In write-alike requests (WRQ, DWQ) the receiver is activated every TWMIN + n \* (4096 TICKs) and deactivated TWWND Ticks later.

## Instruction

```
STT <SPACE> { T1MAX | T1 | T1MIN | T2 | T2MIN | T3 | T3MIN | TWMAX  
| TNRT } <SPACE> {Value} <CR>
```

## Parameters

Name	Type
Timing Type	Enumeration (T1MAX, T1, T1MIN, T2, T2MIN, T3, T3MIN, TWMAX or TNRT)
Value	Decimal Integer ( $0 \leq x \leq 38600$ )

## Examples

```
STT T1MAX 400<CR>
```

Example 69. Set T1MAX to 400µs (the default value)

### 2.30.2. Set Timing TWMIN

TWMIN: This is a timer for writing requests. With option flag this has no meaning. Without option flag it defines the start value of the first receive window. As this will affect any receive window this value is very critical to all writing requests. Therefore the value is NOT in microseconds but in 8 ticks of base frequency (please read the ISO 15693 for a better understanding). Default is 550 (550\*8 ticks = 4400 ticks = 324,5µs). Maximum value is 0xFF00.

#### Instruction

```
STT <SPACE> TWMIN <SPACE> {Value} <CR>
```

#### Parameters

Name	Type
Value	Decimal Integer ( $0 \leq x \leq 65280$ )

## Examples

```
STT TWMIN 550<CR>
```

Example 70. Set TWMIN to 4400 ticks (the default value)

### 2.30.3. Set Timing TWWND

TWWND: This is the window time (the size of the writing request answer windows). This value is just the size, the window always starts at TWMIN and restarts cyclically as defined in the ISO 15693. The value is given in 8 ticks of the base frequency. The default value is 16 (128 ticks (=9,44µs)). Changing this value is discouraged and should only be done in case of real need! Maximum value is 255.

#### Instruction

```
STT <SPACE> TWWND <SPACE> {Value} <CR>
```

#### Parameters

Name	Type
Value	Decimal Integer ( $0 \leq x \leq 255$ )

## Examples

```
STT TWWND 16<CR>
```

Example 71. Set TWWND to 128 ticks (the default value)



## Return Values in Case of Success

OK! <CR>

## Return Values in Case of Failure

"NOR <CR>", "EDX <CR>", "NOS <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## Chapter 3. Tag Manipulation Instructions

Tag Instructions target the tag itself. Since RFID is mostly about tags, their IDs and data stored on tags, the Tag Manipulation Instructions are used extensively in almost any program.

These commands can be prefixed with the CNR command. It changes the tag manipulation instruction following to be executed repeatedly. This CNR mode is stopped by a BRK

Command	Name	Description
<b>INV</b>	Inventory	Inventories tags by unique ID.
<b>REQ</b>	Reading Request	Command to send ISO 15693 command strings to the tag and get its answer.
<b>WRQ</b>	Writing Request	Version of the <b>REQ</b> command using writing timeouts
<b>DRQ</b>	Direct Reading Request	Direct address variant of <b>REQ</b>
<b>DWQ</b>	Direct Writing Request	Direct address version of the <b>WRQ</b> command

Table 2. Overview of Tag Manipulation Instructions

### 3.1. Inventory (INV)

The ISO 15693 specifies only two mandatory commands that all tags must support. One of these is the inventory command which is used to find tags and read their IDs. It is called inventory command because it allows finding all tags in the field using an anti collision sequence. The reader's **INV** command uses the ISO 15693 low level command and performs the anti collision sequence returning the IDs of all tags found.

The inventory command will return the tag IDs found one per line with each line terminated by <CR>. After the IDs of that inventory round have all been reported an additional line is reported back which consists of the keyword **IVF** followed by <SPACE> (0x20) and a two digit number of tags found (e.g. 00 in case no tags were found or 08 in case 8 tags were found). The maximal number of tags stored and reported is 32. Additional tags would be dropped and inventory cycle stopped.

#### 3.1.1. Simple Inventory

In its simplest form, the command consists only of **INV** <CR>. For convenience reasons there are some optional parameters that will allow making some higher level tag ID searches much easier and which will be explained in the following subchapters.

#### Instruction

**INV** <CR>

#### Examples

```
INV<CR>
```

Example 72. Find all tags in the field

### 3.1.2. Application family identifier (AFI)

Application Family Identifier: Setting this optional parameter will lead to only tags with the corresponding AFI answering the INV command - tags in other AFI groups will not answer. This can be used to filter the type of tags responding. Some AFI values are reserved for specific applications - please see the ISO 15693 for further reference.

#### Instruction

```
INV <SPACE> AFI <SPACE> {Application family identifier} <CR>
```

#### Parameters

Name	Type
Application family identifier	Hexadecimal Integer ( $0_{16} \leq x \leq FF_{16}$ )

#### Examples

```
INV AFI 50<CR>
```

*Example 73. Find all tags with application family 0x50 (medical)*

### 3.1.3. Single Slot Inventory (SSL)

Single Slot: If the user can be sure to have only one tag in the reader field at any time, setting this optional parameter makes the reader scan for tag IDs faster (as anti collision is disabled). Bear in mind that if there is more than one tag in the field, you may get a CLD (Collision Detected) error, TCE (Tag communication) error or, if one is a lot stronger coupled, maybe only this one tag is reported.

#### Instruction

```
INV <SPACE> SSL <CR>
```

#### Examples

```
INV SSL<CR>
```

*Example 74. Quickly find single tag in field*

### 3.1.4. Only New Tag (ONT)

The **ONT** parameter makes the reader find each tag only once by automatic use of "stay quiet" command. This causes the tag to not answer any unaddressed and inventory commands long as it stays in quiet state as ISO15693-3 states. It will be set to other states by depowering, "select" command and "reset to ready" command. This command is mostly used in conjunction with the **CNR** prefix to allow detecting all tags once when they enter the field. If **ONT** is not set, the tags will be reported in every inventory cycle as long as they are present within the field which may be disturbing or cause a lot of communication traffic. It might even go over the max number of tags the reader can store and report (which is 32).



## Warning

The TAG will not answer to any request or inventory command until it is repowered. This is also true if the command has no ONT. It uses the "Stay Quiet" command (0x02).

### Instruction

**INV** <SPACE> ONT <CR>

### Examples

```
INV ONT<CR>
```

*Example 75. Find tag only once (as long as it stays powered)*

### 3.1.5. Masking (MSK)

**MSK** Masking allows to just search for specific tags. This might contain the whole tag ID or just parts. The masking starts at the end meaning **MSK** 345 brings all tags with UID XXXXXXXXXXXXXXX345.

### Instruction

**INV** <SPACE> MSK <SPACE> {Mask to Set} <CR>

### Parameters

Name	Type
Mask to Set	Hexadecimal String

### Examples

```
INV MSK 123<CR>
```

*Example 76. Find all tags whose IDs end with 123*

### Return Values in Case of Success

E0... (Tag IDs, one per line) <CR>

### Return Values in Case of Failure

"BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 3.2. Reading Request (REQ)



### Note

This is the base request command. If the command includes address the address is inverted. The timing is the default read-alike timing. To get different modes use the otherwise identical request commands

- **WRQ** (write-alike timing)
- **DRQ** (direct sending (no inverting of address bytes))
- **DWQ** (direct sending, write-alike timing)

For custom commands (not ISO15693-3 defined) use DRQ and DWQ as they send as given.

The ISO 15693 defines the requests a tag has to understand as well as a set of optional commands. Also, it allows for manufacturer specific commands to be defined in the tag IC datasheet. All readers will support the mandatory and some optional commands by recognizing the write-alike commands and handling them as such even if **REQ** is used. If the tag used does not conform to ISO15693-3 in the way that Write or Lock command are timed read-alike please use DRQ to not get into that problem.

The reason for this on first look complicated differentiation is the fact that the addressing of tags in the ISO 15693 is counterintuitive. Usually when talking about tag IDs, we read them most significant byte first - the typical E00 . . . tag IDs are formatted this way. However, when a tag is to be addressed with a command the order of the bytes (meaning two digit blocks) is reversed. The **INV** command gives the tag IDs in the usual way. If one wants to use an addressed command it would usually be necessary to switch around the bytes of the tag ID that were received to get a correct address.

As the command format easily allows the firmware to detect whether the command is addressing a certain tag and usually has the address in a fixed location the **REQ** and **WRQ** commands will automatically switch the bytes around to fit. Basically, with these commands you can use the tag ID as it was read from the tag with **INV** when assembling the command string. However, there are special cases with unusual tag ICs when the usual ISO 15693 location for the tag address is not being used or commands do not fit the expected timing type. In such cases the "direct" versions have to be used instead and in this case the tag ID has to be reformatted according to the ISO 15693 or tag specific requirements. Basically, with the "direct" commands the tag is served exactly the data you pass to the reader. If in doubt, consult the tag IC datasheet or just try using the **REQ** and **WRQ** commands first and only if that doesn't work try the "direct" versions.

The format and contents of the hexadecimal command string is defined in the ISO 15693-3 and in case of custom tag commands in the datasheet of the respective tag IC. Please refer to these sources for full details. metraTec also supplies an extensive set of documented, predefined hexadecimal example strings for use with our metraTerm terminal program.

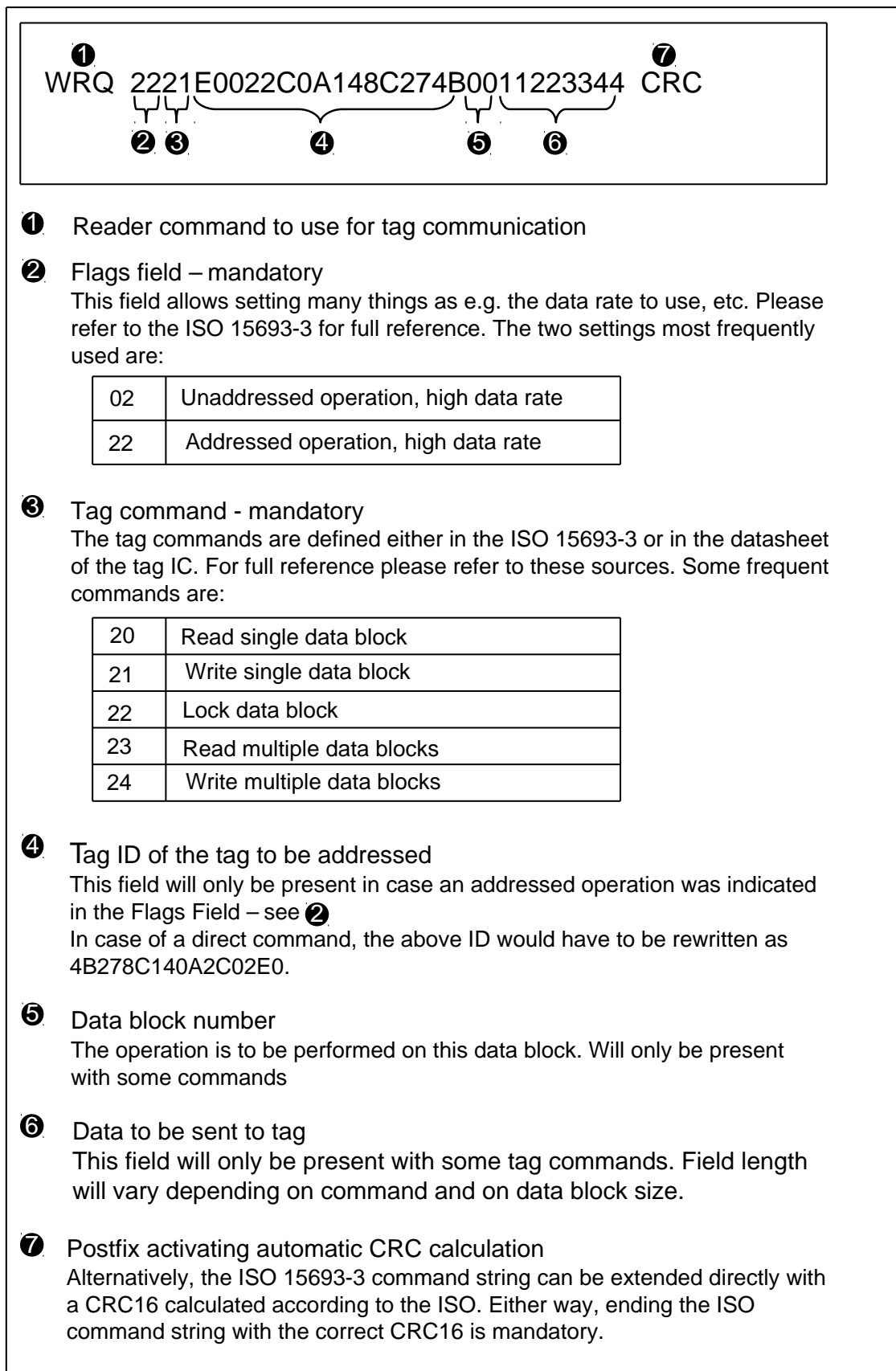


Figure 1. A simplified overview of ISO 15693-3 command syntax

As the ISO 15693 specifies that the hexadecimal string is to be secured against transmission errors using a CRC16 the user has the choice of either adding the correctly calculated (accord-

ing to the ISO 15693 definition) CRC16 directly to the command string or let the firmware calculate and add the correct CRC16 automatically. The firmware will add the CRC16 automatically if the postfix CRC is added.



### Note

This CRC at the end of the ISO 15693 command string is mandated by the norm and secures the over the air communication. It has nothing to do with the CRC checking mode that can be enabled by the **CON** command used to secure correct data transmission between reader and host.



### Note

Please note that the ISO 15693-3 defines commands for reading and writing multiple blocks (command codes 23 and 24). Most modern tag ICs will have implemented these optional commands according to the ISO 15693. However, as these commands and their answers are transmitted over the air between the reader and the tag there is a certain likelihood of transmission errors. The longer the data stream becomes the more likely such transmission errors occur. In case the reader or tag notice transmission errors they will notify the user that such an error has occurred but not at which point. This can lead to the user having to retransmit the complete data several times until it is sent successfully. It is therefore strongly suggested to use the multiblock commands carefully and especially to choose the amount of data to transmit taking the data integrity issue into consideration. Please also conduct tests which determine the effect of electromagnetic disturbances in the location the reader is going to be used as soon as possible as the environment might change the ideal data transfer length.

The response to all four requests either consists of a single line indicating that the transponder is not responding (TNR) or of four lines in case at least one transponder has responded. The first line will then be TDT to show that a tag was detected. The second line will contain the tag answer as defined by the ISO 15693-3. Its first two digits will usually signify whether the command was carried out successfully - 00 indicating success. Please consult the tag IC datasheet or the ISO for the meaning of any error responses you might get. After the success flag there might be data reported back by the tag (e.g. in case of a read data request) and the answer will usually end with the CRC16 computed according to the ISO over the content of the answer.

The third line will either be COK in case the over the air transmission was found to be without CRC errors or CER in case a CRC error was detected. This is of course related to the ISO15693 CRC calculation so if the tag happens to differ from this the CRC needs to be checked by the host. The fourth line will finally either be NCL in case no tag data transfer collisions were detected or CDT in case such a collision was detected.



### Note

Please note that it is not necessary to address a request to a specific tag as can be seen in the discussion of the ISO command syntax. However, whenever an unaddressed command is used multiple tags might react (execute the command) while the answer collides and can't be received. If such a collision occurs neither the tag answer can be assumed to be correct nor can you make any assumptions about whether the original command by the reader was carried out (successfully).

## Instruction

```
REQ <SPACE> { ISO Command } <SPACE> CRC <CR>
```

## Parameters

Name	Type
ISO Command	Hexadecimal String

## Examples

```
REQ 022003 CRC<CR>
```

Example 77. Read block 03 of a single tag in the field without addressing it

```
REQ 2220E0022C0A148C274B03 CRC<CR>
```

Example 78. Read block 03 of a tag with the ID E0022C0A148C274B

## Return Values in Case of Success

"TDT{CR}[DATA]{CR}COK{CR}NCL <CR>" or "TDT{CR}[DATA]{CR}CCE{CR}NCL <CR>"

## Return Values in Case of Failure

"TNR <CR>", "CLD <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 3.3. Writing Request (WRQ)

This is the version of the **REQ** command meant to be used whenever data on the tags is to be changed / written. This is the normal version of this command that will rearrange the byte order of the tag ID automatically as long as the address is in the expected location (and addressed bit is set). This command uses the writing timeouts and will not work reliably in cases where data is to be read only from the tag. For further details please check the chapter on the **REQ** command.

### Instruction

```
WRQ <SPACE> {ISO Command} <SPACE> CRC <CR>
```

## Parameters

Name	Type
ISO Command	Hexadecimal String

## Examples

```
WRQ 02210312345678 CRC<CR>
```

Example 79. Write 12345678 to block 03 of a tag without addressing it



```
WRQ 2221E0022C0A148C274B0312345678 CRC<CR>
```

Example 80. Write 12345678 to block 03 of a tag with the ID E0022C0A148C274B

### Return Values in Case of Success

"TDT{CR}[DATA]{CR}COK{CR}NCL <CR>" or "TDT{CR}[DATA]{CR}CCE{CR}NCL <CR>"

### Return Values in Case of Failure

"TNR <CR>", "CLD <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 3.4. Direct Reading Request (DRQ)

This is the "direct" version of the **REQ** command meaning that the ISO command is not parsed but sent directly to the tag. Please make sure that the order of the bytes in any tag ID being used for addressing is in order as ISO 15693 specifies it. This command uses the reading timeouts (unless it is a known write-alike command). The command is still executed most likely. For further details please check the chapter on the **REQ** command.

### Instruction

**DRQ** <SPACE> {ISO Command} <SPACE> CRC <CR>

### Parameters

Name	Type
ISO Command	Hexadecimal String

### Examples

```
DRQ 022003 CRC<CR>
```

Example 81. Directly read block 03 of a single tag in the field without addressing it

```
DRQ 22204B278C140A2C02E003 CRC<CR>
```

Example 82. Directly read block 03 of a tag with the ID E0022C0A148C274B

### Return Values in Case of Success

"TDT{CR}[DATA]{CR}COK{CR}NCL <CR>" or "TDT{CR}[DATA]{CR}CCE{CR}NCL <CR>"

### Return Values in Case of Failure

"TNR <CR>", "CLD <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

### 3.5. Direct Writing Request (DWQ)

This is the version of the **REQ** command meant to be used whenever data on the tags is to be changed / written and the addressing data is not located in the expected location. The command string is directly sent to the tag IC as is. Please make sure that the tag ID is formatted as expected by the ISO 15693. This command uses the writing timeouts and will not work reliably in cases where data is to be read only from the tag. The option flag in the flag byte will cause timings like in write single block command both with option flag enabled and disabled. For further details to requests please check the chapter on the **REQ** command.

#### Instruction

**DWQ** <SPACE> {ISO Command} <SPACE> CRC <CR>

#### Parameters

Name	Type
ISO Command	Hexadecimal String

#### Examples

```
DWQ 02210312345678 CRC<CR>
```

Example 83. Directly write 12345678 to block 03 of a tag without addressing it

```
DWQ 22214B278C140A2C02E00312345678 CRC<CR>
```

Example 84. Write 12345678 to block 03 of a tag with the ID E0022C0A148C274B

#### Return Values in Case of Success

"TDT{CR}[DATA]{CR}COK{CR}NCL <CR>" or "TDT{CR}[DATA]{CR}CCE{CR}NCL <CR>"

#### Return Values in Case of Failure

"TNR <CR>", "CLD <CR>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## Chapter 4. Precommands

Precommands may help parsing answers or detecting the answer of a device (if more than one device share a communication line).

The precommands always start and end with '#', contain a one character command and the following signs as (optional) parameters.

### 4.1. Command Answer Prefixes ('P')

This command sets a prefix of up to 16 characters which is sent before any answer line. To the prefix a space character (0x20) is appended. The prefix will stay active until it is deleted (using the command without following parameters (#P#)). This may be used to detect specific devices or to "tag" specific answers to specific commands.

```
» #PDEVICE1#INV<CR>
```

```
« DEVICE1 E200FFA039C992<CR>  
DEVICE1 IVF 001<CR>
```

```
» INV<CR>
```

```
« DEVICE1 E200FFA039C992<CR>  
DEVICE1 IVF 001<CR>
```

```
» #P#INV<CR>
```

```
« E200FFA039C992<CR>  
IVF 001<CR>
```

*Example 85. Prefix precommand and answers with INV as command*

### 4.2. Answer Counter ('C')

This precommand causes the counter to be reset to zero and be formatted as given. The counter counts up every answer starting by 0 (not every line ending on <CR>, but every complete answer that would end with <LF> if the EOF command were activated). For this reason it may ease parsing. The prefix data may be given as hex or decimal numbers each one or two digits long. Please keep in mind that this leads to the possible counter overflows at 9->0, 99->0, F->0 or FF->0.

The counter is reset every time the C command is called.

The prefix is deactivated by giving '0' as parameter.

The format is set by the parameters D1, D2, H1, H2 to define decimal (D) or hexadecimal (H) and one or two digits, respectively.

```
» #CD1#INV<CR>

« 0 E200FFA039C992<CR>
0 IVF 001<CR>

» INV<CR>

« 1 E200FFA039C992<CR>
1 IVF 001<CR>

» #CD1#INV<CR>

« 0 E200FFA039C992<CR>
0 IVF 001<CR>

» #C0#INV<CR>

« E200FFA039C992<CR>
IVF 001<CR>
```

*Example 86. Counter precommand and answers with INV as command*

### 4.3. Using 'P' and 'C' together

Both precommands can be active at the same time. In this case, the prefixes have to be specified one after the another in the same command line. Setting or reenabling one if the other one is active is not defined. For the answers, the same prefix order will be used as during the command calls. The prefixes can be disabled independently (the prefix not disabled stays active). Setting of prefixes will be executed even if the command is rejected (UCO, UPA, etc.)

```
» #PDEVICE1##CD1#INV<CR>

« DEVICE1 0 E200FFA039C992<CR>
DEVICE1 0 IVF 001<CR>

» INV<CR>

« DEVICE1 1 E200FFA039C992<CR>
DEVICE1 1 IVF 001<CR>

» #P#INV<CR>
```

```
« 2 E200FFA039C992<CR>
  2 IVF 001<CR>
```

Example 87.

#### 4.4. Single Prefix ('SP')

Single Prefix can be used instead of Prefix 'P' if only the direct answer should get the prefix. If command independent error codes (e.g. BOF) occur during the command execution the error code will also be prefixed. The 'SP' may be used together with **SET IHC** to show which input caused the command execution.

```
» #SPINPUT1#INV<CR>
```

```
« INPUT1 E200FFA039C992<CR>
  INPUT1 IVF 001<CR>
```

```
» INV<CR>
```

```
« E200FFA039C992<CR>
  IVF 001<CR>
```

Example 88.

## Chapter 5. Error Codes

Error Code	Name	Description
BOD	BrownOut detected	The microcontroller detected a brownout. This is a hardware error. If it occurs more repeatedly please check your power supply or contact metraTec for support.
BOF	Buffer Overflow	A UART buffer received an overflow error. Send and receive buffer have 768 bytes each.
CCE	Communication CRC Error	A communications error was detected via CRC checksum while receiving a line from the host system.
CER	CRC error	CRC from tag is wrong. Common error sources: <ul style="list-style-type: none"> <li>• The tag left the field or is too far away</li> <li>• Another device disturbed the communication</li> <li>• A collision between tag answers happened</li> </ul>
CLD	Collision Detected	A collision has been detected during tag communication. Common error source: <ul style="list-style-type: none"> <li>• More than one tag in reader field but single slot inventory set or unaddressed operation requested</li> </ul>
CRT	Command Receive Timeout	Parts of a command were received by the reader but the device never received a carriage return. Always close commands with <CR>.
DNS	Did Not Sleep	The <b>WAK</b> command was sent although the reader wasn't in sleep mode
EDX	Error Decimal Expected	Parameter string cannot be interpreted as a valid decimal value. Common error source: <ul style="list-style-type: none"> <li>• Other character than '0' to '9' sent</li> </ul>
EHF	Error Hardware Failure	The RF interface chip does not match or is damaged. Please try a full reset (power reset) and/or contact support.
EHX	Error Hex Expected	Parameter string cannot be interpreted as a valid hexadecimal number.
FLE	FIFO length error	FIFO buffer overrun - Please try reading / writing less data at once.
FRE	Framing error	The framing of the data packet from the tag was malformed. Common error sources: <ul style="list-style-type: none"> <li>• The tag left the field or is too far away</li> </ul>

Error Code	Name	Description
		<ul style="list-style-type: none"> <li>• Another device disturbed the communication</li> </ul> <p>With the QusasrLR this might also be a reader internal error.</p>
NCM	Not in CNR Mode	BRK was used without a running CNR command.
NOR	Number Out of Range	Common error source: <ul style="list-style-type: none"> <li>• RIP, WOP: Value is higher than number of I/O-Pins</li> <li>• Length of data or parameter value is not allowed</li> </ul>
NOS	Not Supported	Command or parameter not supported by this specific device type
NRF	No RF field active	The RF field is off - did you send the correct SRI string?
RDL	Read data length error	The data requested from the tag is too long.
SRT	Watchdog reset	In case of a critical error this error might occur. If you get this error frequently, your hardware is probably damaged.
TCE	Tag Communication Error	General error during tag communication (but tag was found) <p>Common error source: Write command returned wrong check (handle). Data might be corrupted.</p>
TNR	Tag Not Responding	No valid tag answer to tag request (REQ, WRQ, DRQ, DWQ).
TOE	TimeOut Error	The command timed out meaning the timeout value has run out. This may be caused by an unexpected transceiver error or by too many tags / too long data. The timeout is set to 3 s. In case a timeout happens the reader is reset (as would be the case if a <b>RST</b> command had been sent). Please note that this means that the expected answer (including IVF xx) will not be received.
UCO	Unknown Command	An invalid command has been passed to the reader. Common error source: <ul style="list-style-type: none"> <li>• Typo in command string</li> <li>• Wrong firmware version</li> </ul>
UER[<SPACE> {Two Digit Hex Code}]	Unknown ERror	Internal error reached the API unintendedly. The error code is shown hex encoded. With no hex error code: A bad interrupt occurred, unknown internal tag error code returned or

Error Code	Name	Description
		another "default" case occurred. Please contact metraTec with details.
UPA	Unknown Parameter	An invalid parameter has been passed to a function. Common error source: <ul style="list-style-type: none"> <li>• Typo in command string</li> <li>• Given parameter is out of range</li> <li>• Parameter missing (formerly EPX)</li> </ul>
URE	UART Receive Error	UART data corrupted — this is an internal hardware error. Perhaps check the data link cable and EMC.
WDL	Wrong Data Length	The data given is too long or short. This might occur on commands using data of variable length.
WMO	Wrong MOde	A command can not be executed because it is prohibited in a specific mode. For example setting SUC ON when no SUC command is saved



## Appendix A. Quick Start Guide and Examples

The previous chapters have given a thorough reference to the commands the metraTec ISO 15693 readers support. While this reference is necessary it also creates the impression that using the reader is somehow complicated which it is not. In most practical cases a user will only need to send two or three strings to the reader to make it do everything that is needed. Only in special circumstances more is needed. In the following sections, you will find the sequence of commands to the reader that are needed in the most common cases.

### A.1. Typical Reader Initialization Sequence

Before the reader can start reading tags, the RF field has to be activated. This example shows a typical initialization sequence to read ISO 15693 tags. This is probably the first string you need to send to the reader.

```
» SRI SS 100<CR>
```

```
« OK!<CR>
```

This configures the device for single subcarrier 100% ASK mode - the correct mode for probably more than 99% of all tags available today (see Section 2.29, "Set RF Interface (**SRI**)"). The reader will respond with **OK!**. Afterwards the reader is ready to read from and write to tags. For unusual tag types, please check the tag IC datasheet for the correct SRI values to use.

### A.2. Reading the Tag ID of a tag

By far the most common operation done with HF RFID tags is reading the unique ID of a tag. In many cases this is the only thing needed from the tag in which case this is the second (and last) string you need to send to the reader. There are several possibilities to do this with a metraTec device, depending on what exactly you need to do. All operations however are based on the inventory (**INV**) command. The answer gives the tag ID(s) and the number of tags found.

To simply read the IDs of all tags in the field (using anti collision) the simple **INV** command is enough:

```
» INV<CR>
```

If no tag is found, the answer will look like in Example 89, "Inventory answer if no tag has been found". If two tags are found the answer could look like in Example 90, "Inventory answer if two tags have been found".

```
« IVF 00<CR>
```

*Example 89. Inventory answer if no tag has been found*

```
<< E0040100078E3BB0<CR>
E0040100078E3BB7<CR>
IVF 02<CR>
```

Example 90. Inventory answer if two tags have been found

If you are sure that there will be only a single tag in the field, you can use the single slot (SSL) read. This disables the anti collision algorithms and makes the operation even faster. In this mode it is possible to read HF tags with rates of up to 150 tags/sec.

Instruction:

```
>> INV SSL<CR>
```

Possible responses:

```
<< IVF 00<CR>
```

Example 91. Answer to **INV SSL** if there is no tag

```
<< E0040100078E3BB0<CR>
IVF 01<CR>
```

Example 92. Answer to **INV SSL** if there is exactly one tag

```
<< CLD<CR>
IVF 00<CR>
```

Example 93. Answer to **INV SSL** if there is more than one tag

You can also filter the tags that respond to the request using the application family identifier (AFI). To get only the IDs of tags with AFI code 04 use:

```
>> INV AFI 04<CR>
```

The answers are the same as before. Again, you can get a faster response by using the SSL option additionally.

### A.3. Reading Tag IDs continuously

All commands can be processed by the reader continuously by using the **CNR** prefix. With the help of this prefix it is possible to make the reader read the tag IDs of all tags in the field endlessly. It is also possible to adapt this example to read or write to all tags in the field (very useful in tag producing machines or automation scenarios).

Instruction and response with two tags in the field:

```
>> CNR INV<CR>
```

```

« E0040100078E3BB0<CR>
E0040100078E3BB7<CR>
IVF 02<CR>
E0040100078E3BB0<CR>
E0040100078E3BB7<CR>
IVF 02<CR>
E0040100078E3BB0<CR>
E0040100078E3BB7<CR>
IVF 02<CR>
...<CR>
the output will repeat

```

You can stop the endless sequence by sending the break command (**BRK**).

Instruction and response:

```
» BRK<CR>
```

```
« BRA<CR>
```

#### A.4. Example for writing and reading to and from ISO 15693 tags

Next we show how to write and read data to and from a tag in unaddressed mode. Unaddressed mode means that you do not send the command to a specific tag so you do not need to supply the tag ID as part of the command which is then executed by any tag in the field. Please make sure that there is only one tag in the field as you will otherwise get collisions. In this example we write a single block of 4 bytes using the CRC postfix to conveniently have the reader compute the required reader to tag CRC for us.

Instruction and response:

```
» WRQ 02210311112222 CRC<CR>
```

```

« TDT<CR>           Tag detected
0078F0<CR>         00 (status OK), 78F0 (CRC16)
COK<CR>           CRC Okay
NCL<CR>           No collision detected

```

Example 94. Write 11112222 data to block 3

To read the same data we just wrote to the tag, use:

```
» REQ 022003 CRC<CR>
```

« TDT<CR>	Tag detected
0011112222B7DD<CR>	00 (status OK), data read, B7DD (CRC16)
COK<CR>	CRC Okay
NCL<CR>	No collision detected

Example 95. Read the data from block 3

## A.5. Configuring reader to automatically start reading tag IDs when powered

All metraTec readers will wait for commands when first powered. In some cases, however, the user wants the reader to automatically start searching for tags once it is powered and only start sending messages when it finds tags. To configure the reader to do this we use the SUC command and set the verbosity level to minimum so that the reader stays quiet until it finds tags.

```
» SUC SRI SS 100;VBL 0;CNR INV<CR>
```

```
« OK!<CR>
```

The reader will respond with OK! and will start performing in the way specified after it is reset or repowered. In case you want to end the continuous reading mode you will need to send the **BRK** command.

## Appendix B. CRC Calculation

```
1
2  /**
3  * This function calculates a CRC16 over a unsigned char array
4  * with LSB first.
5  *
6  * @param DataBuf Pointer to data to calculate CRC16 for.
7  * @param SizeOfDataBuf Length of the data buffer (DataBuf)
8  * @param Polynom Value of the generator polynom.
9  *           0x8408 is recommended.
10 * @param Initial_Value Initial value of CRC16.
11 *           0xFFFF is recommended for
12 *           host to reader communication.
13 * @return Calculated CRC16
14 */
15 unsigned short GetCrc(unsigned char *DataBuf,
16                      unsigned char SizeOfDataBuf,
17                      unsigned short Polynom,
18                      unsigned short Initial_Value)
19 {
20     unsigned short Crcl6 = Initial_Value;
21     unsigned char Byte_Counter, Bit_Counter;
22
23     for (Byte_Counter = 0;
24         Byte_Counter < SizeOfDataBuf;
25         Byte_Counter++)
26     {
27         Crcl6 ^= DataBuf[Byte_Counter];
28         for (Bit_Counter = 0; Bit_Counter < 8; j++)
29         {
30             if ((Crcl6 & 0x0001) == 0)
31                 Crcl6 >>= 1;
32             else
33                 Crcl6 = (Crcl6>>1)^Polynom;
34         }
35     }
36
37     return (Crcl6);
38 }
39
```

## Revision History

Version	Change	By	Date
1.0	Build a specific documentation from the former generic ISO15 documentation  Added new features and reworks from firmware release 3.9 and for the new QR15_V2 device.  Commands from the old description can usually be used but may be different or have more parameters now or in the future.	MK	27.04.2020
1.1	Fixing minor documentation errors  Changed FW version to 3.10	MK	06.07.2020

metraTec GmbH

Niels-Bohr-Str. 5  
39106 Magdeburg  
Germany

Tel.: +49 (0)391 251906-00

Fax: +49 (0)391 251906-01

Email: <support@metratec.com>

Web: <http://www.metratec.com>

Copyright © 2009-2020 metraTec GmbH

The content of this document is subject to change without prior notice. Copying is permitted for internal use only or with written permission by metraTec. metraTec is a registered trademark of metraTec GmbH. All other trademarks are the property of their respective owners.