

Scotty Reader
User Manual
Version 1.5 (draft)

TECTUS Transponder Technology GmbH, Moers

May 27, 2013

Contents

1	Serial Communication Protocol	2
1.1	Modes	2
1.1.1	Human mode	2
1.1.2	Machine mode	2
2	The Command Interpreter	4
2.1	Introduction	4
2.2	Arguments	4
2.3	Command Execution	4
2.4	Command Description	5
3	Error Handling	6
3.1	Introduction	6
3.2	Host Errors	6
3.3	Command Errors	6
3.4	Host Errors List	6
4	System Commands	8
4.1	The Main Reader Commands	9
4.1.1	boot	9
4.1.2	mode	9
4.1.3	kill	9
4.1.4	sfw	9
4.1.5	sfr	9
4.1.6	skw	9
4.1.7	skr	9
4.1.8	sv	9
4.1.9	se	9
4.1.10	st	9
4.1.11	lx	10
4.1.12	sc	10
4.1.13	sd	10
4.1.14	hn	10
4.1.15	hl	10
4.1.16	hh	10
4.1.17	ha	10
4.2	Phaser Hardware Commands	11
4.2.1	10	11

4.2.2	l1	11
4.2.3	ld1	11
4.2.4	ld2	11
4.2.5	ls	11
4.3	Signal Processing Debugging Commands	12
4.3.1	ss	12
4.3.2	ahu	12
4.3.3	ahl	12
4.3.4	ahz	12
4.3.5	abu	12
4.3.6	abz	12
4.4	UNIQUE read-only Commands	13
4.4.1	ru	13
4.4.2	rv	13
4.4.3	Function Error Codes	13
4.5	FDX-B 11785 (ZOODIAC) read-only Commands	14
4.5.1	rz	14
4.5.2	rq	14
4.5.3	ro	14
4.5.4	Function Error Codes	14
4.6	TROVAN read-only Commands	15
4.6.1	rt	15
4.6.2	tav	15
4.6.3	Function Error Codes	15
4.7	TIRIS - single page Commands	16
4.7.1	xr	16
4.7.2	xo	16
4.7.3	x0	16
4.7.4	xc	16
4.7.5	x1	16
4.7.6	xf	16
4.7.7	xw	16
4.7.8	xk	17
4.7.9	xd	17
4.7.10	x1	17
4.7.11	xz	17
4.7.12	Function Error Codes	17
4.8	TIRIS - multipage Commands	18
4.8.1	xmr	18
4.8.2	xmw	18
4.8.3	xmx	18
4.8.4	xmg	18
4.8.5	xmp	18
4.8.6	xmk	19
4.8.7	xm1	19
4.8.8	xmsr	19
4.8.9	xmsp	19
4.8.10	xmsk	19
4.8.11	xms1	19
4.8.12	Function Error Codes	19

4.9	EM4305 Commands	20
4.9.1	vl	20
4.9.2	vw	20
4.9.3	vk	20
4.9.4	vr	20
4.9.5	vd	20
4.9.6	vpD	20
4.9.7	vprs	20
4.9.8	vprd	20
4.9.9	vprg	20
4.9.10	vprr	21
4.9.11	vprr	21
4.9.12	vpws	21
4.9.13	vpwd	21
4.9.14	vpwg	21
4.9.15	vpwt	21
4.9.16	vpwr	21
4.9.17	Function Error Codes	21
4.10	HITAG-S (HITAG 1) Commands	22
4.10.1	hv	22
4.10.2	hs	22
4.10.3	hS	22
4.10.4	hc	22
4.10.5	hr	22
4.10.6	hw	22
4.10.7	hR	23
4.10.8	hW	23
4.10.9	hpD	23
4.10.10	hprs	23
4.10.11	hprd	23
4.10.12	hprw	23
4.10.13	hprg	23
4.10.14	hpr0	23
4.10.15	hpr1	23
4.10.16	hprrr	24
4.10.17	hprp	24
4.10.18	hpws	24
4.10.19	hpwd	24
4.10.20	hpww	24
4.10.21	hpwg	24
4.10.22	hpw0	24
4.10.23	hpw1	24
4.10.24	hpwr	24
4.10.25	hpwp	24
4.10.26	Function Error Codes	24
4.11	HITAG 2 Commands	25
4.11.1	h2ac	25
4.11.2	h2ap	25
4.11.3	h2r	25
4.11.4	h2i	25

4.11.5	h2w	25
4.11.6	h2h	25
4.11.7	h2y	26
4.11.8	h2s	26
4.11.9	h2pD	26
4.11.10	h2prs	26
4.11.11	h2prd	26
4.11.12	h2prw	26
4.11.13	h2prv	26
4.11.14	h2prx	26
4.11.15	h2prp	26
4.11.16	h2prg	26
4.11.17	h2pr0	27
4.11.18	h2pr1	27
4.11.19	h2prrr	27
4.11.20	h2pws	27
4.11.21	h2pwd	27
4.11.22	h2pww	27
4.11.23	h2pww	27
4.11.24	h2pwx	27
4.11.25	h2pwp	27
4.11.26	h2pwg	27
4.11.27	h2pw0	27
4.11.28	h2pw1	27
4.11.29	h2pwr	28
4.11.30	Function Error Codes	28
4.12	Safer Commands	29
4.12.1	kx	29
4.12.2	kc	29
4.12.3	ke	29
4.12.4	ks	29
4.12.5	kv	29
4.12.6	kf	29
4.12.7	ka	30
4.12.8	kl	30
4.12.9	kw	30
4.12.10	kr	30
4.12.11	Function Error Codes	31
4.13	Safer Communication Commands	32
4.13.1	kke	32
4.13.2	kks	32
4.13.3	kkv	32
4.13.4	kkw	32
4.13.5	kkrr	32
4.13.6	Function Error Codes	33
4.14	Q5 Commands	34
4.14.1	qw	34
4.14.2	qy	34
4.14.3	qW	34
4.14.4	ql	34

4.14.5	qz	34
4.14.6	qL	34
4.14.7	qa	34
4.14.8	qx	35
4.14.9	qr	35
4.14.10	qR	35
4.14.11	q2	35
4.14.12	q0	35
4.14.13	qm	35
4.14.14	qpD	35
4.14.15	qprs	35
4.14.16	qprg	35
4.14.17	qpr0	35
4.14.18	qpr1	36
4.14.19	qprd	36
4.14.20	qpws	36
4.14.21	qpwg	36
4.14.22	qpw0	36
4.14.23	qpw1	36
4.14.24	qpwd	36
4.14.25	Function Error Codes	36
4.15	TITAN Commands	37
4.15.1	t1	37
4.15.2	ts	37
4.15.3	tw	37
4.15.4	tr	37
4.15.5	t0	37
4.15.6	to	37
4.15.7	tpD	37
4.15.8	tprd	37
4.15.9	tpwd	38
4.15.10	Function Error Codes	38

Chapter 1

Serial Communication Protocol

The *Scotty Reader* has a RS-232 interface, through which it can communicate with a *host*. The communication settings of the reader are fixed at 9600 bps, 8 bits, no parity bit and 1 stop bit. No kind of handshake is used. The communication between the host and the reader is line oriented. A *line* is a sequence of ASCII characters ended by a CR character (ASCII code 13). Two modes of communication are supported: with and without echo.

1.1 Modes

1.1.1 Human mode

The communication mode where every received character is sent back to the host is called *human mode*. It is most suited for handling the reader by a human operator.

This communication mode is entered unconditionally whenever an ESC (ASCII code 27) character is received. The ESC character is not echoed back. An uncompleted command (a command for which the final CR wasn't sent yet), if any, is aborted and a prompt is issued (the 'greater as' character), then the reader is ready to accept a new command. The next character must be sent to the reader only after the previous character was echoed back. This rule must be observed especially if the reader is going to be driven by automated software (the typical case). Since this rule would slow down communication a further communication mode is provided, described below. After command completion a prompt is issued and the reader is ready to accept a new command. Commands which don't have a proper response (certain write commands and similar) don't generate any reply.

1.1.2 Machine mode

Sending a '@' (ASCII code 64) character to the reader sets the reader in the *machine mode*. The '@' character is not echoed back. Any uncompleted command is aborted. The characters following the '@' up to the final CR form the current

command. After the **CR** is received, the response is issued. Every response has at least one line. For commands which have no proper response, a line containing a single dot '.' is printed, for synchronization purposes. No characters are to be sent to the reader until the complete response is received. In this mode it is recommended, although not necessary, that every command be prefixed with the '@' mode switch character.

In both modes, if a command is erroneous, either by syntax, or illegal values, an error response is generated starting with the phrase: '**Error:**', followed by an error number (in decimal) and the final **CR**, of course.

After power-up the reader enters the *human mode*.

Chapter 2

The Command Interpreter

2.1 Introduction

The *Scotty Reader* can be controlled by a *host* using commands. A *command* is one line containing the command name and, as required by the command description, zero, one or more numerical arguments. All lines must be ended with the CR character. Lines printed by the reader follow the same rule.

2.2 Arguments

A numerical argument can be expressed as a decimal number, a hexadecimal number preceded by the 0x or 0X prefix, an octal number which starts with a 0, or a binary number starting with 0b or 0B. The following strings describe the same number:

```
37
0X25
045
0b0100101
```

2.3 Command Execution

The execution of a command is commenced only after the final CR character is received, if no errors were encountered. This guarantees that commands can be safely issued both in *human* and *machine mode*. For commands which always succeed and thus return no data, a line containing a single dot is printed in *machine mode*. In *human mode* nothing is printed. The prompt is issued directly. For all the other commands, one line containing the result is printed.

Success or logical *true* values are signaled with the '+' (plus) character. Failure or logical *false* values are signaled with the '-' (minus) character. Typically, for failed operations a command error number is printed. Its meaning is described at the end of the chapter in which the command is described. Successful operations may print some supplementary data after the '+' character. See the individual command definitions.

2.4 Command Description

Every command has its syntax described in the 'Command:' line. Command names are specified in **constant width** typeface. Arguments are specified in *italic* typeface. The \longrightarrow indicates the end of a command. Thereafter the results are described. A single point indicates that no results are printed (but a line containing a dot is printed in *machine mode*). In all other cases the format of the result is described. Alternative formats are separated by the word 'or '.

Unless otherwise noted, numerical values are output in decimal. Tag ID values and, generally, data stored in tags is output in hexadecimal.

Chapter 3

Error Handling

3.1 Introduction

There are two types of errors.

The first type, called *host errors*, includes errors related to the host-reader link. These can be syntax errors, missing or illegal arguments, unsupported features and other types of errors. They signal that something is wrong with the driving software. These errors must be corrected in order to insure proper reader operation.

The second type, called *command errors*, includes errors related to the reader-tag link. They signal some error during communication with a tag. A host should be prepared to handle this type of errors as required by the application.

3.2 Host Errors

Host errors occur when something is wrong with the data the reader received from the host.

3.3 Command Errors

When executing a command, a successful execution is signaled with a + character followed by zero, one or more result values, as shown in the command description. If the command failed, a - is displayed followed by the error number in decimal.

3.4 Host Errors List

The complete list of host errors is enumerated below. Some errors can never occur in certain firmware versions.

Error: 6 - `ERROR_NUMBER_EXPECTED` – A number was expected at this place. This is a specific kind of a syntax error.

Error: 9 - `ERROR_SYNTAX` – Some kind syntax error has occurred.

Error: 21 - ERROR_BAD_HEX – An illegal character followed a 0x hex number prefix.

Error: 54 - ERROR_NO_COMMAND – no such command is available in the firmware.

Error: 55 - ERROR_SERIAL_RESUME – a fake error message after the `btrw` command.

The following errors should never occur. They signal a bug in the reader firmware. Should any of these errors be encountered, please take a minute to note the context in which the error appeared and contact the supplier.

Error: 128 - ERROR_SYS_UNGET – system error

Error: 129 - ERROR_SYS_SAMPLING_UNEXPECTEDLY – system error

Error: 130 - ERROR_SYS_NOT_SAMPLING – system error

Error: 131 - ERROR_SYS_SMALL_DELAY – system error

Error: 132 - ERROR_SYS_NO_SUCH_COMMAND – system error

Chapter 4

System Commands

In the following sections all commands supported by the *Scotty Reader* are enumerated. A short description of each command is given.

For details pertaining commands which support specific tag types, please refer to the associated tag datasheet.

4.1 The Main Reader Commands

4.1.1 boot

Command: `boot` \rightarrow .
reboots the reader.

4.1.2 mode

Command: `mode val` \rightarrow .
sets the new mode of the reader

4.1.3 kill

Command: `kill` \rightarrow .
halt any running cotask

4.1.4 sfw

Command: `sfw val` \rightarrow .
sets the flushing mode.

4.1.5 sfr

Command: `sfr` \rightarrow *val*
displays the flushing mode in hex.

4.1.6 skw

Command: `skw val` \rightarrow .
sets the I/O turnaround delay in 10 ms increments.

4.1.7 skr

Command: `skr` \rightarrow *val*
displays the I/O turnaround delay.

4.1.8 sv

Command: `sv` \rightarrow *firmware-version-string*
prints the current firmware version.

4.1.9 se

Command: `se` \rightarrow *error-number*
prints the error number of the last executed command.

4.1.10 st

Command: `st val` \rightarrow *val*
this is a test command. It simply prints the input value.

4.1.11 lx

Command: `lx` \rightarrow *val*
always returns 1.

4.1.12 sc

Command: `sc` \rightarrow *val*
returns 1 if the serial connection is not active (the RTS signal is not active), and 0 otherwise. It can be used to maintain a loop while no serial connection is made.

4.1.13 sd

Command: `sd val` \rightarrow .
generates a delay of *val* milliseconds. *val* is silently truncated to a 16-bit quantity.

4.1.14 hn

specify the address of the word(s) to be read

4.1.15 hl

specify the count of the word(s) to be read

4.1.16 hh

check the auto-hitag parameters

4.1.17 ha

setup the hitag automode

4.2 Phaser Hardware Commands

The following commands are used for individual read/write operations in the Phaser readers.

4.2.1 10

Command: 10 \rightarrow .

Turns the carrier off. No reading is possible with this setting. The power consumption of the reader is greatly reduced.

4.2.2 11

Command: 11 \rightarrow .

Turns the carrier on.

4.2.3 1d1

Command: 1d1 *val* \rightarrow

if *val* is nonzero LED1 is lit, otherwise it is turned off.

4.2.4 1d2

Command: 1d2 *val* \rightarrow

if *val* is nonzero LED2 is lit, otherwise it is turned off.

4.2.5 1s

Command: 1s *val* \rightarrow

selects the proper input channel

4.3 Signal Processing Debugging Commands

4.3.1 ss

Command: **ss** \rightarrow *val*

returns the power of signal received in the last command in arbitrary units.

4.3.2 ahu

Command: **ahu** \rightarrow .

a debugging command.

4.3.3 ahl

Command: **ahu** \rightarrow .

a debugging command.

4.3.4 ahz

Command: **ahz** \rightarrow .

a debugging command.

4.3.5 abu

Command: **abu** \rightarrow .

a debugging command.

4.3.6 abz

Command: **abz** \rightarrow .

a debugging command.

4.4 UNIQUE read-only Commands

4.4.1 ru

Command: `ru` \rightarrow + 10-digit-tagID or -
reads an Unique Tag at RF/64 bitlength with ASK modulation and Manchester encoding. If successful, the tag serial number is stored in variables `c` (low 32 bits) and `d` (high 8 bits).

4.4.2 rv

Command: `rv` \rightarrow + 10-digit-tagID or -
reads an Unique Tag at RF/32 bitlength with ASK modulation and Manchester encoding. If successful, the tag serial number is stored in variables `c` (low 32 bits) and `d` (high 8 bits).

4.4.3 Function Error Codes

1	No tag found
---	--------------

4.5 FDX-B 11785 (ZOODIAC) read-only Commands

4.5.1 rz

Command: **rz** \rightarrow + 16-digit-tagID or -
reads a Zodiac Tag at RF/32 bitlength with ASK modulation and Biphasic encoding.

4.5.2 rq

Command: **rq** \rightarrow + 16-digit-tagID or -
reads a Zodiac Tag at RF/32 bitlength with ASK modulation and Biphasic encoding. Output is done in proper LSB first order.

4.5.3 ro

Command: **ro** \rightarrow + 16-digit-tagID 4-digit-CRC or -
reads a Zodiac Tag at RF/32 bitlength with ASK modulation and Biphasic encoding. The data and CRC is output even if the CRC doesn't match.

4.5.4 Function Error Codes

1	No tag found
2	CRC error

4.6 TROVAN read-only Commands

4.6.1 `rt`

Function: `bool rt()`

Command: `rt` \rightarrow + 10-digit-address or -

Word: `rt` (-- flag)

reads an Trovan Tag at RF/16 bitlength with PSK modulation and NRZ encoding. If successful, the tag serial number is stored in variables `c` (low 32 bits) and `d` (high 7 bits).

4.6.2 `tav`

4.6.3 Function Error Codes

1	No tag found
---	--------------

4.7 TIRIS - single page Commands

4.7.1 `xr`

Command: `xr` \rightarrow + 16-digit-data or -
reads a TIRIS Tag. If successful, the 64-bit tag serial number is displayed. Proper checking of framing and CRC is done with this command. Afterwards, the `x0` command can be used to examine the preamble, the `xc` command can be used to examine the CRC and the `x1` command can be used to examine the postamble. The details can be found in the TIRIS datasheets from Texas Instruments.

4.7.2 `xo`

Command: `xo` \rightarrow + 16-digit-data 4-digit-CRC or -
reads a TIRIS Tag unchecked. The 64 data bits and the 16 CRC bits are displayed. An error is generated only if no signal (not even noise) is present. This event is unlikely. The `x0`, `xc` and `x1` commands can be subsequently be used, but if no tag is present, they will yield arbitrary values.

4.7.3 `x0`

Command: `x0` \rightarrow val
displays the preamble bits of the previous `xr` or `xo` command. No actual reading is done by this command. It only fetches the data from the *reader's* memory.

4.7.4 `xc`

Command: `xc` \rightarrow val
displays the 16 CRC bits of the previous `xr` or `xo` command. No actual reading is done by this command. It only fetches the data from the *reader's* memory.

4.7.5 `x1`

Command: `x1` \rightarrow val
displays the postamble bits of the previous `xr` or `xo` command. No actual reading is done by this command. It only fetches the data from the *reader's* memory.

4.7.6 `xf`

Command: `xf` \rightarrow val
displays the measured frequency of the '0' bits in arbitrary units. Used for debugging purposes.

4.7.7 `xw`

Command: `xw` 64-bit-data 16-bit-CRC \rightarrow .
writes data into a R/W TIRIS Tag. The first argument contains 64 bits of the tag data payload. The second argument contains the 16 bit CRC. The CRC must be externally computed. This allows for other possible data validation

schemes, in conjunction with the **xo** command. In this case, a reading must also be validated in the host device.

No read-after-write check is made. Thus, this command never reports an error.

4.7.8 **xk**

*Command: **xw** 64-bit-data \rightarrow .*

writes data into a R/W TIRIS Tag. The argument contains 64 bits of the tag data payload. The 16 bit CRC is automatically computed.

No read-after-write check is made. Thus, this command never reports an error.

4.7.9 **xd**

*Command: **xd** delay \rightarrow .*

sets the delay time in RF cycles between carrier switch-off and begin of data reception. The start-up value is 64.

4.7.10 **x1**

*Command: **x1** \rightarrow .*

enters a loop in which TIRIS reads are performed continuously. Used for hardware debugging. Serial activity (received characters) do interrupt this command.

4.7.11 **xz**

*Command: **xz** \rightarrow val*

reads the '0'-frequency value for the last TIRIS read operation.

4.7.12 **Function Error Codes**

1	Received bit too long - abort
2	Preamble/postamble tag type mismatch
3	Not a R/O or R/W frame
4	CRC invalid

4.8 TIRIS - multipage Commands

In the description of the following commands *data* is a 64 bit quantity, *CRC* is a 16 bit quantity, *write-address* and *read-address* are 8 bit quantities and *selective-address* is a 24 bit quantity. *flag* is either + indicating a correct CRC value, or - otherwise.

For more information about these commands, please consult the *MTP, SAMPT and SAMPTS General Reference Manual* from Texas Instruments, document number 11-09-21-031.

4.8.1 xmr

Command: xmr → + type data CRC flag read-address

Charge Only Read (MPT 0/17) A universal TIRIS read command. *type* can be *r* for a read-only tag, *w* for a read-write tag or *m* for a multi-page tag. The *data* contains the 64-bit payload. The *CRC* contains the subsequent 16 CRC bits. *flag* is + if the CRC matches, or - if the CRC is bad. The *read-address* contains the 8 bit RO/RW pattern or the read address as the name implies for multipage tags. Note that less error checking is done compared to the TI recommendation. However the complete information needed for validation is included here.

This command, as well as all other commands from this section, properly set up the reader (carrier frequency, input channel) for operation. No preparation is necessary.

4.8.2 xmw

Command: xmw data CRC → + type data CRC flag read-address

Write Page (R/W) writes data into a R/W TIRIS Tag. The first argument contains 64 bits of the tag data payload. The second argument contains the 16 bit CRC. The CRC must be externally computed.

4.8.3 xmx

Command: xmx data → + type data CRC flag read-address

Write Page with automatic checksum (R/W) writes data into a R/W TIRIS Tag. The argument contains 64 bits of the tag data payload. The 16 bit CRC is automatically calculated with the CCITT algorithm.

4.8.4 xmg

Command: xmg write-address → + type data CRC flag read-address

General Read Page (MPT 0/17, SAMPT 0/17-24)

4.8.5 xmp

Command: xmp write-address data CRC → + type data CRC flag read-address

Program Page (MPT 0/17)

4.8.6 xmk

Command: xmk write-address data → + type data CRC flag read-address

Program Page - with automatic checksum (MPT 0/17). This is the same operation as xmp but the 16-bit CRC is calculated by the reader.

4.8.7 xml

Command: xml write-address → + type data CRC flag read-address

Lock Page (MPT 0/17)

4.8.8 xmsr

Command: xmsr selective-address write-address → + type data CRC flag read-address

Selective Read Page (SAMPT 0/17-24, SAMPTS 0/17-24)

4.8.9 xmsp

Command: xmsp selective-address write-address data CRC → + type data CRC flag read-address

Selective Program Page (SAMPT 0/17-24, SAMPTS 0/17-24)

4.8.10 xmsk

Command: xmsk selective-address write-address data → + type data CRC flag read-address

Selective Program Page - with automatic checksum (SAMPT 0/17-24, SAMPTS 0/17-24). This is the same operation as xmsp but the 16-bit CRC is calculated by the reader.

4.8.11 xmsl

Command: xmsl selective-address write-address → + type data CRC flag read-address

Selective Lock Page (SAMPT 0/17-24, SAMPTS 0/17-24)

4.8.12 Function Error Codes

1	Received bit too long - abort
2	Unrecognized TIRIS type (nor R0,RW or Multipage)

4.9 EM4305 Commands

4.9.1 vl

Command: **vl password** $\rightarrow +$ or $-$
executes the *Login Command* on the tag. *password* is the password which must match the password stored in the tag.

4.9.2 vw

Command: **vw adr data** $\rightarrow +$ or $-$
writes the data word *data* at address *adr* on the tag. The *Login Command* must have been previously executed.

4.9.3 vk

Command: **vk data** $\rightarrow +$ or $-$
executes the *Protect Command*. *data* contains the 32 bits of the protection word.

4.9.4 vr

Command: **vr adr** $\rightarrow +$ *data* or $-$
reads the data word *data* from address *adr* from the tag. The *Login Command* must have been previously executed.

4.9.5 vd

Command: **vd** $\rightarrow +$ or $-$
executes the *Disable Command* of the EM4305 tag. A successful *Login* command is necessary as a prerequisite.

4.9.6 vpD

Command: **vpD** \rightarrow .
brings all EM4305 operating parameters to default values.
Note: all timing parameters are expressed in RF cycles. This and all subsequent commands are not needed for normal operation.

4.9.7 vprs

reads the duration $b_{tx-speed}$ of a bit for transmitting on the reader side.

4.9.8 vprd

reads the duration t_{delay} of the non-modulating period of the first half of a transmitted logic '0'. See the specific data sheet for details.

4.9.9 vprg

reads the duration t_{gap} of the gap time of a transmitted logic '0'.

4.9.10 vp_{prt}

reads the duration $t_{start-gap}$ of the gap time of a first transmitted logic '0'.

4.9.11 vp_{rr}

reads the command-reponse delay.

4.9.12 vp_{ws}

sets the duration $b_{tx-speed}$ of a bit on the reader side. This is 64 for *Opt64* tags and 32 for *Opt32* tags.

4.9.13 vp_{wd}

sets the duration t_{delay} of the non-modulating period of the first half of a transmitted logic '0'. See the specific data sheet for details.

4.9.14 vp_{wg}

sets the duration t_{gap} of the gap time of a transmitted logic '0'. It can be longer than a half-bit time.

4.9.15 vp_{wt}

sets the duration $t_{start-gap}$ of the gap time of the first transmitted logic '0'. It can have any value.

4.9.16 vp_{wr}

sets the command-response delay.

4.9.17 Function Error Codes

1	no valid preamble
2	negative ('00000001') preamble
3	bad column parity
4	bad row parity
5	bad trailing '0' bit
6	weak bit in preamble

4.10 HITAG-S (HITAG 1) Commands

The following commands correspond one-to-one to the commands supported by the HITAG-S tags. In HITAG slang, a *page* is a 32 bit data word. For further details, please consult the HITAG-S datasheet. These functions can also be used for *HITAG 1* operation.

4.10.1 hv

Command: **hv** \rightarrow *tagUID*

performs the *UID request advanced* command. The result *tagUID* is a 32 bit tag serial number. The result is also stored in the *c* variable. A reading error is not signalled.

4.10.2 hs

Command: **hs** *tagUID* \rightarrow + *config* or -

performs the *Select* command. *tagUID* is the serial number of the tag to be selected. It can be determined with one of the functions presented above. If the command execution is successful, the tag configuration page *config* is stored in the *c* variable.

4.10.3 hS

Command: **hS** \rightarrow + *config* or -

performs the *Select* command. However, the needed serial number is taken from the *c* variable. In this way, a **hS** command can be issued immediately after an *UID request*, without explicitly specifying a tag address.

4.10.4 hc

Command: **hc** \rightarrow + *tagUID* or -

performs the *UID request advanced* command followed by the *Select* command. If successful, the result is the *tagUID* 32 bit tag serial number.

4.10.5 hr

Command: **hr** *adr* \rightarrow + *data* or -

reads one page from address *adr*. The value is left in the *c* variable, or returned as *data*. The tag must have been previously *selected*.

4.10.6 hw

Command: **hw** *adr data* \rightarrow + or -

writes one page containing *data* at address *adr*. The tag must have been previously *selected*.

4.10.7 hR

Command: **hR** *adr* \rightarrow + *data1 ... dataN* or -

reads from one up to four pages starting at address *adr* . Pages are read until the address becomes a multiple of four. The read values are stored consecutively in the variables **c,d,e,f**, respectively, or reported as *data1 ... dataN*. The tag must have been previously *selected*.

4.10.8 hW

Command: **hW** *adr data1 ... dataN* \rightarrow + or -

writes one up to four pages starting at address *adr* . Pages are written until the address becomes a multiple of four. The values are taken from the variables **c,d,e,f**, or from the command arguments, which must be supplied in the correct number. Of course, the tag must have been previously *selected*.

4.10.9 hpD

Command: **hpD** \rightarrow .

brings all HITAG-S operating parameters to default values.

Note: all timing parameters are expressed in RF cycles. This and all subsequent commands are not needed for normal operation.

4.10.10 hprs

(+400); reads the duration of the reset modulation gap. The actual gap is 400 RF cycles longer. This gap is issued before every *UID Request* command, i. e. the **hu**, **hv** and **hV** commands. Corresponds to the t_{reset} parameter.

4.10.11 hprd

(+200); reads the pause length after the reset modulation gap. Corresponds to the $t_{start-up}$ parameter. The actual pause length is 200 RF cycles longer.

4.10.12 hprw

reads the t_{wcom} time. Relevant for all commands.

4.10.13 hprg

reads the t_g gap time.

4.10.14 hpr0

reads the $T[0]$ logic 0 bit length.

4.10.15 hpr1

reads the $T[1]$ logic 1 bit length.

4.10.16 hprp

reads the t_{wresp} time. Relevant for all commands.

4.10.17 hprp

Function: long hprp()

Command: hprp $\rightarrow val$

Word: hprp (-- val)

(+500); reads the t_{PROG} time. Used in the *Write Page* command timing. The actual delay is 500 RF cycles longer.

4.10.18 hpws

(+400); sets the duration of the reset modulation gap. The actual gap is 400 RF cycles longer. This gap is issued before every *UID Request* command, i. e. the hu, hv and hV commands. Corresponds to the t_{reset} parameter.

4.10.19 hpwd

(+200); sets the pause length after the reset modulation gap. Corresponds to the $t_{start-up}$ parameter. The actual pause length is 200 RF cycles longer.

4.10.20 hpww

sets the t_{wcom} time. Relevant for all commands.

4.10.21 hpwg

sets the t_g gap time.

4.10.22 hpw0

sets the $T[0]$ logic 0 bit length.

4.10.23 hpw1

sets the $T[1]$ logic 1 bit length.

4.10.24 hpwr

sets the t_{wresp} time. Relevant for all commands.

4.10.25 hpwp

(+500); sets the t_{PROG} time. Used in the *Write Page* command timing. The actual delay is 500 RF cycles longer.

4.10.26 Function Error Codes

1	No proper SOF bit pattern was received
2	CRC value error

4.11 HITAG 2 Commands

The following commands correspond one-to-one to the commands supported by the HITAG 2 tags. In HITAG slang, a *page* is a 32 bit data word. For further details, please consult the HITAG 2 datasheet. The reference document is *Ht2prot.doc/Revision 2.1/October 1997*.

4.11.1 h2ac

Command: h2ac PRN secret-data \rightarrow + *serno config* or -
executes the *start-auth* instruction in Crypto Mode. *PRN* is the Pseudo Random Number and *secret-data* is the secret 32 bit word. See *Ht2prot.doc/Section 4.2.1/page 15*. The results are the same as described in the **kap** command.

4.11.2 h2ap

Command: h2ap password dummy \rightarrow + *serno config* or -
executes the *start-auth* instruction in Password Mode. *password* is a 32-bit password which must match the password stored in page 1 of the tag. If successful, the serial number *serno* and the configuration word *config* are reported.

4.11.3 h2r

Command: h2r adr \rightarrow + *data* or -
reads the data word *data* from page address *adr* . The tag must have been previously successfully selected with one of the **kap** or **kac** commands.

4.11.4 h2i

Command: h2i adr \rightarrow + *data* or -
reads the *inverted* data word *data* from page address *adr* . The tag must have been previously successfully selected with one of the **kap** or **kac** commands. This weird command *might* be useful for insuring superior data trasmission integrity if other means are not available.

4.11.5 h2w

Command: h2w adr data \rightarrow + or -
writes the word *data* at page address *adr* . The tag must have been previously successfully selected with one of the **kap** or **kac** commands. Check the tag data sheet for the significance of the lower four pages.

4.11.6 h2h

Command: h2h \rightarrow + or -
executes the *HALT* instruction on the tag. The tag must have been previously successfully selected with one of the **kap** or **kac** commands.

4.11.7 h2y

Command: **h2y** \rightarrow + *serno config* or -
executes a **kap** command with the factory default password: 0x4D494b52 (or 'MIKR').

4.11.8 h2s

Command: **h2s** \rightarrow + *serno* or -
executes only the first half of a *start-auth* instruction. Therefore only the serial number is read. The tag is left in an undetermined state. This command should be used only for debugging purposes.

4.11.9 h2pD

Command: **h2pD** \rightarrow + or -
brings all HITAG 2 operating parameters to default values.
Note: all timing parameters are expressed in RF cycles. This and all subsequent commands are not needed for normal reader operation.

4.11.10 h2prs

(+200); reads the duration of the reset modulation gap. The actual gap is 200 RF cycles longer. This gap is issued before every *start-auth* command, i. e. the **kap** and **kac** commands.

4.11.11 h2prd

(+200); reads the pause length after the reset modulation gap. Corresponds to the $t_{PowerUp}$ parameter. The actual pause length is 200 RF cycles longer.

4.11.12 h2prw

reads the t_{WAIT1} time. Relevant for all commands.

4.11.13 h2prv

reads the t_{WAIT2} time. Relevant for all commands.

4.11.14 h2prx

reads the t_{extra} time. Relevant for the 'kap' and 'kac' commands.

4.11.15 h2prp

(+500); reads the t_{PROG} time. Used in the *Write Page* command timing. The actual delay is 500 RF cycles longer.

4.11.16 h2prg

reads the t_{low} low field time. See *Ht2prot.doc/Section 3.4/page 10*.

4.11.17 h2pr0

reads the $T[0]$ logic 0 pulse length. See *Ht2prot.doc/Section 3.4/page 10*.

4.11.18 h2pr1

reads the $T[1]$ logic 1 pulse length. See *Ht2prot.doc/Section 3.4/page 10*.

4.11.19 h2prp

reads the supplemental command sequence repeat count. Default value is 0. See *Ht2prot.doc/Section 4.3/page 20*.

4.11.20 h2pws

(+200); sets the duration of the reset modulation gap. The actual gap is 200 RF cycles longer. This gap is issued before every *start-auth* command, i. e. the *kap* and *kac* commands.

4.11.21 h2pwd

(+200); sets the pause length after the reset modulation gap. Corresponds to the $t_{PowerUp}$ parameter. The actual pause length is 200 RF cycles longer.

4.11.22 h2pww

sets the t_{WAIT1} time. Relevant for all commands.

4.11.23 h2pww

sets the t_{WAIT2} time. Relevant for all commands.

4.11.24 h2pwx

sets the t_{extra} time. Relevant for the 'kap' and 'kac' commands.

4.11.25 h2pwp

(+500); sets the t_{PROC} time. Used in the *Write Page* command timing. The actual delay is 500 RF cycles longer.

4.11.26 h2pwg

sets the t_{low} low field time. See *Ht2prot.doc/Section 3.4/page 10*.

4.11.27 h2pw0

sets the $T[0]$ logic 0 pulse length. See *Ht2prot.doc/Section 3.4/page 10*.

4.11.28 h2pw1

sets the $T[1]$ logic 1 pulse length. See *Ht2prot.doc/Section 3.4/page 10*.

4.11.29 h2pwr

sets the supplemental command sequence repeat count. See *Ht2prot.doc/Section 4.3/page 20*. A large value makes the read/write times unnecessary longer.

4.11.30 Function Error Codes

1	bad bit in preamble
2	missing five consecutive '1's
3	bad direct command echo (the first)
4	bad inverted command echo (the second)
5	bad bit in 32 bit data
6	bad bit in 8 bit data

4.12 Safer Commands

The following commands implement the *safer mode* command set.

4.12.1 **kx**

*Command: **kx** \rightarrow lastkeyidx*

returns the index of the key used in the previous *safer* operation. This can be useful, if automatic key selection was used.

4.12.2 **kc**

*Command: **kc** \rightarrow lastCON2value*

returns the value of CON2 after the last *safer* operation. This is useful, if a tag is locked, because only 'or' operations are allowed on the CON2 lock word in this case.

4.12.3 **ke**

*Command: **ke** \rightarrow + or -*

erases all keys in the reader.

4.12.4 **ks**

*Command: **ks** idx key-h key-m key-l \rightarrow + or -*

sets the key with the index *idx* to the value given by *key-h*, *key-m* and *key-l*. Since those values are 32 bit integers a complete key has 96 bits. The function fails if *idx* is invalid (greater as 9), or if the key is already defined.

4.12.5

*Command: **kv** idx key-h key-m key-l \rightarrow + or -*

checks if the key with the index *idx* is the same as the key specified with *key-h*, *key-m* and *key-l*. Since the key space is really large, a brute-force attack for finding the keys is not feasible.

4.12.6 **kf**

*Command: **kf** idx lock \rightarrow + or -*

formats a tag (Hitag-S) with the key with the index *idx*. The parameter *lock* specifies (if non-zero) that formatting should be definitive, that is, the tag cannot be reformatted again. After formatting, and only after formatting, a tag can be successfully used for the *authenticate* - **ka**, *read* - **kr**, *write* - **kw** and *lock* - **kl** operations. There are several reasons why this function might fail. The reason for failure can be obtained by reading the error number with **kn**. Beware of the one-time-programmable lock operation. After formatting and locking, the data lock bits can still be set (with the **kl** command) but only in OTP mode (i. e. they can be reset no more).

4.12.7 ka

Command: **ka** *idx* \rightarrow + *serno* or -

authenticates a tag as being correctly formatted with the key with index *idx*. If *idx* is greater than 9, then the index is automatically searched, and it can be subsequently found with **kx**. If the authentication failed, the reason for this can be found by consulting the error number.

4.12.8 k1

Command: **k1** *idx lock-bits* \rightarrow + or -

sets the value of the 8 *lock-bits* in the CON2 byte of the tag configuration word. If bit 6 or bit 7 is set, then both bits will be set (in order to always protect 128 bits at once). If the tag was previously formatted and locked, these bits have OTP (one time programmable) behaviour.

- Bit 6(and 7) protects the data unit at adress 1
- Bit 5 protects the data unit at adress 2
- Bit 4 protects the data unit at adress 3
- Bit 3 protects the data unit at adress 4 and 5
- Bit 2 protects the data unit at adress 6 and 7
- Bit 1 protects the data unit at adress 8, 9, 10 and 11
- Bit 0 protects the data unit at adress 12, 13, 14 and 15

If the tag is locked, the bits which are already set must be kept set. The value of these bits can be read with a **ka**, **kc** command sequence.

4.12.9 kw

Command: **kw** *idx adr data0 data1 data2 data3* \rightarrow + or -

writes a 128 data unit, specified as four 32 bit words *data0* to *data3*, at address *adr* in the tag which is authenticated with the key *idx*. Automatic key search is possible, as above. If the operation failed, the reason can be found by consulting the error number. The valid address range depends on the actual type of the tag used. For a Hitag-S 2048 *adr* can range from 1 to 15. Address 0 cannot be used.

4.12.10 kr

Command: **kr** *idx adr* \rightarrow + *data0 data1 data2 data3* or -

reads a 128 bit data unit, stored at address *adr* in the tag which is authenticated with the key *idx*. Automatic key search is possible, as above. If the operation failed, the reason can be found by consulting the error number. The valid address range is the same as for the **kw** command.

4.12.11 Function Error Codes

1	key index out of range (greater than 9)
2	zero address not accepted for secure R/W operations
3	Login - tag data mismatch
4	Hash - read after write mismatch
5	Key is already defined
6	Key is different
7	Key is not defined
8	Format - read after write mismatch
9	Configure/Format - read after write mismatch
10	Auth - Invalid hash
11	Auth - No valid key found
12	Write - read after write mismatch
21,22	Login GetUID - operation failed
23,24	Login Select - operation failed
25,26	Login ReadBlock(0) - operation failed
31,32	Write Hash Write(2) - operation failed
33,34	Write Hash Write(3) - operation failed
35,36	Read Hash Read(2) - operation failed
37,38	Read Hash Read(3) - operation failed
41,42	Format WriteBlock(2) - operation failed
43,44	Format ReadBlock(2) - operation failed
45,46	Format/Config Write(1) - operation failed
51,52	Read Data ReadBlock(x) - operation failed
61,62	Write Data WriteBlock(x) - operation failed
63,64	Write Data ReadBlock(x) - operation failed

4.13 Safer Communication Commands

The following commands implement the *safer mode with encrypted communication* command set.

4.13.1 kke

Command: **kke** $\rightarrow +$ or $-$
erases all communication keys in the reader.

4.13.2 kks

Command: **kks** *idx key-h key-mh key-ml key-l* $\rightarrow +$ or $-$
sets the communication key with the index *idx* to the value given by *key-h*, *key-mh*, *key-ml* and *key-l*. Since those values are 32 bit integers a complete key has 128 bits. The function fails if *idx* is invalid (greater as 7), or if the key is already defined.

4.13.3 kkv

Command: **kkv** *idx key-h key-mh key-ml key-l* $\rightarrow +$ or $-$
checks if the communication key with the index *idx* is the same as the key specified with *key-h*, *key-mh*, *key-ml* and *key-l*. Since the key space is really large, a brute-force attack for finding the keys is not feasible.

4.13.4 kw

Function: `bool kw(long comm, long idx, long adr)`
Command: **kw** *comm idx adr data0 data1 data2 data3* $\rightarrow +$ or $-$
Word: **kw** (*comm idx adr -- flag*)
writes a 128 data unit, specified as four 32 bit words *data0* to *data3*, at address *adr* in the tag which is authenticated with the key *idx*. Automatic key search is possible, as above. If the operation failed, the reason can be found by consulting the error number. The valid address range depends on the actual type of the tag used. For a Hitag-S 2048 *adr* can range from 1 to 15. Address 0 cannot be used. The 128 data bits are decrypted with the communication key *comm*.

4.13.5 kkr

Function: `bool kr(long comm, long idx, long adr)`
Command: **kr** *comm idx adr* $\rightarrow +$ *data0 data1 data2 data3* or $-$
Word: **kr** (*comm idx adr -- flag*)
reads a 128 bit data unit, stored at address *adr* in the tag which is authenticated with the key *idx*. Automatic key search is possible, as above. If the operation failed, the reason can be found by consulting the error number. The valid address range is the same as for the **kw** command. The 128 data bits are encrypted with the communication key *comm*.

4.13.6 Function Error Codes

15	communication key index out of range (greater than 7)
16	communication key is different
17	communication key is already defined
18	communication key is not defined

4.14 Q5 Commands

The following commands correspond one-to-one to the commands supported by the Q5 tags. For further details, please consult the Q5 datasheet.

4.14.1 qw

Command: **qw** *adr data* $\rightarrow +$ or $-$
writes the data word *data* at address *adr* (in page 0). No read after write check is done.

4.14.2 qy

Command: **qy** *adr data* $\rightarrow +$ or $-$
writes the data word *data* at address *adr* in page 1. No read after write check is done.

4.14.3 qW

Command: **qW** *password adr data* $\rightarrow +$ or $-$
using *password* , writes the data word *data* at address *adr* . No check for operation success is done.

4.14.4 ql

Command: **ql** *adr data* $\rightarrow +$ or $-$
writes the data word *data* at address *adr* and locks it. As above, no checks are done.

4.14.5 qz

Command: **qz** *adr data* $\rightarrow +$ or $-$
writes and locks the data word *data* at address *adr* in page 1. As above, no checks are done.

4.14.6 qL

Command: **qL** *password adr data* $\rightarrow +$ or $-$
using *password* , writes and locks the data word *data* at address *adr* . No checks are done.

4.14.7 qa

Command: **qa** *password* $\rightarrow +$ or $-$
executes the Answer-On-Request command. Useful only for tags with the AOR bit set. The returned value is always true and has no meaning.

4.14.8 qx

Command: qx *idx* \longrightarrow *data*

reads the *idx* -th word after an Answer-On-Request command. Up to eight words can be read in this manner.

4.14.9 qr

Command: qr *adr* \longrightarrow + *data* or -

reads the *data* at address *adr* . The function leaves the result in variable *c*, if successful.

4.14.10 qR

Command: qR *password val* \longrightarrow + *data* or -

using *password* , reads the *data* at address *adr* . The function leaves the result in variable *c*, if successful.

4.14.11 q2

Command: q2 \longrightarrow + *16-digit-number* /alt -

sends the *64 bit page access* command to the tag, and then reads the 64 bit output by the tag. The function leaves the bits in variables *c*, *d*.

4.14.12 q0

Command: q0 \longrightarrow + or -

sends the *Reset* command to the tag. The return value is useless.

4.14.13 qm

Command: qm \longrightarrow + or -

sends the *Modulation defeat* command to the tag. The return value is useless.

4.14.14 qpD

Command: qpD \longrightarrow .

brings all Q5 operating parameters to default values.

Note: all timing parameters are expressed in RF cycles. This and all subsequent commands are not needed for normal operation.

4.14.15 qprs

read the start gap duration S_{gap} .

4.14.16 qprg

read the gap duration W_{gap} .

4.14.17 qpr0

read the '0' bit carrier-on time d_0 .

4.14.18 qpr1

read the '1' bit carrier-on time d_1 .

4.14.19 qprd

read the time from reader modulation stop to the start of the first received bit.

4.14.20 qpws

write the start gap duration S_{gap} .

4.14.21 qpwg

write the gap duration W_{gap} .

4.14.22 qpw0

write the '0' bit carrier-on time d_0 .

4.14.23 qpw1

write the '1' bit carrier-on time d_1 .

4.14.24 qpwd

write the time from reader modulation stop to the start of the first received bit.

4.14.25 Function Error Codes

1	Read operation failed
---	-----------------------

4.15 TITAN Commands

4.15.1 `tl`

Command: `tl password` $\rightarrow +$ or $-$
executes the *Login* command. *password* is the password which must match the password stored in the tag.

4.15.2 `ts`

Command: `ts oldpass newpass` $\rightarrow +$ or $-$
executes the *Write Password* command. *oldpass* is the actual (current) password. *newpass* is the new password which is to be written.

4.15.3 `tw`

Command: `tw adr data` $\rightarrow +$ or $-$
writes the data word *data* at address *adr* on the tag. The *Login* must have been previously executed, under certain circumstances. This command can write a single word. Multiple word writing is not supported.

4.15.4 `tr`

Command: `tr adr` $\rightarrow +$ *data* or $-$
reads the data word *data* from address *adr* from the tag. It is stored in the *c* variable. The *Login* must have been previously executed, under certain circumstances.

4.15.5 `t0`

Command: `t0` $\rightarrow +$ or $-$
executes the *Reset Command* on the tag.

4.15.6 `to`

Command: `to idx` $\rightarrow +$ or $-$
reads the *idx*-th data word output by the Titan tag in read-only mode. Word 0 starts after a double LIW sequence, then follows word 1 and so on. The read word is stored in the variable *c*.

4.15.7 `tpD`

Command: `tpD` \rightarrow .
brings all TITAN operating parameters to default values.
Note: all timing parameters are expressed in RF cycles. This and all subsequent commands are not needed for normal operation.

4.15.8 `tprd`

read the tpp duration t_{pp} .

4.15.9 tpwd

write the tpp duration t_{pp} .

4.15.10 Function Error Codes

1	no LIW sequence could be found
2	no double LIW in readonly operation
3	NAK received
4	no valid ACK or NAK received
6	no second, synchronous LIW in write password
7	no subsequent and complete LIW after LIW
8	no delimiting LIW in readonly operation
10	bit encoding error (no valid Manchester)
11	bit parity error
12	byte parity error
13	'0' trailing bit missing
14	NAK received in second sequence
15	no valid ACK or NAK receive in second sequence