ISO 15693 Protocol Guide

for metraTec RFID Readers and Modules

Date: September 2012

Document Version: 2.2
For Firmware Version: 2.5

Table of Content

1. Introduction

This document describes the metraTec firmware protocol for all metraTec RFID readers that work with RFID transponders according to ISO 15693. This includes the DeskID ISO, QuasarMX, Dwarf15, QuasarLR and the QR15 OEM module. This guide does not cover the protocol of our older QuasarMR1 reader or older Firmware versions. A description of this protocols can be found on the website, too. (http://www.metratec.com → Support → Downloads → Documents)

The target audience for this document are programmers, who need to communicate with the reader and want to write their own software for this task. This software can be written in any programming language, such as C#, Java, Delphi, Ansi-C or IEC/EN (6)1131-3, e.g. with CoDeSys. An alternative to this low level protocol is to use our free .NET DLL if you use a modern Windows operating system.

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

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

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

All Instructions have Error-Codes which are described in section 5.

1.1. Further Documents

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

2. Communication Principle

The communication between the reader and the host system is based on ASCII strings. Each string is terminated with a carriage-return (0x0D), NOT NULL and will be transmitted with MSB first.

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

Hint: The answer may is finished by line-feed <LF> (0x0A) if the command EOF was called.

General line:

```
Instruction<SPACE>Parameter<Space>Parameter<CR>
```

Example without Parameter:

```
REV<CR>
```

in ANSI C:

char Rev[4] = {'R','E','V',13};

The first values which will be sent is 'R' (52h), followed by 45h, 56h, 0dh. Some instructions may be specified with parameters, which are separated by a space (20h).

Example with Parameter

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

char Inv[8] = {'I','N','V',0x20,'S', 'S', 'L' ,0x0D};

2.1. Helpful Tools

For debugging purpose 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 could be:

- If you communicate via a (real or virtual) COM-Port: a Com-Port Monitor (several free version available in 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, also available on our website.

3. 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 transponder, can be found in the next chapter.

Command	Name	Description
RST	Reset	Resets the reader
REV	Revision	Returns hardware requirement and firmware version
RSN	Read Serial No.	Returns the Serial Number of the reader
RHR	Read Hardware-Revision	Return the hardware revision
STB	Standby	Sends the reader into standby/sleep mode for power saving
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
MOD	Mode	Sets the ISO Norm (14434-B, 15693)
SRI	Set RF-Interface	Sets modulation depth and subcarrier mode or turns RF off
STT	Set Timings	Allows to set all important timings (for Iso15693), not for QuasarLR
CON	CRC on	Turns on CRC checking of computer / reader communication
COF	CRC off	Turns off CRC checking of computer / reader communication
EOF	End of Frame	Turns on the End of Frame delimiter <LF>
NEF	No End of Frame	Turns off the End of Frame delimiter <LF>
VBL	Verbosity Level	Sets the verbosity level
SET PWR	Set Power Level	Sets the RF output power (only QuasarLR)
STA	Status	Gets the Status of the QuasarLR (only QuasarLR)
RSV	Read sub version	Reads the version of underlaying modules. Used for support only. (Only QuasarLR)

Table 1: Overview of reader control instructions

3.1. Reset (RST)

The reset command hard resets the reader. The Reset command has no parameters. After sending the RST command and receiving the OK! The reader will behave like after (re-)powering. The Bootloader starts, the basic configuration (UART etc.) is done, the HF power is turned off and the reader has to be initialized again.

The startup (from OK! to taking commands) takes about 200ms. For QuasarLR the time to take commands is about 50ms but the first answer is coming not before 1.7 seconds.

Instruction:

```
RST<CR>
```

Response, if successful:

```
OK!<CR>
```

Possible Error Response:

```
UPA<CR>
```

3.2. Revision (REV)

The revision command requests the device type and hard- and software revision of the reader. The reader returns its device type and the hardware requirement and software revision of the firmware. The hardware requirement revision is the hardware type needed to use the firmware. It's not identical to the RHR commands hardware revision which may change without a change of the firmware. REV has no parameters and returns no error codes.

Instruction:

```
REV<CR>
```

Response, if successful:

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

QuasarLR: 16 Bytes product name (filled with Spaces) + 4 bytes HW-Revision + 4 Bytes Software-Revision + <CR>

All other: 15 Bytes product name (filled with Spaces) + 4 bytes HW-Revision + 4 Bytes Software-Revision + <CR>

Possible Error Response:

```
UPA<CR>
```

Example for a response:

```
DESKID_ISO<5 Times Space>01000101<CR>
```

Interpretation: Product name: DESKID_ISO

Hardware-Revision: 01.00

Software-Revision: 01.01

3.3. Read Serial Number (RSN)

The RSN command gets the serial number of the Reader. It will be printed via UART interface can be needed for support reasons and has the form JJJJMMDDHHMMSS01 (Date and Time stamp).

3.4. Read Hardware Revision (RHR)

The RHR command gets the hardware revision of the Reader. It will be printed via UART interface, might be needed for support reasons and has the form MMSS (MainVersion Subversion). Old Readers might not have a coded hardware revision. The answer is ÿÿÿÿ in this case.

3.5. Standby (STB)

The standby command sets the reader in a power save mode. The RF power is turned off. This means that all tags that might be in the field will also be powered down. If successful it returns GN8 (“Good Night”). The reader will not accept any commands except reset (RST) until a Wake Up Command (WAK) is received. Standby has no parameters. Standby saves the antenna state. After wake it will be active or inactive like before. During Standby the RF power is switched off.

Instruction:

```
STB<CR>
```

Response, if successful:

```
GN8<CR>
```

Possible Error Response:

```
UPA<CR>
```

3.6. Wake Up (WAK)

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

Instruction:

```
WAK<CR>
```

Response, if successful:

```
GMO<CR>, DNS<CR> (if not in Standby-Mode)
```

Possible Error Response:

```
UPA<CR>
```

3.7. Read Input Pin (RIP)

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

0 to 1 for unidirectional GPIOs (seperated GPIs and GPOs) or 0 to 7 for bidirectional GPIOs (every one is both input and output). Dwarf15 and QR15 use bidirectional GPIOs. Quasar_MX and QuasarLR uses unidirectional. DeskID_ISO does not support any GPIOs.

If successful, it returns either HI! or LOW depending on whether the input pin is high or low. If the GPIOs are bidirectional the direction is changed to reading. Call RIP on a pin before connecting to the source if WOP was used on this pin or use RST! If the pin is in write mode the connected device may be damaged.

Instruction:

`RIP<SPACE>Pin_No<CR>`

e.g. (to read the first input pin): RIP 0<CR>

Response, if successful:

`HI!<CR>` for High-State

`LOW<CR>` for Low-State

Possible Error Response:

`NOR<CR>, EHX<CR>, UPA<CR>, NOS<CR>`

3.8. Write Output Pin (WOP)

This command is used to set the state of an output pin either to high or to low. It takes two parameters. The first parameter is the zero-based number of the output pin to be written to. The second parameter is either "HI" or "LOW" to set the according pin to high or low respectively. The possible parameter range is 0 to 3 for unidirectional GPIOs (seperated GPIs and GPOs) or 0 to 7 for bidirectional GPIOs (every one is both input and output). Dwarf15 and QR15 use bidirectional GPIOs. Quasar_MX and QuasarLR uses unidirectional. DeskID_ISO does not support any GPIOs.

If the GPIOs are bidirectional the direction is changed to writing. Call RIP on a pin before connecting to a source if WOP was used on this pin or use RST! If the pin is in write mode the connected device may be damaged.

Instruction:

`WOP<SPACE>Pin_No<SPACE>PIN_Setting<CR>`

e.g. Set pin 0 high: `WOP<SPACE>0<SPACE>HI<CR>`

e.g. Set pin 3 low: `WOP<SPACE>3<SPACE>LOW<CR>`

Response, if successful:

`OK!<CR>`

Possible Error Response:

NOR<CR>, EHX<CR>, UPA<CR>, NOS<CR>

3.9. Mode (MOD)

The mode command selects the ISO Anti-collision and transmission protocol to be used for tag-communication. Default value is ISO 15693-3 and currently no other protocol is supported by the standard firmware. On request a firmware with ISO 14443-B Mode (MOD 14B) is available except for QuasarLR). MOD has no effect at all on the QuasarLR (just compatibility reason).

If successful it returns OK!

Valid parameters for the Mode command are:

- 156 - selects ISO 15693-3

Instruction:

MOD<SPACE>Iso_Standard<CR>

Response, if successful:

OK!<CR>

Error Response:

UPA<CR>; NOS<CR>

Example for ISO/IEC 15693-3:

MOD<SPACE>156<CR>

3.10. Set RF Interface (SRI)

The Set RF-Interface Command uses predefined values to configure the RF transceiver chip for a given tag type or turns the RF off. SRI TIM may be used to disable the RF for a defined time. It will be enabled after the time even if the Subcarrier was not defined. The state is undefined in this case!

The two parameters for sub-carrier type and modulation depth (10% or 100%) are mandatory. Possible subcarrier types are Single-Subcarrier (423,75 kHz) and Double-Subcarrier (423,75 kHz, 484,28 kHz). See the datasheet of your tag for more details on this subject. Double subcarrier is not supported by QuasarLR.

NOTE

You have to use this command after starting the reader to start reading tags.

Instruction:

```
SRI<SPACE>Subcarrier<SPACE>Modulation<CR>

SRI<SPACE>OFF<CR>

SRI<SPACE>TIM<SPACE><Time>
```

Subcarrier: ss for Single-Subcarrier

ds for Double-Subcarrier

Modulation: 10 for 10% ASK (only with Single-Subcarrier)

100 for 100% ASK

TIM: 1 to 2000 in milliseconds

Response, if successful:

```
OK!<CR>
```

Possible Error Response:

```
UPA<CR>, NOS<CR>
```

Example (Possible combination):

```
SRI<SPACE>SS<SPACE>10<CR>     (e.g. most FRAM based tags)

SRI<SPACE>SS<SPACE>100<CR>    (e.g. ICode, Tag-It)

SRI<SPACE>DS<SPACE>100<CR>    (very rarely used, not supported
                              by QuasarLR)

SRI<SPACE>OFF<CR>             disables RF

SRI<SPACE>TIM<SPACE>500<CR>   disables RF for half a second
```

3.11. Set Timings (STT)

The Set Timings Command is used to write internal parameters used to control request or inventory timings. Most times default values work fine and you don't have to use this command. This command is not supported by QuasarLR.

NOTE

This command should only be used with deeper knowledge of ISO15693. The basic values and meaning are defined there. The data sheet of the used tag might be very helpful, too.

Instruction:

```
STT<SPACE>TimingType<SPACE>Value<CR>
```

Timing Types are:

T1max(==T1), T1min, T2, T3,TWmax, TWmin, TWwnd

Response, if successful:

```
OK!<CR>
```

Possible Error Response:

```
UPA<CR>, NOR<CR>, EDX<CR>
```

Example (Possible combination):

```
STT<SPACE>T3<SPACE>400<CR>
```

```
STT<SPACE>T1min<SPACE>310<CR>
```

Meanings of timings:

T1max(==T1): This is the maximal answer time from the end of sending any read alike request (including inventory requests) to the begin of the answer. Answers coming after this time will be ignored. The value is set in microseconds. Default is 310µs. Maximal value is 38600µs.

T1min: This is the minimal answer time for any read-alike request (including inventory request) to the begin of the answer. Answers coming before this time will be ignored. The value is set in microseconds. Default is 400µs. Maximal value is 38600µs. Values of 350µs are fine for some tags, this will give better results, especially in noisy environment.

T2: This is the minimal time after any answer (including inventory answer) to the begin of the next request. Commands coming before this time will be delayed. The value is set in microseconds. Default is 400µs. Maximal value is 38600µs.

T3: This is the minimal time after any request (including inventory reqest) to the begin of the next request. Commands coming before this time will be delayed. The value is set in microseconds. Default is 400µs. Maximal value is 38600µs. T3 needs a change for Inventory on 10% ASK! Add the nominal response time (T_{nrt}) to the default value as a first try if the value is given in the tags data sheet. If not try values manualy. Typical values are around 4000µs. A T3 value to high has no negative sides except the speed.

TWmax: This is a timing for write-alike requests. With option flag this is the time from the end of the initial request to the EOF triggering the answer. Without option flag it's the timeout value. Any answer after this time will be ignored. The value is set in microseconds. Default is 21000µs. A shorter time might be ok for your tag (or lager time needed). Maximal value is 38600µs.

TWmin: This is a timing for write-alike requests. With option flag this has no meaning. Without option flag it's the start value of the first receive window. As this will affect any receive window this value is very critical to all write-alike requests. Therefore the value is NOT in microseconds but in 8 ticks of base frequency (also look at the Iso15693 for better understanding. Default is 550. (550*8 ticks = 4400 ticks = 324,5µs). Maximal value is 0xFF00.

TWwnd: This is the window time (the size of the write-alike answer windows). This value is just the size, the window always starts at TWmin and restarts cyclically like Iso15693 states. The value is given in 8 ticks of base frequency. The default value is 16 (128 ticks (=9,44µs)). Changing this value is discouraged and should only be done in case of real need! Maximal value is 255.

3.12. Cyclic Redundancy Check On (CON)

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

NOTE

This has nothing to do with the Reader-to-tag-communication crc

If this feature is activated (default is off), the reader firmware expects a CRC16 (4 hex numbers) between all commands to the reader and the respective <CR>. Between the command and the CRC there is a space character which is included in the CRC calculation. All answers from the reader will also be extended accordingly. The CRC used uses the 8408 polynomial, starting value is 0xFFFF. This command will work with or without the (optional) CRC.

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

Appendix 1 shows a function in C/C++ to calculate the correct CRC16.

Instruction:

```
CON<CR>
```

or:

```
CON 819E<CR>, con 2EC5<CR>
```

Response, if successful:

```
OK! 9356<CR>
```

Possible Error Response:

```
UPA<CR>
```

3.13. Cyclic Redundancy Check Off (COF)

This command turns off the Cyclic Redundancy Check (CRC) of the computer-to-reader communication. This is the default setting. This command will work with or without the (optional) CRC.

If successful it returns OK!.

Instruction:

```
COF<CR>, or COF 4F5E<CR>, or cof E005<CR>
```

Response, if successful:

```
OK!<CR>
```

Possible Error Response:

```
UPA<CR>
```

3.14. End of Frame Mode (EOF)

This command turns on the End of Frame Delimiter (EOF). This means that after every complete message (frame) the last CR will be followed by an additional line feed (LF, 0x0A). This allows the user to build simpler parsers since it is clear when not to expect any further message from the reader. The EOF returns on the end of any Instruction (<CR>) indifferent to actions done or answer (if at least on byte answer is given) and on any CNR mode answer. CNR INV itself gives no EOF answer of its own. It comes with the first Inventory. Please keep in mind that in case of a watchdog reset you get a SRT errorcode after the reset. This SRT is without the EOF because of the reset (which has set back EOF to off)!

If successful it returns OK!.

Instruction:

```
EOF<CR>
```

Response, if successful:

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

Possible Error Response:

```
UPA<CR>
```

3.15. No End of Frame Mode (NEF)

This command turns off the End of Frame Delimiter (NEF). Now all messages from the reader are only signaled by a CR at the end.

Instruction:

```
NEF<CR>
```

Response, if successful:

```
OK!<CR>  (no <LF>)
```

Possible Error Response:

```
UPA<CR>
```

3.16. Verbose Level (VBL)

Most metraTec modules send a lot of data to the host about different states, error responses or other data. While this is useful to understand exactly what the reader is doing, in some situations you only want a response from the reader when something important is happening. The VBL command gives you the possibility to set the amount of data coming from the reader to the level you need.

Instruction:

```
VBL<SPACE>[Mode in Decimal]<CR>
```

Modes:

0: Only send necessary data

1: Leave out optional information

2: Send all data (default)

Possible error codes:

`UPA<CR>` Unknown parameter

`EDX<CR>` Mode-Parameter is missing or other characters than 0-9 given

The following responses form the reader are suppressed in VBL Mode 1:

```
NCL<CR>
```

```
CER<CR>
```

```
TDT<CR>
```

The following responses form the reader are suppressed in VBL Mode 0:

```
IVF<SPACE>XX<CR>
```

Error Codes will still be sent in VBL Mode 0!

3.17. SET Power Level (SET)

The QuasarLR allows different output power levels to match antenna size, tag size or tag position. The only parameter to SET is PWR. The power level is given in milliwatt (mW). The minimal value is 500, the maximum is 8000 with steps of 250.

Instruction:

```
SET<SPACE>PWR<SPACE>[Powerlevel decimal mW]<CR>
```

Response, if successful:

OK!<CR>

Possible error codes:

UPA<CR>	Unknown parameter
EDX<CR>	Mode-Parameter is missing or other characters than 0-9 given
NOR<CR>	power level lower than 500, higher than 8000 or not divisible by 250

3.18. Status (STA)

This command is only supported by the QuasarLR which prints out some status information.

Instruction:

```
STA<CR>
```

Response, if successful:

```
OK!<CR> if no error, ARH<CR> if antenna short circuit
Reader Mode:        AUTO<CR>, HOST<CR> or EXTERN<CR>
RF Status:          ON<CR> or OFF<CR>
RF Power:           in mW
Forward power:           in mW
Reflected power:    in mW
```

```
SWR:            no unit
Temperature:        XX.YY °C
```

Example Answer:

```
OK!<CR>
ReaderMode: HOST<CR>
RF Status OFF<CR>
RF Power: 500 mW<CR>
Forward Power: 390 mW<CR>
Reflected Power: 10 mW<CR>
SWR: 1.48<CR>
Temperature: 27.95 °C<CR>
```

Possible error codes:

`UPA<CR>` Unknown parameter

3.19. Read Sub-Version (RSV)

The QuasarLR contains submodules. The version information of each module might become handy in case of support questions. The command returns two versions containing main version, subversion and revision.

Instruction:

```
RSV<CR>
```

Response, if successful:

```
x.y.z<CR>
X.Y.Z<CR>
```

Possible error codes:

`UPA<CR>` Unknown parameter

4. Tag Manipulation Instructions

This list gives an overview of the existing Tag Manipulation Instructions.

Command	Name	Description
INV	Inventory	Sends the Inventory request to the tag (as defined in ISO)
REQ	Request	Sends a tag command (according to ISO) to the tag
CNR	Continous Repeat	Prefix for continuous / automatically repeated requests
BRK	Break	Stops continuous operation (ends a CNR prefix operation)

Table 2: *Overview of tag manipulation instructions*

The difference between Reader Instructions and Tag Instructions is that with the latter the target of the instruction is the tag itself. Since RFID is mostly about tags, their IDs and data stored on tags, the Tag Manipulation Instructions are used extensively in almost any program. Among these commands the CNR command has a special role, since it is not strictly a command by itself but a flag or prefix which changes the way the command following CNR is interpreted by the firmware of the reader.

4.1. Inventory (INV)

One of the two mandatory commands of the ISO 15693 is the Inventory command (“01”) which is used to read tag IDs. It is called inventory command because it allows finding all tags in the field using an anti-collision sequence. Even though this command could be called using the REQ command as described below, the parsing of the answers coming from the tags is complicated especially in case collisions are detected and the anti-collision sequence has to be executed. For this reason, the reader command set has the INV command which automatically takes care of this sequence in a single simple command.

The INV command takes a variable number of parameters in arbitrary order:

1. “SSL” (Single Slot): If one can be sure to have only one tag in the reader field at any time, setting this optional parameter makes the reader scan for tag IDs faster (as anti-collision is disabled). Bare in mind that if there is more than one tag in the field, you will get a collision - Reader response: CLD (Collision Detected) for all readers except for QuasarLR. The QuasarLR just finds no tag at all (IVF 00). This command is optional, the default is off.

2. “AFI”: Setting this optional parameter will lead to tags with the corresponding Application Field Identifier answering the INV command only – tags in other AFI groups do not answer. This can be used to filter the type of responding tags. The AFI group has to be supplied as a two-digit hexadecimal number. This command is optional, the default is off.

3. “ONT” only new tag sends only answers from tags being new in the field. This includes tags reentering the field. ONT is mainly used together with the CNR suffix and a low verbosity level. This command is optional, the default is off.

4. “BAR”: If used with CNR this command will cause the reader to stop CNR mode after and inventory with at least one tag found
5. “MSK”: Masking allows to just search for specific tags. This might contain the whole tag ID or just parts. The masking starts at the end meaning MSK 345 brings all tags with UID XXXXXXXXXXXXX345. MSK is not supported by QuasarLR

Some examples will help clarify the use of the INV command (these cover 99% of all cases):

- “INV”: Get the tag IDs of all tags in the field using the parameters from SRI
- “INV AFI XX”: Get tag IDs of all tags in the field in AFI group XX (these two digits have to be hex numbers for this to work)
- “INV SSL”: Get single tag ID (faster than INV)
- “CNR INV ONT” Continuously get all tags in the field and disable any tag found. The tags are re-enabled if they leave the field

Instruction:

```
INV<Space><Parameters with Spaces><CR>
```

Response, if successful:

In case the “SSL” parameter was set (get single tag ID, no anti-collision), the response will be one line:

for 0 Tags: IVF 00<CR> (IVF = Inventory Found)

for 1 Tag: e.g. E0040100078E3636<CR> (the tag ID)

for 2 Tags: CLD<CR> (collision detected – error! Use anti-collision (no SSL) in this case). CLD might also be caused by noise. CLD does not occur on QuasarLR.

In case the “SSL” parameter was not set (get all tag IDs, use anti-collision algorithm) the result is one tag ID per line followed by a closing line reporting the number of tags found. For e.g. two tags in the field:

```
E0040100078E3636<CR>

E0040100078E362E<CR>

IVF 02<CR>
```

Possible Error Response:

```
UPA<CR>
```

Examples:

INV<CR> Multi-Slot inventory

INV<SPACE>SSL<CR> Single-Slot inventory

INV<SPACE>AFI<SPACE>0F<CR> Multi-Slot inventory, AFI Group 0Fh

INV<SPACE>AFI<SPACE>0F<SPACE>SSL<CR> Single-Slot inventory, AFI Group 0Fh

4.2. Request (REQ, DRQ, WRQ, DWQ)

The general request command directly sends a sequence of hexadecimal digits passed to it to the tag(s). This is the most powerful command possible as it allows implementing all commands any tag IC offers without waiting for metraTec to support a given feature. It also requires the user to look up the tag commands in the according transponder IC datasheet, however. REQ takes the UID like printed from INV if addressed bit is set. In speacial cases this might cause problems. Use the DRQ command to directly send the raw data even in addressed mode. WRQ can be used to show the command used is write-alike. This is important for timings (see STT command and Iso15693). Standard (mandatory and optional) write commands are known and work with REQ, but manufacturer specific commands will need WRQ. DWQ combined DRQ and WDT being an direct mode for write-alike commands. The QuasarLR only supports REQ.

4.2.1. Commands according to ISO/IEC 15693-3

As an example, the commands defined in ISO 15693 falls into three categories: mandatory, optional and custom. Mandatory commands have to be supported by all tags claiming to adhere to this ISO norm. Optional commands can be supported as defined in the ISO, tag ICs do not have to support (all of) these, however. Custom commands finally are tag IC dependent and will differ between the different tag ICs even if they are all ISO compatible.

A reader that supports the reading of ISO 15693 tags thus has to offer the (two) mandatory commands defined in the ISO. Also it will usually offer a (manufacturer dependent) choice of the optional ISO commands. Here the manufacturer has to decide which of the optional tag commands it wants to implement – the users of most readers have to make do with these manufacturer decisions. Usually, the custom commands that each IC defines differently will not be supported by the reader command set as it does not know which IC it will face in an application. Potentially useful commands as e.g. EAS (electronic article surveillance) can thus not be supported in many cases by other reader manufacturers because different ICs implement them differently.

This is exactly where the REQ command comes in as it allows using the complete set of supported commands of all tag ICs. The sequence of hex digits that have to be sent to the tag IC can be taken out of the tag IC data sheet. Most common options and ISO15693 command codes are also listed on the following pages of this guide.

The command frame always consists of two flag bytes as well as two command bytes, followed by optional command parameters, data as well as a CRC16 which can be calculated by the reader automatically. The complete command frame is:

SOF	Flags	Command Code	Command-Parameters	Data	CRC16	EOF

SOF and EOF are added by REQ command by them selves. The CRC may be automatically added with parameter CRC. The general command format starts with two flag digits (1 hex Byte) which depend on some factors like type of subcarrier, whether the command is addressed or unaddressed and whether the option flag is used or not (needed for some commands by ISO15693). The following table shows the most common combinations. In case of addressed read the command contains the UID. Use the value given by INV command. REQ switches sequence to match ISO15693 by itself. If you do not want this service (e.g. for special manufacturer commands with other position of address parameter) use DRQ instead of REQ. For custom write-alike commands use DWQ and WRQ. As the QuasarLR only supports REQ custom write-alike commands will most likely not work on QuasarLR.

Hex Code	Settings
02	Single subcarrier, high data rate, unaddressed
22	Single subcarrier, high data rate, addressed
42	Single subcarrier, high data rate, unaddressed, with option flag
62	Single subcarrier, high data rate, addressed, with option flag
03	dual subcarrier, high data rate, unaddressed

Then come two further hex digits which signify the command itself, sometimes followed by additional parameters:

Hex Code	Command Name	Parameters
02	Stay Quiet	UID (m)
20	Read Single Block	UID(o)BlockNo(m)
21	Write Single Block	UID(o)BlockNo(m)Data(m)
22	Lock Block	UID(o)BlockNo(m)
23	Read Multiple Blocks	UID(o)BlockNo(m)#Blocks(m)
24	Write Multiple Blocks	UID(o)BlockNo(m)#Blocks(m)Data(m)
25	Select	UID(m)
26	Reset to Ready	UID(o)
27	Write AFI[1)]	UID(o)AFI(m)
28	Lock AFI	UID(o)
29	Write DSFID[2)]	UID(o)DSFID(m)
2A	Lock DSFID	UID(o)
2B	Get System Information	UID(o)
2C	Get Multiple Block Security Status	UID(o)BlockNo(m)#Blocks(m)

[1)] *Application Family Identifier*

[2)] *Data Storage Format ID*

The last column of the table shows which parameters can (o = optional) or have to (m = mandatory) be supplied to the command and their order. The tag ID (UID) for all ISO 15693 tags consists of 16 hex digits. BlockNo is the first block to be processed by the command and is a two digit hex number. #Blocks is the number of blocks to be processed and is also a two digit hex number. AFI and DSFID are two digit hex numbers. "Data" is the data to be written. The length of this depends on the individual tag IC (current tags have either 4 or 8 bytes per block) – for every byte of data per tag block, two hex digits are needed.

Every ISO 15693 request always has to end with a CRC to ensure safe communication between reader and tag (not to be confused with the optional CRC for computer-to-reader communication). The CRC algorithm is the same as the one used to secure PC to reader communication. To make your life easier, there is an auto-CRC routine which calculates the CRC automatically for you. To use this feature, simply add the characters "CRC" to each REQ command.

Instruction:

```
REQ<SPACE>FFXXYYYYYYYYYYYYYYYYPP<SPACE>CRC<CR>
```

where "FF" signifies the flag bytes, "XX" signifies the request as defined in the table above, "YYYYYYYYYYYYYYYY" is the UID of the tag being addressed (if used), optional parameters are shown as "PP".

4.2.2. Responses from Transponder

The response to the request can either be "TNR<CR>" (Tag did Not Respond) when no tag responded or "TDT<CR>" (Tag Detected) when a tag was found. In this case, three more lines will follow. In the second line contains the general response format, without SOF and EOF.

SOF	Flags	Parameters	Data	CRC16	EOF

The second line of response should start with 00 (if there was no error) followed by data in case the reply contains data (e.g. the UID or data read from a tag) and terminated by the according CRC. If the tag responded with an error, you will only get a two-digit error code and its corresponding CRC in response. The meaning of this code can be found in the datasheet of the respective tag IC.

The third line will state whether the CRC transmitted from the tag to the reader was correct by stating "COK" or "CER" in case an error occurred. You will typically get a CER error, when the tag is at the limit of the reading range or when there is a source of electromagnetic disturbance near the reader (like an electric motor as well as another reader). In this case you will have to repeat the command.

The final fourth line reports whether a tag collision was detected ("CLD" – this is an error) or not ("NCL"). If you receive a CLD response there was more than one tag in the field and you

did not use the addressed mode / anti-collision algorithm or there is an electromagnetic disturbance (noise). This will happens more often the less coupling there is between reader and tag so decreasing the distance might help.

CLD is not supported by the QuasarLR. The answer will always be NCL+TDT or TNR. CRC is checked on QuasarLR like written above.

Example successful response:

```
TDT<CR>

0078F0<CR>

COK<CR>

NCL<CR>
```

Possible Error Response:

WDL<CR>, UPA<CR>, EPX<CR>, EHX<CR>

4.3. Continuous Repeat Prefix (CNR)

Both commands INV and REQ/DRQ/WRQ/DWQ can be written after the "CNR" prefix and will then be repeated indefinitely or until the "BRK" command is sent (see below) or RST or, with BAR, after a tag was found. This is a very powerful mechanism for unassisted operations where the reader is initialized at the beginning 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.

Example: Read all tag IDs repeatedly until stopped

Instruction:

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

Response (exemplary, with two tags in the field):

```
E0040100078E3BB0<CR>

E0040100078E3BB7<CR>

IVF 02<CR>

E0040100078E3BB0<CR>

E0040100078E3BB7<CR>

IVF 02<CR>

E0040100078E3BB0<CR>
```

```
E0040100078E3BB7<CR>

IVF 02<CR>

...
```

Optional Parameter: ONT (Only New Tags)

Using this parameter, the reader will only report new tag ids to the host so you don't have to filter for already known tags. As long as a card/transponder stays in the field (and is powered) it will not respond a second time.

Optional Parameter: BAR (Break At Read)

To automatically break with the first inventory run that finds at least one tag use the BAR parameter. This saves having to use BRK when after finding a tag.

Example: Wait silently for a tag to enter the field, report its ID and then stop. For this to be silent, VBL should be set to 0 (see Verbosity Level).

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

Response when a tag enters the field like with normal inventory plus additional Break Acknowledge to confirm the continuous mode has been left.

```
E0040100078E3BB0<CR>

IVF 01<CR>

BRA<CR>
```

4.4. Break (BRK)

To end the continuous mode entered into by the "CNR" prefix, the break command can be sent. This will lead to the complete execution of the current command iteration and will then lead to a "BRA" (break acknowledged) response. The command needs no parameter and returns no error codes.

Instruction:

```
BRK<CR>
```

Response:

```
BRA<CR>
```

Possible Error Response:

```
UPA<CR>
```

5. Error Codes

Error Code	Name	Description
ARH	Antenna Reflectivity High	Short circuit on antenna. Shut RF down (might be shut down already as it is tested periodic)
BOD	BrownOut Detected	The reader was reset because of inappropriate power supply. This is a normal error if the reader was de-powered before.
BOF	Buffer OverFlow	To many data in buffer. The buffer will be deleted. This error might be caused from UART input or output error. Common error source: - Read (or other) command for to many blocks, answer to long - To many or to long command (Formerly DAT error code) - Error in answer length caused by noise
CCE	Communication CRC Error	A CRC-Error has detected while receiving a line from Host-System. On QuasarLR might also be a internal CRC error
CER	CRC Error RFID	The Tag communication was disturbed. Perhaps by noise or tag problems (for example incomplete answer)
CLD	Collision detected	Collision detected, a collision has been detected during the tag communication. Not supported by QuasarLR. Common error source: - More than one tag in reader field but single slot inventory or unaddressed operation requested
CRT	Command Received Timeout	The reader received a character (as part of a command) but did not receive the final <CR> in time (5ms) resulting in this error. Common error source: - wrong programming - noise on the communication line
DNS	Did Not Sleep	A Break Command (WAK) 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'
EHF	Error Hardware Failure	Internal hardware error. If it occures more than once contact metraTec support.
EHX	Error hex expected	Parameter string cannot be interpreted as a valid hexadecimal number. Common error source: - Other character than 0-F
NCM	Not in CNR Mode	Common error source: - BRK send while no CNR mode is active

NOR	**N**umber **O**ut of **R**ange	Common error source: - RIP, WOP: Value is higher than I/O-Pins - Length of data or parameter value is not allowed
NOS	**N**o**t** **S**upported	Command or parameter not supported by this specific device type
RNW	**R**egisters **n**ot **w**ritten	The reader's configuration registers are not configured. Common error source: - No configuration register settings were entered since the last power up or reset but a tag operation attempted, SRI operation required first This error does not occur on QuasarLR. SRI should be used after the start anyways for a defined state
RXE	**R**eceived not as e**x**pecte**d**	Internal communication error on QuasarLR
SRT	**S**elf **R**ese**T**	Watchdog occurred. Might occur with many tags, extreme collisions and retries, and/or high times in STT command. If nothing of this there might be a hardware failure. Please contact metraTec support if it occurs more often than once. The watchdog time is 2 seconds
TMT	**T**o **M**any **T**ags	More than the maximal savable number of tags was found (26)
TNR	**T**ransponder **O**ut of **R**ange	No valid tag answer on request (REQ, WRQ, DRQ, DWQ)
UCO	**U**nknown **co**mmand	An invalid command has been passed to a function. Common error source: - Typo in command string - Wrong firmware version
UPA	**U**nknown **pa**rameter	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)
UER	**U**nknown **er**ror	General error that is not further specified. Should not occur at all. Used for unspecified states or behavior
UER XX	**U**nknown **E**rrorcode	Decoding of internal error code failed. The code is shown as hex code. This is a bug. Contact metratec support for more information or feedback.
URE	**U**art **R**eceive **E**rror	Uart Data corrupted. Try again. Perhaps check cable and EMV.
WDL	**W**rong **d**ata **l**ength	The length of the parameter data string does not match the requested operation. Common error source: - Too few or to many bytes were passed to a write operation

6. Examples

This section lists some of the more common tasks in connection with our RFID readers/writers. The examples will enable you to start programming our products in shorter time and without too much consultation of the official ISO1569.

6.1. Typical Reader Initialization Sequence

This example shows a typical initialization sequence to read ISO15693 tags, esp. tuned for fast reading of the very common NXP Icode transponders.

First configure the right mode for communication, in this case being single subcarrier, 100% ASK modulation:

Instruction:

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

Response:

```
OK!<CR>
```

Now the reader is ready to read from and write to tags. For other tag types, use different SRI (esp. when working with FRAM tags, which almost always use 10% ASK modulation).

6.2. Reading the Tag-ID of a transponder

By far the most common operation done with an HF RFID tag is reading the unique ID of a transponder. There are several posibilities 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.

To simply read the Ids of all tags in the field (anti-collision) use only the INV command:

Instruction:

```
INV<CR>
```

Response:

```
IVF 00<CR>
```

if there is no tag

or a number of tag-IDs and a line showing the number of tags found. In case there are two tags in the field this might look like this:

```
E0040100078E3BB0<CR>

E0040100078E3BB7<CR>

IVF 02<CR>
```

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 anti-collision algorithms which makes the operation even faster. In this mode it is possible to read HF tags with rates of up to 150 tags/sec (80 tags per second for QuasarLR).

Instruction:

```
INV SSL<CR>
```

Response:

```
INV 00<CR>
```
if there is no tag

or a tag-ID:

```
E0040100078E3BB0<CR>
```

or an error showing that there was more than one tag in the field (Collision detected, not supported by QuasarLR):

```
CLD<CR>
```

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.

6.3. Reading Tag IDs continuously

All commands can be processed by the reader continiuously be using the CNR pre-fix. With the help of this command it is possible to make the reader read all Ids 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).

Example:

Instruction:

```
CNR INV
```

Response:

```
E0040100078E3BB0

E0040100078E3BB7

IVF 02
```
(if 2 tags are in field)

```
E0040100078E3BB0
E0040100078E3BB7
IVF 02                    (if 2 tags are in field)
E0040100078E3BB0
E0040100078E3BB7
IVF 02                    (if 2 tags are in field)
....
....
```

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

Instruction:

```
BRK
```

Response:

```
BRA
```

6.4. Example for writing and reading to and from ISO 15693 tags

As a last example we show how to write and read data to and from a tag in unaddressed mode. This is special since more readers do not support this feature (you always have to know the tag ID first). In this example we write a single block (4 bytes) in single subcarrier communication.

Instruction:

```
REQ 02210311112222 CRC<CR>    (write 11112222 data to block 3)
```

Response:

```
TDT                 (Tag Detected)
0078F0              (00=> status Okay; 78F0 = CRC16)
COK                 (CRC Okay)
NCL                 (No collision)
```

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

Instruction:

`REQ 022003 crc`	(Read the data from block 3)

Response:

`TDT`	(Tag Detected)
`00111122220000000013BA`	(00=> status Okay, the data, followed by CRC16)
`COK`	(CRC Okay)
NCL	*(No collision)*

7. Appendix

Appendix 1: CRC Calculation

```
// this function calculates a CRC16 over a unsigned char Array with, LSB first
// @Param1 (DataBuf): An Array, which contains the Data for Calculation
// @Param2 (SizeOfDataBuf): length of the Data Buffer (DataBuf)
// @Param3 (Polynom): Value of the Generatorpolynom, 0x8408 is recommended
// @Param4 (Initial_Value):  load value for CRC16, 0xFFFF is recommended for
//                           host to reader communication
// return: calculated CRC16

unsigned short GetCrc(              unsigned char *DataBuf,
                                    unsigned char SizeOfDataBuf,
                                    unsigned short Polynom,
                                    unsigned short Initial_Value)
{
        unsigned short Crc16;
        unsigned char Byte_Counter, Bit_Counter;

        Crc16 = Initial_Value;
        for (Byte_Counter=0; Byte_Counter < SizeOfDataBuf; Byte_Counter++)
        {
                Crc16^=DataBuf[Byte_Counter];
                for (Bit_Counter=0; Bit_Counter<8; j++)
                {
                        if (( Crc16 & 0x0001)==0) Crc16>>=1;
                        else Crc16=(Crc16>>1)^Polynom;
                }
        }
         return (Crc16);
}
```

8. Version Control

Version	*Change*	*by*	*Date*
1.0	created	KD	15.07.2009
1.1	RSN command added	KD	27.10.2010
1.2	VBL, CRT added, SRT, WRR removed	KD	12.07.2011
2.0	Changes for Firmware Version 2.0	MK	12.08.2011
2.1	Minor corrections, matches FW version 2.4	MK	24.10.2011
2.2	Reset Timings added, QuasarLR added, SRI completed, Added error codes to table, merged the two tables Added max values to STT commands. Corrected milliseconds to microseconds in STT description	MK	12.09.2012

Contact / Support

metraTec GmbH
Werner-Heisenberg-Str. 1
D-39106 Magdeburg

Tel.: +49 (0)391 251906-00
Fax: +49 (0)391 251906-01

Email: support@metratec.com
Web: http://www.metratec.com

Copyright