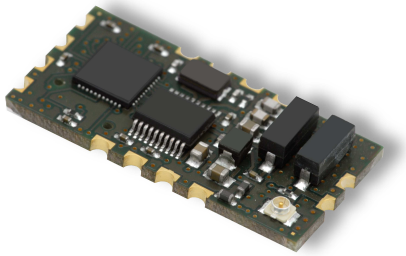
Protocol guide ISO15_V2 Family

of metraTec HF RFID readers and modules for ISO 15693



Date: April 2021

Version: 1.4
For firmware version: 3.14 (QUASAR_MX, DESKID_ISO, DWARF15) / 1.4 (Qr15_V2, DMI15, RR15)
Customer Edition

Table of Contents

Introduction

1. Devices

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

DWARF15, HW Revision 02xx
DESKID_ISO, HW Revision 02xx
QUASAR_MX, HW Revision 02xx
UM15, HW Revision 02xx, Hardware name is DeskID_ISO
DMI15, HW Revision 02xx
QR15_V2
RR15
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 casing 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 QuasarLR shares commands. This device has different hardware or causing it to differ. To a lesser degree this also includes older hardware revisions of DWARF15, DESKID_ISO, QUASAR_MX and QR15 (not QR15_V2, this is part of this document). MetraTec aims at keeping the differences low.

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 You can expect commands to not be removed if possible but to get added parameters or getting outdated by new, stronger or more precise functions.

Metratec aims to keep other RFID reader protocols as close as possible especially regarding general functions like GPIO access, heartbeat etc. Keeping a device compatible with it's former revisions is a priority so this is not always possible. Please always check the devices matching protocol guide.

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 of the tag IC and norms regarding, esp. ISO 15693(-3).

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
`COMMAND`	A command name, i.e. the *literal* name of a command in a metraTec protocol. For instance, **RST** would correspond to the literal characters of a command that could be sent to a metraTec device.
`Literal`	Highlights a *literal value* 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.
`Token`	This convention highlights a replaceable (abstract) *`Token`* that in contrast to a literal token is a placeholder for some other value that must be substituted by the user. The abstract *`Token`* 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.
`{` *`Construct`* `}`	This convention highlights that a *`Construct`* is required. It is most commonly used in syntax descriptions to highlight that a parameter *must* be specified in the position that this construct is used.
`[` *`Construct`* `]`	Highlights that a *`Construct`* is optional. It is most commonly used in syntax descriptions to highlight that a parameter *may* be specified in the position that this construct is used.
`Construct` `...`	Highlights that a *`Construct`* may be repeated many times.
…Name…	The horizontal ellipsis "…" may be used in syntax descriptions to represent arbitrary characters. The arbitrary character field may be given a *`Name`* in order to document it in more detail.
$Alternative_1$ | $Alternative_2$ ... | $Alternative_n$	Highlights that in the position of this construct one of *n* alternatives may be used.
`Literal Line 1` *`Literal Line 2`* *`Literal Line 3`*	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.
» `Command` « `Response`	In examples of a command-response exchange, the literal examples of the ***`Command`*** and ***`Response`*** may be highlighted differently. Otherwise these literal blocks of text are formatted the same as described above.
 **Note** Paragraph	A paragraph set off from the text to highlight noteworthy information.
 **Warning** 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, reffered 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 descripted int 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 metraTerm 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
RST	Reset	Resets the reader
RS485	Control RS485 support	Sets or get the RS485 support status
REV	Revision	Returns information on reader (partially deprecated)
RFW	Read Firmware	Returns firmware name and version
RHW	Read Hardware	Returns hardware name and version
RBL	Read Bootloader Revision	Returns bootloader name and version
RHR	Read Hardware Revision	This command is deprecated.
RSN	Read Serial Number	Reads the Serial Number of the reader
RSV	Read Subversion	Answers `NOS` on this devices with or without any parameter.
EOF	End Of Frame Mode	Controls the use of End of Frame delimiter <LF>
NEF	No End of Frame Mode	This command is deprecated.
CRC	Cyclic Redundancy Check Control	Controls the CRC usage on the Host-Reader communication interface.
CON	Cyclic Redundancy Check On	This command is deprecated.
COF	Cyclic Redundancy Check Off	This command is deprecated.
HBT	Heartbeat	Sends a "HBT" every given number of seconds
SUC	Start Up Commands	Commands to execute on startup
RSC	Read Start Up Commands	This command is deprecated.
EGC	Edge Commands	Controls the edge triggered command feature
CNR	Continuous Repeat Prefix	A meta command used to repeat a tag command until stopped.
BRK	Break	Command to end **CNR** mode
STB	Standby	Sends the reader into standby / sleep mode to save power
WAK	Wake Up	Ends standby / sleep mode
RIP	Read Input Pin	Reads the state of an input pin
WOP	Write Output Pin	Writes the state of an output pin
SAP	Set Antenna Port	Sets the antenna port that is active on a metraTec multiplexer
ECH	Echo	Echos up to 16 characters
VBL	Verbosity Level	Sets the amount of details the reader communicates to the user
SET	Settings	Command that allows configuring some reader settings
SRI	Set RF Interface	Configures RF interface

Command	Name	Description
STT	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 identical to (re-)powered state. The bootloader starts, the basic configuration (UART etc.) is set, the HF power is still turned off and the reader has to be initialized again. This includes Startup commands from **SUC** command being executed.

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 savely 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. Control RS485 support (RS485)

This command allows control of RS485 support. Depending on the device type one IO pin can be used as control pin for a RS485 driver. This is only supported by devices with direct access to UART and IOs (QR15_V2 and Dwarf15). All others will return NOS as they lack IOs or UART access. The used pin is GPIO 5. There is no collision with **SET** HOT or **EGC** features. There is however a collision with **RIP** and **WOP** on the given pin. They should not be used with RS485 activated.

The GPIO5 will be set to high before sending UART data and set back to low when sending is done. This is intended to control a RS485 hardware driver.

Note

RS485 is default disabled on standard devices. It is possible to get devices with default enabled RS485. This will be delivered with a RS485 supporting bootloader instead of the usual UART bootloader. For standard devices **SUC** may be a solution if updates are done manually via UART.

Instruction

RS485 *<SPACE>* { ON | OFF | SHW } *<CR>*

Parameters

Name	Type
RS485 state	Enumeration (ON, OFF or SHW)

Examples

```
RS485 SHW<CR>
```

Example 4. Read RS485 state

```
RS485 ON<CR>
```

Example 5. Enable RS485 support

```
RS485 OFF<CR>
```

Example 6. Disable RS485 support

Return Values in Case of Success

"OK! *<CR>*", "ON *<CR>*" or "OFF *<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. Revision (REV)

This command is 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. The FW name is shortened for compatiblity reasons to 15 bytes.

Instruction

REV *<CR>*

Examples

```
REV<CR>
```

Example 7. Read reader information (deprecated)

Return Values in Case of Success

FIRMWARE_NAME [4bytesHWArch][4bytesFW]<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 Firmware (RFW)

This command returns the name of the firmware - identical to the product name - and the version of the firmware. The firmware name is exactly 16 bytes (ASCII coded charakters). The last 3 Bytes may name a customer specific firmware, coded as Fxx with xx being the variant code.

Instruction

RFW *<CR>*

Examples

```
RFW<CR>
```

Example 8. Read firmware name and revision from reader

Return Values in Case of Success

FIRMWARE_NAME FW_revision[4bytes FW Revision]<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 9. Read hardware name and revision from reader

Return Values in Case of Success

FIRMWARE_NAME [4bytes HW_revision]<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 10. Read bootloader name and version

Return Values in Case of Success

BOOTLOADER_NAME MAYOR_REV[2bytes]MINOR_REV[2bytes]<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 11. 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. It is an array of 16 bytes. The content should be ASCII coded decimal numbers only (0x30 - 0x39, '0' to '9' in ASCII).

Instruction

RSN *<CR>*

Examples

```
RSN<CR>
```

Example 12. 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. Read Subversion (RSV)

Answers NOS on this devices with or without any parameter.

Instruction

RSV *<CR>*

Examples

```
RSV<CR>
```

Example 13. Read subversion information from QuasarLR

Return Values in Case of Success

NOS *<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. 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 more simple 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: Asynchronous errors that reset the reader (like brownout (BOD) or watchdog(SRT)) 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 excecuted after the check of reset causes.

There are 3 parameter, ON, OFF and SHW. ON will enable the EOF mode, OFF will disable the EOF mode and SHW will send the current EOF mode. Using no parameter is identical to **EOF** ON to be compatible with older firmware versions and device types. No parameter is deprecated.

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 14. Turn on End of Frame delimiter <LF>

```
EOF OFF<CR>
```

Example 15. Turn off End of Frame delimiter <LF>

```
EOF SHW<CR>
```

Example 16. Show End of Frame delimiter state answerering '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>"

2.11. 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 17. 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>"

2.12. 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 with a lot 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.

There are 3 parameter, `ON`, `OFF` and `SHW`. `ON` will enable CRC checking / sending, `OFF` disables and `SHW` sends the current state (`ON` or `OFF`).

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 must be a space character which is included in the CRC calculation. All lines from reader to host will also be extended accordingly. This is already true for the answer to **CRC**`ON`.

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. 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.

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** to make a save reset possible.

Instruction

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

Parameters

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

Examples

CRC ON***<CR>***

Example 18. Turn on CRC checking (only with CRC disabled). Answer will have a CRC.

CRC ON B6A8***<CR>***

Example 19. Turn on CRC checking.

CRC OFF FFB1***<CR>***

Example 20. Turn off CRC checking. Answer will have no CRC.

CRC SHW B6A8***<CR>***

Example 21. 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>*"

2.13. Cyclic Redundancy Check On (CON)

This command is deprecated. It is identical to **CRC** ON.

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 819E<CR>
```

Example 22. Turn on reader to computer CRC checking

```
CON<CR>
```

Example 23. Turn on reader to computer CRC checking (only when CRC is off)

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>*"

2.14. Cyclic Redundancy Check Off (COF)

This command is deprecated. Disables the CRC just like **CRC** OFF.

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 *<CR>*

Examples

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

Example 24. Turn off reader to computer CRC checking (please note the 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>*"

2.15. 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 25. **HBT** command and answer*

2.15.1. Get heartbeat parameter state (SHW)

Instruction

HBT [<SPACE> SHW] <CR>

Parameters

Name	Type	Description
SHW	Optional Keyword	The parameter is the SWH keyword. It's use is optional.

2.15.2. 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

2.15.3. 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 ≤ x ≤ 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>*"

2.16. Start Up Commands (SUC)

Any time the reader firmware (re-)starts - by using the **RST** command, enabling the power supply, error or any other reason - the reader will set all parameters to their default settings.

If the user needs 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 nonvolatile 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. 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 truely 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** is not usable (both because it is dangerous at startup and because it is used to reset the startup commands). If the continuous prefix **CNR** is used the command will be executed continuously and will therefor show results as soon as startup is done. As usual, continuously executing commands can be terminated by the **BRK** command.

2.16.1. 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 26. SUC instruction for resetting the command sequence previously defined

2.16.2. 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 27. SUC instruction for automatic start of continuous ID reading

2.16.3. Enable (ON)

SUC followed by the keyword `ON` will reenable 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 28. SUC instruction for reenabling startup commands

2.16.4. Disable (OFF)

SUC followed by the keyword `OFF` will disable startup command execution but leaves the commands in momory 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 therefor not be used frequently.

Instruction

SUC *<SPACE>* OFF *<CR>*

Examples

```
SUC OFF<CR>
```

Example 29. SUC instruction for turning off startup commands

2.16.5. 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 30. SUC instruction for resetting startup commands

2.16.6. 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>* EXE *<CR>*

Examples

```
SUC EXE<CR>
```

Example 31. SUC instruction for executing current startup commands

2.16.7. 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 32. SUC instruction for printing 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>*"

2.17. Read Start Up Commands (RSC)

This command is deprecated. It behaves identical to **SUC SHW**.

Instruction

RSC *<CR>*

Examples

RSC*<CR>*

Example 33. 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>*"

2.18. Edge Commands (EGC)

Note

The **EGC** command is not supported by DeskID_ISO and RR15 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. Supported input pins are 0 and 1 (Input 0/1 on QuasarMX and DMI15, GPIO 0/1 on Dwarf15 and QR15) and for both pins a different set of commands and trigger can be saved.

Trigger 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).

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 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" / "ECH EDGE 1" and "NONE").

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 may collide with other IO usage on the same pin. This should never 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 NONE<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 34. **EGC** command and answer*

Instruction

EGC <SPACE> {Pin} <SPACE> { NONE | FALL | RISE | BOTH } <SPACE> { SHW | EXE | RST | DEL } <SPACE> {…} <CR>

Parameters

Name	Type
Pin	Decimal Integer (0 ≤ x ≤ 1)
Trigger mode	Enumeration (NONE, FALL, RISE or BOTH)
Controls	Enumeration (SHW, EXE, RST or DEL)

Examples

EGC 0 SRI SS 100;CNR INV<CR>

Example 35. Set Edge command for pin 0

EGC 1 BRK<CR>

Example 36. Set Edge command for pin 1

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

Example 37. Set Edge trigger to both edges for pin 0

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

Example 38. Show current settings for pin 0

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

Example 39. 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>"

2.19. Continuous Repeat Prefix (CNR)

CNR starts **Tag Instructions** in a continuous mode (CNR mode). This will lead to the respective command being repeated until it is either stopped by the **BRK** command or a triggered BRA, the **RST** command or another reason causes a reset. Other commands will be rejected without reply.

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.

CNR gives no answer by itself. The executed command does though as soon it is called. The command is executed instantly once. This is true even if commands are pending in the input buffer. One answer is therefor guaranteed.

Instruction

CNR <SPACE> {…} <CR>

Examples

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

Example 40. Continuously search for single tags in the field

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

Example 41. 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>*"

2.20. Break (BRK)

If the reader is in continuous read mode (**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 itteration is finalized before the **BRK** command is executed so the answer can never interfere the current answer.

Instruction

BRK *<CR>*

Examples

BRK*<CR>*

Example 42. 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>*"

2.21. Standby (STB)

The standby command sets the reader into power save mode. The RF power is turned off. After waking up it will be active or inactive like before going into standby mode. If successful it returns GN8 ("Good Night"). The reader will not accept any commands except **RST** until a **WAK** command is received. Standby has no parameters.

Instruction

STB *<CR>*

Examples

```
STB<CR>
```

Example 43. 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>"

2.22. 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 44. 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>"

2.23. Read Input Pin (RIP)

This command reads the current state of an input pin. It takes one parameter: 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 DMI15 accept 0 and 1 as pin numbers. Dwarf15 and QR15 accept 0 to 7 as input pin numbers. The DeskID ISO / UM15 and RR15 do not have any input pin.

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 of the used pin 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 45. 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>`*"

2.24. Write Output Pin (WOP)

This command sets 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 device still accepts "HI" instead as backwards compatiblity (deprecated).

The possible parameter range depends on the number of output pins the hardware has. QuasarMX and DMI15 accept 0 to 3 as output pin numbers. Dwarf15 and QR15 accept 0 to 7 as output pin numbers. The DeskID ISO and RR15 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 of the pin is changed to output. Please make sure the external hardware connected to the pin will not exceed the pin's maximum limits in output mode before calling **WOP** as this may damage the reader or external device.

Instruction

WOP *<SPACE>* {*WOP_PIN_0_7*} *<SPACE>* { HI | LOW } *<CR>*

Parameters

Name	Type	Description
WOP_PIN_0_7	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 46. Set output pin 0 to high state

Return Values in Case of Success

OK! *<CR>*

Return Values in Case of Failure

“EHX *<CR>*”, “NOR *<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. 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 and RR15 have no outputs, QR15 has no external antenna. They will respond with NOS.

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.

2.25.1. Activate specific Antenna using MAN

For accessing a specific antenna manually - 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 47. Set 16 port multiplexer to activate antenna 10

2.25.2. Activate specific Antenna not using MAN

For accessing a specific antenna manually - the first antenna being antenna 0. 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 48. Set 16 port multiplexer to activate antenna 10

2.25.3. Automatic switching mode

AUT will automatically switch between multiple antennas (e.g. to find all tags in a search area that can only be searched using multiple antennas). The host hast 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. Setting it to zero equals OFF

Examples

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

Example 49. Set the reader to automatically switch between the first five antennas (antennas 0-4) with every reading instruction

2.25.4. Automatic antenna reporting

Controls antenna reporting. If enabled the reader will send an antenna information after all tag information and before the tag number information (IVF XX). The format is ARP X in a seperate line. The information will be send both in manual and automatic mode. If no mode was set the information will still give the current output setting. For bidirectional IOs this will set them to output.

Instruction

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

Parameters

Name	Type
New state	Enumeration (ON or OFF)

Examples

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

Example 50. Enable automatic antenna report

2.25.5. 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. If ARP is missing the reporting is disabled.

Instruction

SAP *<SPACE>* SHW *<CR>*

Examples

```
SAP SHW<CR>
```

Example 51. Get current antenna settings

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>*"

2.26. 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 52. **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>*"

2.27. Verbosity Level (VBL)

This command allows the user to adjust the amount of communication coming from the reader. For 2 the communication is as describes (default value).

For VBL set to one the answer to **INV** is stripped of the number of tags found. On request commands like **REQ** the TDT, NCL and COK lines will be removed. The error variants CLD and CER are still send. This will reduce a successful (and most failed) request(s) to a one line answer with only answer data or error code.

For VBL set to zero a minimun amount of data is sent. All changes from setting 1 stay in place. Additionally for inventory there is no answer at all in case no tag is found. For requests error codes including CLD and CER are surpressed now.

Without a parameter it reports the current value (like SHW).

Instruction

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

Parameters

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

Examples

```
VBL 2<CR>
```

Example 53. 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>*"

2.28. 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. 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 DMI15. 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 ≤ x ≤ 255)
Low time	Decimal Integer (0 ≤ x ≤ 255)

Examples

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

Example 54. 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 55. 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 56. Gets the current SET HOT state

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

Example 57. Disables 'High on tag' mode

2.28.3. Configure Input High Commands (IHC)

Note

This subcommand is depricated 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 ≤ x ≤ 1)

2.28.4. Set Input High Commands (IHC)

Instruction

SET *<SPACE>* IHC *<SPACE>* {*Pin*} *<SPACE>* {…} *<CR>*

Parameters

Name	Type
Pin	Decimal Integer (0 ≤ x ≤ 1)

Examples

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

Example 58. Set the reader to search for tags once input pin 1 goes high

2.28.5. Power Level (PWR)

Note

SET PWR is depricated. 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 59. Set power level to 100mW (e.g. for 2nd generation DeskID_ISO)

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

Example 60. 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 usually the best choice (see following subchapters). The default value is single subcarrier, 100% ASK (also OOK). RF power needs to be enabled before communicating with tags. If not enabled the `NRF` error is given when trying to communicate with tags. **`INV`** will also enable power.

Note

Before interacting with tags the RF needs to be enabled.

2.29.1. RX1 input

The devices 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 an external antenna

is used 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 61. Set reader to use the first receiving port (which is the default)

2.29.2. RX2 input

The **SRI** RX2 command allows selecting the second port.

Instruction

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

Examples

```
SRI RX2<CR>
```

Example 62. 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. This will also enable RF (like **SRI** ON does).

Instruction

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

Parameters

Name	Type
Modulation depth	Enumeration (10 or 100)

Examples

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

Example 63. 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 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. This will also enable RF (like **`SRI ON`** does).

Instruction

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

Parameters

Name	Type
Modulation	Enumeration (`10` or `100`)

Examples

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

Example 64. Activate RF for interacting with tags expecting double subcarrier, 100% ASK modulation (unusual, please check tag IC datasheet)

2.29.5. Enable RF

`ON` parameter will enable the RF field. The modulation type and modulation depth is not changed by this command. It is single subcarrier 100% ASK by default but may differ if it was changed since the start. The output may still be overwritten by power saving SAV

Instruction

`SRI` *`<SPACE>`* `ON` *`<CR>`*

Examples

```
SRI ON<CR>
```

Example 65. Activate RF

2.29.6. Disable RF

`OFF` parameter will disable the RF field and depowers any tags.

Instruction

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

Examples

```
SRI OFF<CR>
```

Example 66. Deactivate RF manually

2.29.7. 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 67. Set T3M to short mode

2.29.8. Disable for some Time

TIM parameter automatically turns off the RF field for a specified amount of time (e.g. to depower a tag). The settings the RF interface has before the TIM kept. 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 ≤ x ≤ 2000)

Examples

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

Example 68. Deactivate RF for 100 ms

2.29.9. Config power level

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

Instruction

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

Parameters

Name	Type
Power value	Enumeration (`100` or `200`)

Examples

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

Example 69. Set RF Transmitter output power to 100 mW

2.29.10. 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 diabled 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" µs before the tag manipulation is executed if TX was deactivated (so in burst mode no time is lost if multiple commands are executed in fast order). For slowing down mode the power down time is max 1000 (not 300k) because a long pause may cause a watchdog error otherwise. 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 70. Set power saving to burst with default timing

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

Example 71. Set power saving to slowed mode with default power up and 300ms power down

2.29.11. 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). 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.

Note

SHW is most likely to get changed in the future as additional settings may need to be shown. Additional parameter will be added at the end.

Instruction

SRI *<SPACE>* SHW *<CR>*

Examples

```
SRI SHW<CR>
```

Example 72. 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 / undesireable consequences resetting the reader will restore the default settings. All values might be subject to internal change of usage (e.g. the length of SOF is added to T1MAX as without that the answer would not be recognized).

All values can use the unit suffixes TCK (to be carrier clock wise) or MS (microseconds, not milliseconds) to have a clear usage. Without suffix the default meaning (which can be milliseconds, ticks or multiple ticks) is used (legacy mode, deprecated). A tick is 1 / fC = 1/13.56 MHz ~= 73.7ns.

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

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 without suffix is set in microseconds. Maximum value is 38600µs.

T1MIN: This is the minimum answer time for any read-alike request (including inventory request) to the beginning of the answer. Answers coming before this time will be ignored. The value without suffix is set in microseconds. Maximum value is 38600µs.

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 without suffix is set in microseconds. Maximum value is 38600µs.

T3MIN: T3 can be used equivalently. This is the minimum time after any request (including inventory reqests) to the beginning of the next request if there was no answer. Commands coming before this time will be delayed. The value without suffix is set in microseconds. Maximum value is 38600µs. A T3MIN value that is set too high has no negative side effects except reducing speed.

Note

The T3MIN value usually differs by communication mode. Needs to be changed for inventory requests on tags using 10% ASK modulation compared to 100% ASK mode! TNRT parameter controls this extra time.

Maximum value is 38600µs.

TNRT: This time is added to T3 in specific conditions setable by SRI T3M. It defaults to the answer time of an inventory answer which is 12 bytes + SoF + EoF + 4192 ticks (T2 minimal value). This equals the requirements if a tag answered but was not recognized. SRI T3M is defaulted to AUTO meaning it is added to communication if the ASK is 10. SRI T3M can be change to force or disable the use of TNRT (so T3 can be controled by the host manually). The value without suffix is set in microseconds.

TWMAX: TW can be used equivalently. 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 without suffix is set in microseconds. 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 (TWMIN + n * TWCYC) and (TWMIN + TWWND + n * TWCYC), n is a natural number >= 0. The value without suffix is set in microseconds.

TWWND: This is the windowsize. It is given in ticks as it is short. In write-alike requests (WRQ, DWQ) the recevier is activated every (TWMIN + n * TWCYC) and deactivated TWWND Ticks later. The value without suffix is set in microseconds.

TWCYC: This is the write alike window cycle time so the time when the window is reenabled. It starts from the first enablign (TWMIN). The default value is 4096 ticks like the ISO15693-3 requires. Set the value to 0 to use no windows at all but receive all the time. The value without suffix is set in microseconds.

Instruction

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

Parameters

Name	Type
Timing Type	Enumeration (`T1MAX`, `T1`, `T1MIN`, `T2`, `T2MIN`, `T3`, `T3MIN`, `TNRT`, `TWMAX`, `TW` or `TWCYC`)
Value	Decimal Integer (0 ≤ x ≤ 38600)

Examples

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

Example 73. Set T1MAX to 400 µs

```
STT TWCYC 4096 TCK<CR>
```

Example 74. Set TWCYC to 4096 ticks

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. Maximum value is 0xFF00. The value without suffix is set in (8 ticks).

Instruction

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

Parameters

Name	Type
Value	Decimal Integer (0 ≤ x ≤ 65280)

Examples

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

Example 75. Set TWMIN to 4400 ticks

```
STT TWMIN 4400 TCK<CR>
```

Example 76. Set TWMIN to 4400 ticks

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 without suffix is set in (8 ticks). Changing this value is discouraged and should only be done in case of real need!

Instruction

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

Parameters

Name	Type
Value	Decimal Integer (0 ≤ x ≤ 255)

Examples

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

Example 77. Set TWWND to 128 ticks

```
STT TWWND 128 TCK<CR>
```

Example 78. Set TWWND to 128 ticks

```
STT TWWND 10 MS<CR>
```

Example 79. Set TWWND to 10 microseconds

2.30.4. Get timings

SHW: This parameter gives all settings back. They are printed with name, value and unit (always TCK). You may use any element as it is printed to not change it. There is no specified order (though it is fix currently). Every timing is given with the most specific name (like T3MIN instead of T3).

Instruction

STT *<SPACE>* SHW *<CR>*

Examples

```
STT SHW<CR>
```

Example 80. Get timings

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>*"

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.

Any request instruction command expects the field to be enabled and a subcarrier setting to be selected. Use **SRI** command to activate and select. The default subcarrier setting is SS 100, the by far most used and usually best working. INV will also enable the field.

Command	Name	Description
INV	Inventory	Inventories tags by unique ID.
REQ	Reading Request	Command to send ISO 15693 command strings to the tag and get its answer.
WRQ	Writing Request	Version of the **REQ** command using writing timeouts
DRQ	Direct Reading Request	Direct address variant of **REQ**
DWQ	Direct Writing Request	Direct address version of the **WRQ** 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 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 81. 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 82. 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 83. 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 automatical 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 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 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 84. 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 XXXXXXXXXXXX345.

Instruction

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

Parameters

Name	Type
Mask to Set	Hexadecimal String

Examples

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

Example 85. 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)

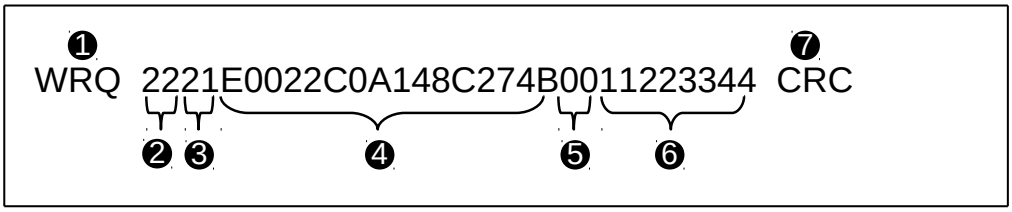
For custom commands (not ISO15693-3 defined) use DRQ and DWQ as they send as given.

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.



❶ Reader command to use for tag communication

❷ Flags field – mandatory
This field allows setting many things as e.g. the data rate to use, etc. Please refer to the ISO 15693-3 for full reference. The two settings most frequently used are:

02	Unaddressed operation, high data rate
22	Addressed operation, high data rate

❸ Tag command - mandatory
The tag commands are defined either in the ISO 15693-3 or in the datasheet of the tag IC. For full reference please refer to these sources. Some frequent commands are:

20	Read single data block
21	Write single data block
22	Lock data block
23	Read multiple data blocks
24	Write multiple data blocks

❹ Tag ID of the tag to be addressed
This field will only be present in case an addressed operation was indicated in the Flags Field – see ❷
In case of a direct command, the above ID would have to be rewritten as 4B278C140A2C02E0.

❺ Data block number
The operation is to be performed on this data block. Will only be present with some commands

❻ Data to be sent to tag
This field will only be present with some tag commands. Field length will vary depending on command and on data block size.

❼ Postfix activating automatic CRC calculation
Alternatively, the ISO 15693-3 command string can be extended directly with a CRC16 calculated according to the ISO. Either way, ending the ISO command string with the correct CRC16 is mandatory.

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 **`CRC`** 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 notices a transmission error they will notify the user that such an error 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 86. Read block 03 of a single tag in the field without addressing it

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

Example 87. 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 88. Write 12345678 to block 03 of a tag without addressing it

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

Example 89. 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 for UID (to invert) but sent directly to the tag. 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 90. Directly read block 03 of a single tag in the field without addressing it

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

Example 91. 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 a variant of the **REQ** command meant to be used for write-alike commands. DWQ does send the data "as is" without parsing to invert UID byte order. 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 92. Directly write 12345678 to block 03 of a tag without addressing it

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

Example 93. 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 '#' and differ in length. Any setting will replace any prior setting (so setting 'P' only will disable 'C', too). Setting of prefixes will be executed even if the command is rejected (UCO, UPA, etc.).

4.1. Command Answer Prefixes ('P')

This command sets a prefix of up to 16 characters which is sent before any answer line. A space character (0x20) is appended to the prefix. The prefix will stay active until it is deleted (using the command without following parameters (##)) or replaced with a new prefix command (including 'C' (counter)). This may be used to detect specific devices or to "tag" specific answers to specific commands. For the later the 'P' may be prefixed with an 'S' for a single shot (only change the answer of this command but no further).

» `#PDEVICE1#INV<CR>`

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

» `INV<CR>`

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

» `#SPDEVICE1#INV<CR>`

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

» `INV<CR>`

« `E200FFA039C992<CR>`
`IVF 001<CR>`

Example 94. 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 at 0 (not every line ending on <CR>, but every com-

plete 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->00, F->0 or FF->00.

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

The format is set by the parameters `D1`, `D2`, `H1`, `H2` to define decimal (D) or hexadecimal (H) and one or two digits, respectively. All are optional. H is default, 2 is default. This means "C" equals "CH2" and "CD" equals "CD2".

It is possible to use 'C' instead for "current" to keep a running counter (may be useful together with a single shot prefix command).

» `#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>`

» `#CC#INV<CR>`

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

» `##INV<CR>`

« `E200FFA039C992<CR>`
`IVF 001<CR>`

Example 95. 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. For the answers, the same prefix order will be used as during the command calls. The use of single shot is possible and will only disable the specified command.

» #PDEVICE1##CD1#INV<CR>

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

» INV<CR>

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

» ##INV<CR>

« E200FFA039C992<CR>
2 IVF 001<CR>

Example 96.

4.4. Single Short ('S')

Single shot is prefixed to 'P'. It allows to single shot a prefix. It will only be printed once with the command given. This is independent of 'C' (this will keep counting if it was part of the command line). The 'S'+'P' may be used together with **SET IHC** to show which input caused the command execution.

» #SPINPUT1##CH1#INV<CR>

« INPUT1 0 E200FFA039C992<CR>
INPUT1 IVF 001<CR>

» 1 INV<CR>

« E200FFA039C992<CR>
IVF 001<CR>

» #SPINPUT1##CC#INV<CR>

« INPUT1 2 E200FFA039C992<CR>
INPUT1 IVF 001<CR>

» 3 INV<CR>

« E200FFA039C992<CR>
IVF 001<CR>

Example 97.

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: • The tag left the field or is too far away • Another device disturbed the communication • A collision between tag answers happened
CLD	Collision Detected	A collision has been detected during tag communication. Common error source: • More than one tag in reader field but single slot inventory set or unaddressed operation requested
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 **WAK** 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: • Other character than '0' to '9' sent
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: • The tag left the field or is too far away

Error Code	Name	Description
		• Another device disturbed the communication
NCM	Not in CNR Mode	BRK was used without a running CNR command.
NOR	Number Out of Range	Common error source: • RIP, WOP: Value is higher than number of I/O-Pins • Length of data or parameter value is not allowed
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) Common error source: Write command returned wrong check (handle). Data might be corrupted.
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 **RST** 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: • Typo in command string • Wrong firmware version
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.

Error Code	Name	Description
UPA	Unknown Parameter	An invalid parameter has been passed to a function. Common error source: • Typo in command string • Given parameter is out of range • Parameter missing (formerly EPX)
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 98, "Inventory answer if no tag has been found". If two tags are found the answer could look like in Example 99, "Inventory answer if two tags have been found".

« `IVF 00<CR>`

Example 98. Inventory answer if no tag has been found

« E0040100078E3BB0<CR>
E0040100078E3BB7<CR>
IVF 02<CR>

Example 99. 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 100. Answer to **INV** SSL if there is no tag*

« E0040100078E3BB0<CR>
IVF 01<CR>

*Example 101. Answer to **INV** SSL if there is exactly one tag*

« CLD<CR>
IVF 00<CR>

*Example 102. 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 103. 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 104. 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 Crc16 = Initial_Value;
21         unsigned char Byte_Counter, Bit_Counter;
22
23         for (Byte_Counter = 0;
24              Byte_Counter < SizeOfDataBuf;
25              Byte_Counter++)
26         {
27                 Crc16 ^= DataBuf[Byte_Counter];
28                 for (Bit_Counter = 0; Bit_Counter < 8; j++)
29                 {
30                         if ((Crc16 & 0x0001) == 0)
31                                 Crc16 >>= 1;
32                         else
33                                 Crc16 = (Crc16>>1)^Polynom;
34                 }
35         }
36
37         return (Crc16);
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
1.2	Added new parameters for STT Updated timing informations in STT Changed FW version to 3.11 / 1.1	MK	21.08.2020
1.3	Internal reworking Differention of device types Changed FW version Minor description changes, more device type specific	MK	26.03.2021
1.4	Clearifications Added SRI ON Corrected Precommands	MK	27.04.2021

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