

Scotty Reader  
User Manual  
Version 1.4

TECTUS Transponder Technology GmbH, Moers

September 2, 2009

# Contents

<b>1</b>	<b>Communication Protocol</b>	<b>5</b>
1.1	Modes . . . . .	5
1.1.1	Human mode . . . . .	5
1.1.2	Machine mode . . . . .	5
<b>2</b>	<b>The Command Interpreter</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Arguments . . . . .	7
2.3	Command Execution . . . . .	7
2.4	Command Description . . . . .	8
<b>3</b>	<b>Error Handling</b>	<b>9</b>
3.1	Introduction . . . . .	9
3.2	Host Errors . . . . .	9
3.3	Command Errors . . . . .	9
3.4	Host Errors List . . . . .	9
<b>4</b>	<b>System Commands</b>	<b>11</b>
4.1	The Main Reader Commands . . . . .	12
4.1.1	<code>boot</code> . . . . .	12
4.1.2	<code>sv</code> . . . . .	12
4.1.3	<code>se</code> . . . . .	12
4.1.4	<code>st</code> . . . . .	12
4.2	Common Hardware Commands . . . . .	13
4.2.1	<code>l1</code> . . . . .	13
4.2.2	<code>l0</code> . . . . .	13
4.2.3	<code>lx</code> . . . . .	13
4.2.4	<code>lf</code> . . . . .	13
4.2.5	<code>sc</code> . . . . .	13
4.2.6	<code>sd</code> . . . . .	13
4.2.7	<code>ss</code> . . . . .	13
4.2.8	<code>sws</code> . . . . .	14
4.3	Scotty Hardware Commands . . . . .	15
4.3.1	<code>sl1</code> . . . . .	15
4.3.2	<code>sl2</code> . . . . .	15
4.3.3	<code>spd</code> . . . . .	15
4.3.4	<code>spk</code> . . . . .	15
4.4	UNIQUE read-only Commands . . . . .	16

4.4.1	ru	16
4.4.2	rv	16
4.4.3	Function Error Codes	16
4.5	FDX-B 11785 (ZOODIAC) read-only Commands	17
4.5.1	rz	17
4.5.2	ro	17
4.5.3	Function Error Codes	17
4.6	TIRIS - single page Commands	18
4.6.1	xr	18
4.6.2	xo	18
4.6.3	x0	18
4.6.4	xc	18
4.6.5	x1	18
4.6.6	xf	18
4.6.7	xw	18
4.6.8	xk	19
4.6.9	xd	19
4.6.10	x1	19
4.6.11	Function Error Codes	19
4.7	TIRIS - multipage Commands	20
4.7.1	xmr	20
4.7.2	xmw	20
4.7.3	xmx	20
4.7.4	xmg	20
4.7.5	xmp	20
4.7.6	xmk	21
4.7.7	xm1	21
4.7.8	xmsr	21
4.7.9	xmsp	21
4.7.10	xmsk	21
4.7.11	xms1	21
4.7.12	Function Error Codes	21
4.8	HITAG-S (HITAG 1) Commands	22
4.8.1	hv	22
4.8.2	hs	22
4.8.3	hS	22
4.8.4	hr	22
4.8.5	hw	22
4.8.6	hpD	22
4.8.7	hprs	23
4.8.8	hprd	23
4.8.9	hprw	23
4.8.10	hprg	23
4.8.11	hpr0	23
4.8.12	hpr1	23
4.8.13	hprrr	23
4.8.14	hprp	23
4.8.15	hpws	23
4.8.16	hpwd	23
4.8.17	hpww	24

4.8.18	hpwg	24
4.8.19	hpw0	24
4.8.20	hpw1	24
4.8.21	hpwr	24
4.8.22	hpwp	24
4.8.23	Function Error Codes	24
4.9	HITAG 2 Commands	25
4.9.1	kac	25
4.9.2	kap	25
4.9.3	kr	25
4.9.4	ki	25
4.9.5	kw	25
4.9.6	kh	25
4.9.7	ks	26
4.9.8	ky	26
4.9.9	kpD	26
4.9.10	kprs	26
4.9.11	kprd	26
4.9.12	kprw	26
4.9.13	kprv	26
4.9.14	kprx	26
4.9.15	kprp	26
4.9.16	kprg	26
4.9.17	kpr0	27
4.9.18	kpr1	27
4.9.19	kprrr	27
4.9.20	kpws	27
4.9.21	kpwd	27
4.9.22	kpww	27
4.9.23	kpww	27
4.9.24	kpwx	27
4.9.25	kpwp	27
4.9.26	kpwg	27
4.9.27	kpw0	27
4.9.28	kpw1	27
4.9.29	kpwr	28
4.9.30	Function Error Codes	28
4.10	Q5 Commands	29
4.10.1	qw	29
4.10.2	qy	29
4.10.3	qW	29
4.10.4	q1	29
4.10.5	qz	29
4.10.6	qL	29
4.10.7	qa	29
4.10.8	qx	30
4.10.9	qr	30
4.10.10	qR	30
4.10.11	q2	30
4.10.12	q0	30

4.10.13	qm	30
4.10.14	qpD	30
4.10.15	qprs	30
4.10.16	qprg	30
4.10.17	qpr0	30
4.10.18	qpr1	31
4.10.19	qprd	31
4.10.20	qpws	31
4.10.21	qpwg	31
4.10.22	qpw0	31
4.10.23	qpw1	31
4.10.24	qpwd	31
4.10.25	Function Error Codes	31
4.11	TITAN Commands	32
4.11.1	t1	32
4.11.2	ts	32
4.11.3	tw	32
4.11.4	tr	32
4.11.5	t0	32
4.11.6	to	32
4.11.7	tpD	32
4.11.8	tprd	32
4.11.9	tpwd	33
4.11.10	Function Error Codes	33

# Chapter 1

## 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.

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

## 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`  $\longrightarrow$  `.`  
reboots the reader.

### 4.1.2 sv

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

### 4.1.3 se

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

### 4.1.4 st

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

## 4.2 Common Hardware Commands

The following functions are for general use.

### 4.2.1 11

*Command:* 11  $\longrightarrow$  .  
activates the output carrier at full power.

### 4.2.2 10

*Command:* 10  $\longrightarrow$  .  
deactivates the output carrier. This command can be used for power saving purposes, to allow a nearby reader to operate, to reset a number of tags in the field or any other purposes. The output carrier can be turned back on with the 11 command.

### 4.2.3 1x

*Command:* 1x  $\longrightarrow$  *val*  
returns 1 if the module is running on the external clock (TLM-30) or from the internal ceramic resonator (TLM-40), or 0 if it runs from the RC oscillator of the processor.

### 4.2.4 1f

*Command:* 1f *val*  $\longrightarrow$  .  
sets the frequency driving parameter to *val* . If *val* is 0, then the carrier has a frequency of 125 kHz with resonator precision. If *val* positive, then the carrier has a frequency of 134 kHz with resonator precision.

### 4.2.5 sc

*Command:* sc  $\longrightarrow$  *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.

### 4.2.6 sd

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

### 4.2.7 ss

*Command:* ss  $\longrightarrow$  *val*  
returns the power of signal received in the last command in arbitrary units.

#### 4.2.8 `sws`

*Command:* `sws val`  $\longrightarrow$  .

activates signal witnessing on the proper pins. This is a debugging command.

## 4.3 Scotty Hardware Commands

The following functions are available only in the *Scotty* readers.

### 4.3.1 s11

*Command:* **s11** *val*  $\longrightarrow$  .

lights up the LED1 if the argument is not zero, or clears it otherwise.

### 4.3.2 s12

*Command:* **s12** *val*  $\longrightarrow$  .

lights up the LED2 if the argument is not zero, or clears it otherwise.

### 4.3.3 spd

*Command:* **spd** *val*  $\longrightarrow$  .

activates per software the power-down outputs and friends

### 4.3.4 spk

*Command:* **spk** *val*  $\longrightarrow$  .

kills the reader -*j*, no consumption at all



## 4.4 UNIQUE read-only Commands

### 4.4.1 ru

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

### 4.5.2 ro

*Command:* **ro**  $\longrightarrow$  + *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.

### 4.5.3 Function Error Codes

1	No tag found
2	CRC error

## 4.6 TIRIS - single page Commands

### 4.6.1 `xr`

*Command:* `xr`  $\longrightarrow$  + 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.

### 4.6.2 `xo`

*Command:* `xo`  $\longrightarrow$  + 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 very unlikely. The `x0`, `xc` and `x1` commands can be subsequently be used, but if no tag is present, they will yield arbitrary values.

### 4.6.3 `x0`

*Command:* `x0`  $\longrightarrow$  val  
diplays 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.

### 4.6.4 `xc`

*Command:* `xc`  $\longrightarrow$  val  
diplays 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.

### 4.6.5 `x1`

*Command:* `x1`  $\longrightarrow$  val  
diplays 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.

### 4.6.6 `xf`

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

### 4.6.7 `xw`

*Command:* `xw` 64-bit-data 16-bit-CRC  $\longrightarrow$  .  
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.

#### 4.6.8 **xk**

*Command: **xw** 64-bit-data  $\longrightarrow$  .*

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.

#### 4.6.9 **xd**

*Command: **xd** delay  $\longrightarrow$  .*

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

#### 4.6.10 **x1**

*Command: **x1**  $\longrightarrow$  .*

enters a loop in which TIRIS reads are performed continuously. Used for hardware debugging. Serial activity (received characters) do interrupt this command (not necessarily on the first character received).

#### 4.6.11 **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

## 4.7 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.

### 4.7.1 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.

### 4.7.2 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.

### 4.7.3 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.

### 4.7.4 xmg

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

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

### 4.7.5 xmp

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

*Program Page* (MPT 0/17)

#### 4.7.6 xmk

*Command: xmk write-address data  $\longrightarrow$  + 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.*

#### 4.7.7 xml

*Command: xml write-address  $\longrightarrow$  + type data CRC flag read-address*

*Lock Page (MPT 0/17)*

#### 4.7.8 xmsr

*Command: xmsr selective-address write-address  $\longrightarrow$  + type data CRC flag read-address*

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

#### 4.7.9 xmsp

*Command: xmsp selective-address write-address data CRC  $\longrightarrow$  + type data CRC flag read-address*

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

#### 4.7.10 xmsk

*Command: xmsk selective-address write-address data  $\longrightarrow$  + 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.*

#### 4.7.11 xmsl

*Command: xmsl selective-address write-address  $\longrightarrow$  + type data CRC flag read-address*

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

#### 4.7.12 Function Error Codes

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

## 4.8 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.

### 4.8.1 hv

*Command:* **hv**  $\longrightarrow$  *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.

### 4.8.2 hs

*Command:* **hs tagUID**  $\longrightarrow$  + *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.

### 4.8.3 hS

*Command:* **hS**  $\longrightarrow$  + *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.

### 4.8.4 hr

*Command:* **hr adr**  $\longrightarrow$  + *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*.

### 4.8.5 hw

*Command:* **hw adr data**  $\longrightarrow$  + or -

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

### 4.8.6 hpD

*Command:* **hpD**  $\longrightarrow$  .

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.

#### 4.8.7 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.

#### 4.8.8 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.

#### 4.8.9 hprw

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

#### 4.8.10 hprg

reads the  $t_g$  gap time.

#### 4.8.11 hpr0

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

#### 4.8.12 hpr1

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

#### 4.8.13 hprr

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

#### 4.8.14 hprp

*Function:* `long hprp()`

*Command:* `hprp`  $\longrightarrow$  *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.

#### 4.8.15 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.

#### 4.8.16 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.



#### 4.8.17 hpww

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

#### 4.8.18 hpwg

sets the  $t_g$  gap time.

#### 4.8.19 hpw0

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

#### 4.8.20 hpw1

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

#### 4.8.21 hpwr

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

#### 4.8.22 hpwp

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

#### 4.8.23 Function Error Codes

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

## 4.9 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*.

### 4.9.1 kac

*Command: kac PRN secret-data*  $\longrightarrow$  + *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.

### 4.9.2 kap

*Command: kap password dummy*  $\longrightarrow$  + *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.

### 4.9.3 kr

*Command: kr adr*  $\longrightarrow$  + *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.

### 4.9.4 ki

*Command: ki adr*  $\longrightarrow$  + *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.

### 4.9.5 kw

*Command: kw adr data*  $\longrightarrow$  + 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.

### 4.9.6 kh

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

#### 4.9.7 ks

*Command:* ks  $\rightarrow$  + *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.

#### 4.9.8 ky

*Command:* ky  $\rightarrow$  + *serno config* or -

executes a *kap* command with the factory default password: 0x4D494b52 (or 'MIKR').

#### 4.9.9 kpD

*Command:* kpD  $\rightarrow$  + 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.

#### 4.9.10 kprs

(+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.

#### 4.9.11 kprd

(+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.

#### 4.9.12 kprw

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

#### 4.9.13 kprv

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

#### 4.9.14 kprx

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

#### 4.9.15 kprp

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

#### 4.9.16 kprg

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

#### 4.9.17 kpr0

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

#### 4.9.18 kpr1

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

#### 4.9.19 kprrr

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

#### 4.9.20 kpws

(+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.

#### 4.9.21 kpwd

(+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.

#### 4.9.22 kpww

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

#### 4.9.23 kpww

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

#### 4.9.24 kpwx

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

#### 4.9.25 kpwp

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

#### 4.9.26 kpwg

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

#### 4.9.27 kpw0

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

#### 4.9.28 kpw1

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

#### 4.9.29 kpwr

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.

#### 4.9.30 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

## 4.10 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.

### 4.10.1 qw

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

### 4.10.2 qy

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

### 4.10.3 qW

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

### 4.10.4 ql

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

### 4.10.5 qz

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

### 4.10.6 qL

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

### 4.10.7 qa

*Command: qa password  $\longrightarrow$  + 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.

#### 4.10.8 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.

#### 4.10.9 qr

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

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

#### 4.10.10 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.

#### 4.10.11 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**.

#### 4.10.12 q0

*Command:* **q0**  $\longrightarrow$  + or -

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

#### 4.10.13 qm

*Command:* **qm**  $\longrightarrow$  + or -

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

#### 4.10.14 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.

#### 4.10.15 qprs

read the start gap duration  $S_{gap}$ .

#### 4.10.16 qpRG

read the gap duration  $W_{gap}$ .

#### 4.10.17 qpr0

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

#### **4.10.18 qpr1**

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

#### **4.10.19 qprd**

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

#### **4.10.20 qpws**

write the start gap duration  $S_{gap}$ .

#### **4.10.21 qpwg**

write the gap duration  $W_{gap}$ .

#### **4.10.22 qpw0**

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

#### **4.10.23 qpw1**

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

#### **4.10.24 qpwd**

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

#### **4.10.25 Function Error Codes**

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



## 4.11 TITAN Commands

### 4.11.1 `tl`

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

### 4.11.2 `ts`

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

### 4.11.3 `tw`

*Command:* `tw adr data`  $\longrightarrow$  + 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.

### 4.11.4 `tr`

*Command:* `tr adr`  $\longrightarrow$  + *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.

### 4.11.5 `t0`

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

### 4.11.6 `to`

*Command:* `to idx`  $\longrightarrow$  + 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*.

### 4.11.7 `tpD`

*Command:* `tpD`  $\longrightarrow$  .  
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.

### 4.11.8 `tprd`

read the tpp duration  $t_{pp}$ .

#### 4.11.9    tpwd

write the tpp duration  $t_{pp}$ .

#### 4.11.10    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