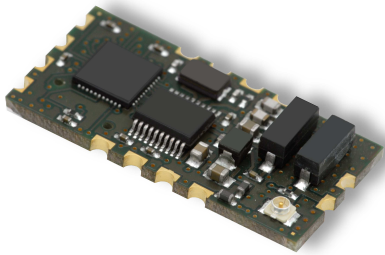


# ISO 15693 Protocol Guide

for metraTec RFID Readers and Modules



Date: May 2016

Version: 3.6

For Firmware Version: 3.5

Customer Edition

## Table of Contents

Introduction .....	5
1. Further Documents .....	5
Typographic Conventions .....	6
1. Communication Principle .....	8
1.1. Helpful Tools .....	8
2. Reader Control Instructions .....	10
2.1. Reset ( <b>RST</b> ) .....	11
2.2. Revision ( <b>REV</b> ) .....	11
2.3. Read Serial Number ( <b>RSN</b> ) .....	12
2.4. Read Hardware Revision ( <b>RHR</b> ) .....	12
2.5. Read Firmware ( <b>RFW</b> ) .....	13
2.6. Read Bootloader Revision ( <b>RBL</b> ) .....	13
2.7. Read Hardware ( <b>RHW</b> ) .....	14
2.8. Standby ( <b>STB</b> ) .....	14
2.9. Wake Up ( <b>WAK</b> ) .....	15
2.10. Read Input Pin ( <b>RIP</b> ) .....	15
2.11. Write Output Pin ( <b>WOP</b> ) .....	16
2.12. Set Antenna Port ( <b>SAP</b> ) .....	17
2.12.1. Activate specific Antenna .....	18
2.12.2. Automatic switching mode .....	18
2.13. Start Up Commands ( <b>SUC</b> ) .....	19
2.13.1. Reset Command Sequence .....	19
2.13.2. Set Command Sequence .....	19
2.13.3. Enable (ON) .....	20
2.13.4. Disable (OFF) .....	20
2.14. Read Start Up Commands ( <b>RSC</b> ) .....	20
2.15. Mode ( <b>MOD</b> ) .....	21
2.16. Set RF Interface ( <b>SRI</b> ) .....	22
2.16.1. RX1 input .....	22

2.16.2. RX2 input .....	22
2.16.3. Single Subcarrier .....	23
2.16.4. Double Subcarrier .....	23
2.16.5. Disable RF .....	24
2.16.6. Disable for some Time .....	24
2.17. Set Timings ( <b>STT</b> ) .....	25
2.17.1. Set Timings T1MAX, T1, T1MIN, T2, T3 and TWMAX .....	25
2.17.2. Set Timing TWMIN .....	26
2.17.3. Set Timing TWWND .....	26
2.18. Cyclic Redundancy Check On ( <b>CON</b> ) .....	27
2.19. Cyclic Redundancy Check Off ( <b>COF</b> ) .....	28
2.20. End Of Frame Mode ( <b>EOF</b> ) .....	28
2.21. No End of Frame Mode ( <b>NEF</b> ) .....	29
2.22. Verbosity Level ( <b>VBL</b> ) .....	30
2.23. Settings ( <b>SET</b> ) .....	30
2.23.1. Power Level (PWR) .....	30
2.23.2. High on Tag (HOT) .....	31
2.23.3. Configure Input High Commands (IHC) .....	31
2.23.4. Set Input High Commands (IHC) .....	32
2.24. Status ( <b>STA</b> ) .....	33
2.25. Read Subversion ( <b>RSV</b> ) .....	33
3. Tag Manipulation Instructions .....	35
3.1. Inventory ( <b>INV</b> ) .....	35
3.1.1. Simple Inventory .....	35
3.1.2. Application family identifier (AFI) .....	36
3.1.3. Single Slot Inventory (SSL) .....	36
3.1.4. Only New Tag (ONT) .....	36
3.1.5. Masking (MSK) .....	37
3.1.6. Single Subcarrier (deprecated) .....	37
3.1.7. Double Subcarrier (deprecated) .....	37

3.2. Reading Request ( <b>REQ</b> ) .....	38
3.3. Direct Reading Request ( <b>DRQ</b> ) .....	42
3.4. Writing Request ( <b>WRQ</b> ) .....	43
3.5. Direct Writing Request ( <b>DWQ</b> ) .....	44
3.6. Continuous Repeat Prefix ( <b>CNR</b> ) .....	44
3.7. Break At Read Postfix ( <b>BAR</b> ) .....	45
3.8. Break ( <b>BRK</b> ) .....	46
4. Error Codes .....	47
A. Quick Start Guide and Examples .....	50
A.1. Typical Reader Initialization Sequence .....	50
A.2. Reading the Tag ID of a tag .....	50
A.3. Reading Tag IDs continuously .....	51
A.4. Example for writing and reading to and from ISO 15693 tags .....	52
A.5. Configuring reader to automatically start reading tag IDs when powered .....	53
B. CRC Calculation .....	54

## Introduction

This document describes the metraTec firmware protocol for all metraTec RFID readers that work with RFID tags according to ISO 15693. This includes the DeskID ISO, UM15, QuasarMX, Dwarf15, QuasarLR and the QR15 OEM module. This guide does not cover the protocol of our older QuasarMR1 reader. Older firmware versions might only support a subset of the commands described here. However, great care has been taken to assure backwards compatibility as much as possible. A description of the other protocols can be found on the website, at (<http://www.metratec.com> → Support → Downloads → Documents).

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.

The reader firmware offers an ASCII based programming interface. The instructions are identified by an easy to remember, three character string usually followed by mandatory parameters and/or optional parameters. The response format depends on the type and result of an instruction.

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 4, *Error Codes*.

## 1. 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. Communication Principle

The communication between the reader and the host system is based on ASCII strings. Each string is terminated with a carriage-return (0x0D), *not* the null byte, and will be transmitted with the most significant byte first.

The communication from the reader to the host system (i.e. the response) is the same as above but in most cases the response from the reader comprises more than one line.



### Note

The answer (not every line individually) may be terminated by a line-feed (0x0A) if the **EOF** command was called.

General syntax:

```
{ Instruction } [ <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. Some instructions may be specified with parameters, which are separated by a space (0x20).

```
INV SSL<CR>
```

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

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

### 1.1. 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 next chapter.

Command	Name	Description
<b>RST</b>	Reset	Resets the reader
<b>REV</b>	Revision	Returns information on reader (deprecated)
<b>RSN</b>	Read Serial Number	Returns the Serial Number of the reader
<b>RHR</b>	Read Hardware Revision	Returns Hardware Revision of the reader (deprecated)
<b>RFW</b>	Read Firmware	Returns firmware name and version
<b>RBL</b>	Read Bootloader Revision	Returns bootloader name and version
<b>RHW</b>	Read Hardware	Returns hardware name and version
<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>SUC</b>	Start Up Commands	A set of commands that are executed automatically on <b>RST</b> or power up
<b>RSC</b>	Read Start Up Commands	Read the set of commands that are executed automatically on <b>RST</b> or power up
<b>MOD</b>	Mode	Sets the ISO norm to use
<b>SRI</b>	Set RF Interface	Configures RF interface and turns RF power on and off
<b>STT</b>	Set Timings	Allows to set all important timings (for ISO 15693), not supported by the QuasarLR
<b>CON</b>	Cyclic Redundancy Check On	Turns on CRC checking of computer / reader communications
<b>COF</b>	Cyclic Redundancy Check Off	Turns off CRC checking of host / reader communications
<b>EOF</b>	End Of Frame Mode	Turns on the End of Frame delimiter <LF>
<b>NEF</b>	No End of Frame Mode	Turns off the End of Frame delimiter <LF>
<b>VBL</b>	Verbosity Level	Sets the amount of details the reader communicates to the user
<b>SET</b>	Settings	Prefix that allows configuring some reader settings
<b>STA</b>	Status	QuasarLR specific command to read status information
<b>RSV</b>	Read Subversion	QuasarLR specific command to read additional version information

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 **RST** command and receiving the answer OK! the reader will behave 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.

The startup process (from the time the OK! is received until new commands to the reader are accepted) takes about 200ms. The QuasarLR starts accepting commands after about 50ms but won't answer before 1.7 seconds have passed.

### 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. Revision (REV)

This command is deprecated since firmware version 0300 / hardware version 0200 and for all QuasarLR. Though the firmware still supports it for legacy reasons future versions might not. For reading the firmware name (which is identical to the product name) and version use **RFW**, for hardware name and version use **RHW**.

### Instruction

**REV** <CR>

### Examples

```
REV<CR>
```

*Example 4. Read reader information (deprecated)*

### Return Values in Case of Success

PRODUCT\_NAME[16 bytes for QuasarLR, 15 bytes any other]  
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>"

## 2.3. Read Serial Number (RSN)

The RSN command returns the serial number of the reader.

### Instruction

**RSN** <CR>

### Examples

```
RSN<CR>
```

*Example 5. 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>"

## 2.4. Read Hardware Revision (RHR)

This command is deprecated since firmware version 0300 / hardware version 0200 and for all QuasarLR. Though the firmware still supports it for legacy reasons future versions might not. Use **RHW** for Read Hardware instead.

### Instruction

**RHR** <CR>

### Examples

```
RHR<CR>
```

*Example 6. Read hardware revision of reader (deprecated)*

## Return Values in Case of Success

`MMSS<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>"`

## 2.5. 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 7. 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>"`

## 2.6. Read Bootloader Revision (RBL)

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

### 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

*"UCO <CR>", "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 (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 9. Read hardware name and revision from reader*

## Return Values in Case of Success

*PRODUCT\_NAME HW\_arch[4bytes]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>"*

## 2.8. Standby (STB)

The standby command sets the reader into power save mode. The RF power is turned off. This means that all tags that might be in the field will also be powered down. If successful it returns GN8 ("Good Night"). The reader will not accept any commands except reset (RST) until a Wake Up Command (WAK) 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 10. 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>"

## 2.9. 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 11. Wake reader from standby mode

## Return Values in Case of Success

```
GMO <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. 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 changed to being an input pin. Please make sure that the hardware connected to the pin is meant to be operated this way before calling **RIP** as this can destroy the hardware.

## 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 12. 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>"

## 2.11. 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 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 respec-



tively. 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	Hexadecimal Integer ( $0_{16} \leq x \leq 7_{16}$ )	Number of pin to write
Pin Setting	Enumeration (HI or LOW)	New state of pin

## Examples

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

*Example 13. Set output pin 0 to high state*

## Return Values in Case of Success

OK! <CR>

## 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>"

## 2.12. Set Antenna Port (SAP)

This command is used to set all 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 and UM15 don't have 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 all 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) and using other outputs for something else it might be better to switch the multiplexer using the **WOP** command.

### 2.12.1. Activate specific Antenna

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> {Antenna Port} <CR>
```

#### Parameters

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

#### Examples

```
SAP 10<CR>
```

Example 14. Set 16 port multiplexer to activate antenna 10

### 2.12.2. 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 (**INV** or **REQ**-family commands).

#### Instruction

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

#### Parameters

Name	Type	Description
No. Antennas	Decimal Integer ( $1 \leq x \leq 16$ )	Number of antennas connected

#### Examples

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

Example 15. Set the reader to automatically switch between the first five antennas (antennas 0-4) with every reading instruction

## 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>"

## 2.13. Start Up Commands (SUC)

Any time the reader is reset - either by using the **RST** command or by toggling the power - the reader will set all parameters to their default settings. In case you want your setup to be retained 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. These commands are persistently stored in EEPROM or FLASH memory. Upon start up of the device the commands are loaded and executed as if they had been sent to the reader at that time. The only difference is that the responses to the commands are suppressed. Due to this, please make sure to check the spelling of the commands you are setting as any error messages are also suppressed.

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 show results. As usual, continuously executing commands can be terminated by the **BRK** command.

### 2.13.1. Reset Command Sequence

**SUC** without further parameters sets an empty string as the command sequence which resets any previous settings.

#### Instruction

**SUC** <CR>

#### Examples

```
SUC<CR>
```

*Example 16. SUC instruction for resetting the command sequence previously defined*

### 2.13.2. 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 17. SUC instruction for automatic start of continuous ID reading

### 2.13.3. Enable (ON)

**SUC** followed by the keyword **ON** will reenables startup command execution previously disabled.

#### Instruction

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

## Examples

```
SUC ON<CR>
```

Example 18. SUC instruction for reenabling startup commands

### 2.13.4. Disable (OFF)

**SUC** followed by the keyword **OFF** will disable startup command execution.

#### Instruction

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

## Examples

```
SUC OFF<CR>
```

Example 19. 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>"

## 2.14. Read Start Up Commands (RSC)

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!.

### Instruction

**RSC** <CR>

### Examples

```
RSC<CR>
```

*Example 20. 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>"

## 2.15. Mode (MOD)

The mode command selects the ISO anti-collision and transmission protocol to be used for tag communication. Default value is ISO 15693-3 and currently no other protocol is supported by the standard firmware. ISO 14443-A / B though listed is currently not supported. Devices with firmware revision >= 03.00 may get this support later. The command MOD 156 is supported but is no longer necessary as this is the default (and only) setting and current firmware versions no longer require calling it before reading tags.

### Instruction

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

### Parameters

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

### Examples

```
MOD 156<CR>
```

*Example 21. 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>"

## 2.16. Set RF Interface (SRI)



### Note

Before you can start reading tags you will need to switch on the RF field. The Set RF Interface Command lets you do this - it is thus mandatory before being able to communicate with tags.

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). A special TIM parameter allows switching the RF off for a specific amount of time to depower tags in the field and then automatically return to the previous RF settings.

### 2.16.1. 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 22. Set reader to use the first receiving port (which is the default)*

### 2.16.2. RX2 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** RX2 command allows selecting the second port.

## Instruction

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

## Examples

```
SRI RX2<CR>
```

*Example 23. Set reader to use the second receiving port*

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

### 2.16.4. 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 25. Activate RF for interacting with tags expecting double subcarrier, 100% ASK modulation (unusual, please check tag IC datasheet)

### 2.16.5. 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 26. Deactivate RF manually

### 2.16.6. 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.

#### Instruction

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

#### Parameters

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

## Examples

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

Example 27. Deactivate RF for 100 ms

## 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. Set Timings (STT)

The Set Timings Command is used to write parameters used to control request or inventory timings. This command is not supported by QuasarLR.



### Note

This command should only be used with deeper knowledge of the 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. Only the basic parameters and their meaning are defined here. 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.

### 2.17.1. Set Timings T1MAX, T1, T1MIN, T2, T3 and TWMAX

**T1MAX:** 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 400µs. Maximum value is 38600µs.

**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 310µs. Maximum value is 38600µs. Values of 350µs are fine for some tags, this will give better results, especially in noisy environments.

**T2:** 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 400µs. Maximum value is 38600µs.

**T3:** 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 400µs. Maximum value is 38600µs. A T3 value that is set too high has no negative side effects except reducing speed.



### Note

The T3 value 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.

**TWMAX:** This is a timer for writing requests. With option flag 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 21000µs. A shorter time might be ok for your tag (or larger time needed). Maximum value is 38600µs.

### Instruction

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

## Parameters

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

## Examples

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

*Example 28. Set T1MAX to 400 $\mu$ s (the default value)*

### 2.17.2. 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 = 324,5 $\mu$ s). 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 29. Set TWMIN to 4400 ticks (the default value)*

### 2.17.3. 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 (=9,44 $\mu$ s)). 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 30. 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>", "BOD <CR>", "BOF <CR>", "CCE <CR>", "CRT <CR>", "SRT <CR>", "UER[<SPACE> {Two Digit Hex Code}] <CR>", "UPA <CR>" or "URE <CR>"

## 2.18. Cyclic Redundancy Check On (CON)

This command turns on the Cyclic Redundancy Check (CRC) of the computer to reader communication. This is used to detect transmission errors between the reader and the computer. 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.



### Note

This has nothing to do with the reader to tag communication CRC mandated by the ISO 15693.

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 answers from the reader will also be extended accordingly. The CRC uses the 0x8408 polynomial, starting value is 0xFFFF. The **CON** command will work with or without the CRC.

If successful the command returns OK! plus the according CRC of "OK! ".

Appendix B, *CRC Calculation* shows a function in C/C++ to calculate the correct CRC16.

## Instruction

**CON** <CR>

## Examples

```
CON<CR>
```

Example 31. 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>"

## 2.19. Cyclic Redundancy Check Off (COF)

This command turns off the Cyclic Redundancy Check (CRC) of the host to reader communication. This is the default setting. In CRC mode this command will only work with the correct CRC (4F5E).

### Instruction

**COF** <CR>

### Examples

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

*Example 32. 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>"

## 2.20. End Of Frame Mode (EOF)

This command turns on the End of Frame delimiter. This means that after every complete reader answer to a request the last <CR> will be followed by an additional line feed (<LF>, 0x0A). This allows the user to build a simpler parser since it is clear when not to expect any further message from the reader.

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



## Note

Please keep in mind that asynchronous errors that reset the reader 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.

### Instruction

**EOF** <CR>

### Examples

```
EOF<CR>
```

*Example 33. Turn on End of Frame delimiter <LF>*

### 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>"

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

This command will turn off the End of Frame delimiter. The answer to the command will already not include it.

### Instruction

**NEF** <CR>

### Examples

```
NEF<CR>
```

*Example 34. 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>"

## 2.22. 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.

### Instruction

**VBL** <SPACE> {Mode} <CR>

### Parameters

Name	Type
Mode	Decimal Integer ( $0 \leq x \leq 2$ )

### Examples

```
VBL 1<CR>
```

Example 35. Set verbosity level to 1 (default value)

### Return Values in Case of Success

OK! <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>"

## 2.23. Settings (SET)

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

### 2.23.1. Power Level (PWR)

The QuasarLR allows different output power levels to match antenna size, tag size or tag position. The power level is given in milliwatt (mW). The minimum value is 500, the maximum is 4000 with steps of 250.

The second generation ISO 15693 devices with hardware revision  $\geq 02.00$  (DeskID ISO, UM15, Dwarf15, QR15 and QuasarMX) allow setting power values of 100 or 200 (mW).

### Instruction

**SET** <SPACE> PWR <SPACE> { 100 | 200 | 500 | 750 | 1000 | 1250 | 1500 | 1750 | 2000 | 2250 | 2500 | 2750 | 3000 | 3250 | 3500 | 3750 | 4000 } <CR>

## Parameters

Name	Type
Power level	Enumeration (100, 200, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500, 2750, 3000, 3250, 3500, 3750 or 4000)

## Examples

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

Example 36. Set power level to 100mW (e.g. for 2nd generation DeskID\_ISO)

### 2.23.2. High on Tag (HOT)

This parameter is only usable on readers that have output pins. It makes the reader set an output pin to high state and reset it to low after a specified amount of time if a valid tag communication has taken place. 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. 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 and QuasarLR. The time set is in ms.



#### 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.

## 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 100 50<CR>
```

Example 37. Set the reader to turn the output on for 100ms and then off for 50ms in case a tag is found

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

This parameter is only usable on readers that have input pins (Dwarf15, QR15, QuasarLR and QuasarMX). It allows defining a set of commands that is to be executed on a falling edge of

an input pin - e.g. when a light barrier triggers. The pin that is used for this can be set as part of the command with pins 0 and 1 supported. Besides the pin that is being monitored, the command expects a second parameter. This can either be a flag (turning the behaviour ON, OFF or with SHW showing the command set being used) or it expects the command set to be used with individual commands separated by ';'.



## 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 **SET IHC** the direction the pin is being used in is changed to being an input pin. Please make sure that the hardware connected to the pin is compatible to the pin's maximum limits in input mode before calling **SET IHC** as this can otherwise destroy the reader or connected hardware.



## Note

Please also note that using **WOP** on the pin sets the pin to output mode and deactivates IHC mode.

## Instruction

```
SET <SPACE> IHC <SPACE> {Pin} <SPACE> { OFF | ON | SHW } <CR>
```

## Parameters

Name	Type
Pin	Hexadecimal Integer ( $0_{16} \leq x \leq 1_{16}$ )
Flag	Enumeration (OFF, ON or SHW)

## Examples

```
SET IHC 1 ON<CR>
```

Example 38. Enable the command sequence to be carried out upon input pin 1 going high

```
SET IHC 0 SHW<CR>
```

Example 39. Show the command sequence to be carried out upon input pin 0 going high

## 2.23.4. Set Input High Commands (IHC)

### Instruction

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

### Parameters

Name	Type
Pin	Hexadecimal Integer ( $0_{16} \leq x \leq 1_{16}$ )



## Examples

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

Example 40. Set the reader to search for tags once input pin 1 goes high

## Return Values in Case of Success

"OK! <CR>" or "Commands<CR>"

## Return Values in Case of Failure

"UPA <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>"

## 2.24. Status (STA)

This command is only usable on QuasarLR and gives some status values that can be useful for debugging purposes. Any other ISO 15693 reader will answer NOS

### Instruction

**STA** <CR>

### Examples

```
STA<CR>
```

Example 41. Read status information from QuasarLR

## 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>"

## 2.25. Read Subversion (RSV)

Reads the version of underlying modules. Used for support only (only QuasarLR).

### Instruction

**RSV** <CR>

## Examples

```
RSV<CR>
```

*Example 42. Read subversion information from QuasarLR*

## Return Values in Case of Success

```
X.Y.Z<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>"

## Chapter 3. Tag Manipulation Instructions

The difference between Reader Instructions and Tag Instructions is that with the latter the target of the instruction is 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. Among these commands the **CNR** command has a special role, since it is not strictly a command by itself but a flag or prefix which changes the way the command following **CNR** is interpreted by the firmware of the reader.

Command	Name	Description
<b>INV</b>	Inventory	This command automatically searches for all tags within reading range and returns their ID
<b>REQ</b>	Reading Request	Command to send ISO 15693 command strings to the tag with reading timeouts
<b>DRQ</b>	Direct Reading Request	Direct version of <b>REQ</b> using reading timeouts
<b>WRQ</b>	Writing Request	Version of the <b>REQ</b> command using writing timeouts
<b>DWQ</b>	Direct Writing Request	Direct version of the <b>WRQ</b> command
<b>CNR</b>	Continuous Repeat Prefix	A command prefix to repeat a command indefinitely.
<b>BAR</b>	Break At Read Postfix	A command postfix to automatically stop <b>CNR</b> mode once a tag is read
<b>BRK</b>	Break	Command to end <b>CNR</b> mode

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> 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).

#### 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 43. Find all tags in the field

### 3.1.2. Application family identifier (AFI)

Setting this optional parameter will lead to tags with the corresponding Application Family Identifier answering the INV command only - 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 44. Find all tags with application family 50 (medical)

### 3.1.3. Single Slot Inventory (SSL)

Single Slot: If one 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 will get a **CLD** (Collision Detected) message from all readers except the QuasarLR. The QuasarLR just finds no tag at all (IVF 00).

#### Instruction

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

## Examples

```
INV SSL<CR>
```

Example 45. Quickly find single tag in field

### 3.1.4. Only New Tag (ONT)

The **ONT** parameter makes the reader find each tag only once as long as it stays powered within the RF field of the reader. If the tag is depowered (e.g. by removing it from the field or by turning the field off and back on) the tag will answer again. In conjunction with the **CNR** prefix this allows 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.

## Instruction

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

## Examples

```
INV ONT<CR>
```

Example 46. 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 XXXXXXXXXXXXX345. **MSK** is not supported by the QuasarLR.

## Instruction

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

## Parameters

Name	Type
Mask to Set	Hexadecimal String

## Examples

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

Example 47. Find all tags whose IDs end with 123

### 3.1.6. Single Subcarrier (deprecated)

In some older versions of the firmware it was possible to set the number of subcarriers as part of the **INV** command. This is deprecated and might not be supported in later firmware versions.

## Instruction

**INV** <SPACE> SS <CR>

## Examples

```
INV SS<CR>
```

Example 48. Find all tags in the field using a single subcarrier

### 3.1.7. Double Subcarrier (deprecated)

In some older versions of the firmware it was possible to set the number of subcarriers as part of the **INV** command. This is deprecated and might not be supported in later firmware versions.

## Instruction

**INV** <SPACE> DS <CR>

## Examples

```
INV DS<CR>
```

Example 49. Find all tags in the field using double subcarrier mode

### 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)

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 but in many cases the reader manufacturer will need to supply custom firmware in cases where custom, tag IC specific commands are needed for special functions.

All metraTec readers for ISO 15693 avoid this systematic drawback by allowing direct communication between the host and the tag IC. This functionality is supplied by the general request commands (**REQ**, **DRQ**, **WRQ**, **DWQ**). They allow sending a sequence of hexadecimal digits to the tag.

The reason there are four versions of this command is due to two effects. For one there is the ISO 15693 specification for timeouts which differs between commands that read data from a tag and those that change / write data on the tag. The **REQ** and **DRQ** commands send the data using the reading timeouts, the **WRQ** and **DWQ** commands use the writing timeouts.

The second effect that leads to the doubling of the commands 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. 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 requirements. Basically, with the "direct" commands the tag is served exactly the string 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 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.

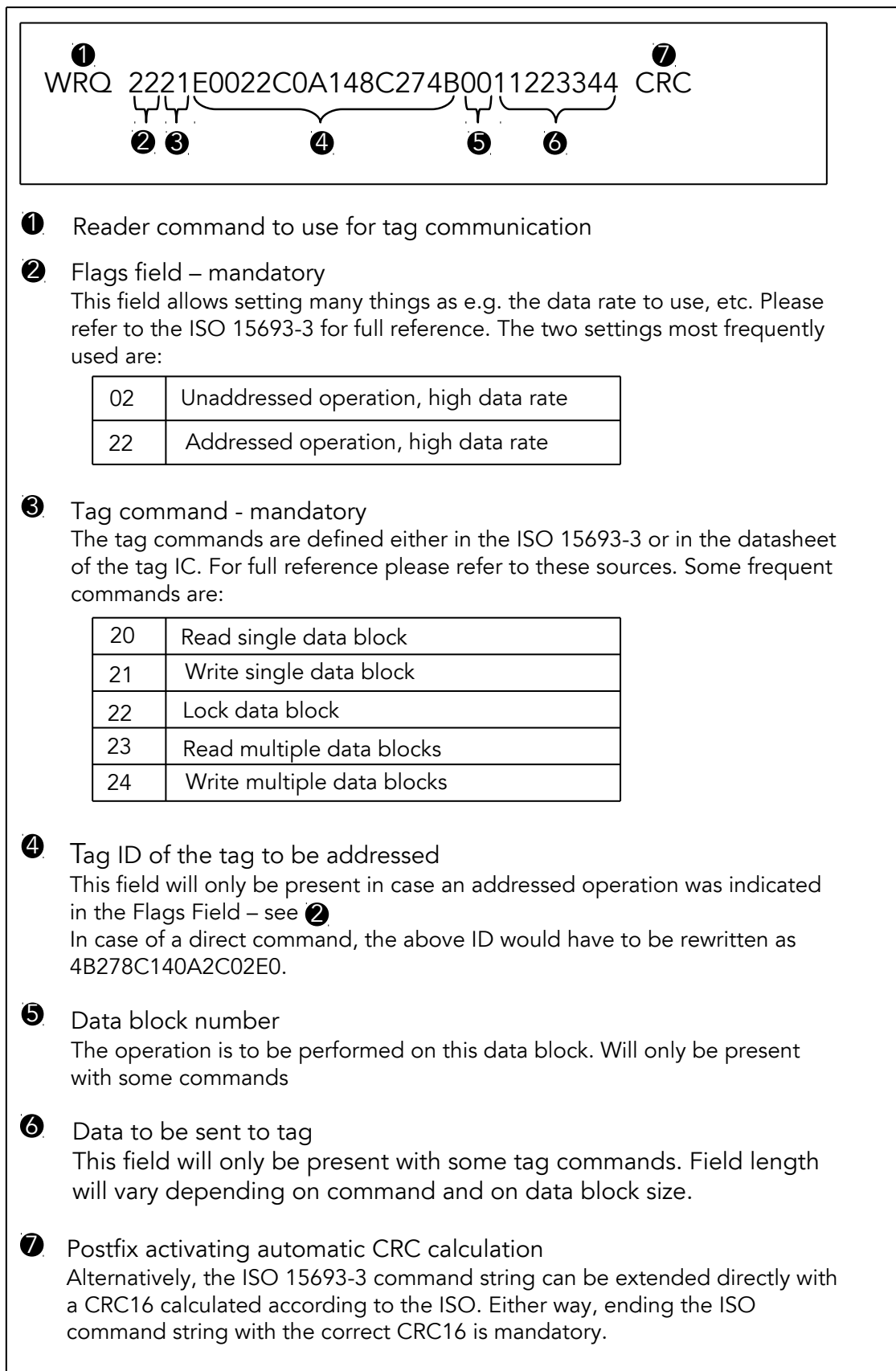


Figure 1. A simplified overview of ISO 15693-3 command syntax

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 having the firmware calculate the correct CRC16 automatically. In case the firmware is to calculate the CRC16 automatically this is indicated using the postfix CRC.



### 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 which is used to secure correct data transmission between the reader and the host computer and which is optional.



### 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 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 always 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. 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 you use an unaddressed command you need to be sure that there is not more than one transponder in the field as the tag answers will otherwise lead to the collisions reported with the tag response. 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.

## Instruction

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

## Parameters

Name	Type
ISO Command	Hexadecimal String

## Examples

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

Example 50. Read block 03 of a single tag in the field without addressing it

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

Example 51. Read block 03 of a tag with the ID E0022C0A148C274B

## Return Values in Case of Success

TDT <CR>

## Return Values in Case of Failure

"TNR <CR>", "RNW <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. Direct Reading Request (DRQ)

This is the "direct" version of the **REQ** command meaning that the ISO command is not parsed and sent directly to the tag. Please make sure that the order of the bytes in any tag ID being used for addressing is in the way the ISO 15693 specifies it. This command uses the reading timeouts and will not work reliably in cases where data is to be changed on the tag. 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 52. Directly read block 03 of a single tag in the field without addressing it

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

Example 53. Directly read block 03 of a tag with the ID E0022C0A148C274B

### Return Values in Case of Success

TDT <CR>

### Return Values in Case of Failure

"TNR <CR>", "RNW <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. 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. 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 54. Write 12345678 to block 03 of a tag without addressing it

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

Example 55. Write 12345678 to block 03 of a tag with the ID E0022C0A148C274B

### Return Values in Case of Success

TDT <CR>

### Return Values in Case of Failure

"TNR <CR>", "RNW <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 not parsed and 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. For further details 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 56. Directly write 12345678 to block 03 of a tag without addressing it

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

Example 57. Write 12345678 to block 03 of a tag with the ID E0022C0A148C274B

#### Return Values in Case of Success

TDT <CR>

#### Return Values in Case of Failure

"TNR <CR>", "RNW <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>"

### 3.6. Continuous Repeat Prefix (CNR)

Both the **INV** command and the **REQ** family of commands can be written after the **CNR** prefix. This will lead to the respective command being repeated indefinitely or until either the **BRK** command is sent, the **RST** command is sent or, with **BAR** appended, until a tag is found. This is a very powerful mechanism for unassisted operations where the reader is initialized at the beginning (e.g. via **SUC**) 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 58. Continuously search for single tags in the field

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

Example 59. Continuously try reading block 03 from any (single) tag in the field

## Return Values in Case of Success

Command Responses

## 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>"

## 3.7. Break At Read Postfix (BAR)

In some situations the reader is set to read continuously, expecting only rare reading events (e.g. an access application via tag). In the case of finding a tag, however, some operation has to be performed with the tag so that the continuous operation has to be interrupted. This is where the **BAR** postfix comes in. Any command that uses the **CNR** command prefix to enter continuous scanning mode can be automatically ended once a tag is found using this postfix.



### Note

In case the command used can generate several answer lines (as e.g. **INV** that generates a whole inventory of tag IDs with each command execution) you can actually get several answers even though **CNR** mode is terminated. Basically, the last command is completely executed.

## Instruction

**BAR** <CR>

## Examples

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

Example 60. Continuously search for single tags in the field automatically stopping when a tag is found

```
CNR REQ 022003 BAR<CR>
```

Example 61. Continuously try reading block 03 from any (single) tag in the field automatically stopping when a tag is found

### Return Values in Case of Success

BRA <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>"

## 3.8. Break (BRK)

If the reader is in **CNR** mode you can stop it by sending **BRK**.

### Instruction

**BRK** <CR>

### Examples

```
BRK<CR>
```

Example 62. 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>"

## Chapter 4. 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. On a QuasarLR this might also be an internal CRC error.
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	Collision detected, a collision has been detected during tag communication. Not supported by QuasarLR. 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:

Error Code	Name	Description
		<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> </ul> <p>With the QuasarLR 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.
RNW	Registers Not Written	The reader's configuration registers are not configured. Common error source: <ul style="list-style-type: none"> <li>• No configuration register settings were entered since the last power up or reset but a tag operation attempted, SRI command required first</li> </ul> <p>This error does not occur on QuasarLR. SRI needs to be used after every restart</p>
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



Error Code	Name	Description
		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 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 2.16, "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 63, "Inventory answer if no tag has been found". If two tags are found the answer could look like in Example 64, "Inventory answer if two tags have been found".

```
« IVF 00<CR>
```

*Example 63. Inventory answer if no tag has been found*

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

Example 64. 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 (80 tags per second for QuasarLR).

Instruction:

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

Possible responses:

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

Example 65. Answer to **INV SSL** if there is no tag

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

Example 66. Answer to **INV SSL** if there is exactly one tag

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

Example 67. 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 68. 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 69. 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  * This function calculates a CRC16 over a unsigned char array
3  * with LSB first.
4  *
5  * @param DataBuf Pointer to data to calculate CRC16 for.
6  * @param SizeOfDataBuf Length of the data buffer (DataBuf)
7  * @param Polynom Value of the generator polynom.
8  *           0x8408 is recommended.
9  * @param Initial_Value Initial value of CRC16.
10 *           0xFFFF is recommended for
11 *           host to reader communication.
12 * @return Calculated CRC16
13 */
14 unsigned short GetCrc(unsigned char *DataBuf,
15                      unsigned char SizeOfDataBuf,
16                      unsigned short Polynom,
17                      unsigned short Initial_Value)
18 {
19     unsigned short Crcl6 = Initial_Value;
20     unsigned char Byte_Counter, Bit_Counter;
21
22     for (Byte_Counter = 0;
23         Byte_Counter < SizeOfDataBuf;
24         Byte_Counter++)
25     {
26         Crcl6 ^= DataBuf[Byte_Counter];
27         for (Bit_Counter = 0; Bit_Counter < 8; j++)
28         {
29             if ((Crcl6 & 0x0001) == 0)
30                 Crcl6 >>= 1;
31             else
32                 Crcl6 = (Crcl6>>1)^Polynom;
33         }
34     }
35
36     return (Crcl6);
37 }
```

## Version Control

Version	Change	By	Date
1.0	created	KD	15.07.2009
1.1	RSN command added	KD	27.10.2010
1.2	VBL, CRT added, SRT, WRR removed	KD	12.07.2011
2.0	Changes for Firmware Version 2.0	MK	12.08.2011
2.1	Minor corrections, matches FW version 2.4	MK	24.10.2011
2.2	Reset Timings added, QuasarLR added, SRI completed, Added error codes to table, merged the two tables  Added max values to STT commands. Corrected milliseconds to microseconds in STT description	MK	12.09.2012
2.3	Added TCE error code, changed to firmware 2.5	MK	09.01.2013
2.4	Clarified REV command answer meaning	MK	01.02.2013
2.5	Added SUC, SET IHC, SET HOT	MK	03.05.2013
3.0	Fixed typos, consistent typography	RH	01.08.2013
3.1	Re-Added Example code and corrected an error	MK	07.05.2015
3.2	Added QR15 2.0 / 3.0 Hardware  Re-Added Example code and corrected an error  Added RBL, RFW, RHW (for Hardwareversion >= 02.00 only)  REV and RHR are now deprecated (for Hardwareversion >= 02.00 only)	MK	07.05.2015
3.3	Added DeskID-ISO / 2.0 Hardware	JP	14.08.2015
3.4	Major rework, fixed errors, typos and added some missing content	FS	14.09.2015
3.5	Reworked testing  Changes for release 3.5 (firmware revision)	MK	01.03.2016
3.6	Change at REV description	MK	13.05.2016

metraTec GmbH

Werner-Heisenberg-Str. 1  
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 © 2009-2016 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.