TagTrans Programming Guide

Rev 3.1

Table of Contents

TagTrans V2.2 FW 0.12.4

1 Introduction

This guide is for programmers wishing to communicate with the TagTrans.

This document assumes that the battery is fully charged before any work or changes are carried out.

It defines the Power Management and the Instruction set to communicate with the TagTrans.

Communication between the TagTrans and the host computer is via Bluetooth. The Bluetooth service presents a communication port for the TagTrans. This runs at up to 9600 baud with 8 data bits, no parity, 1 stop bit and no handshake.

The TagTrans consists of several internal modules: the Bluetooth engine, the Imager engine and the MicroEngine with the RFID Module.

Some of the commands affect the fundamental operations of the TagTrans and should be used with caution. It is possible to alter the configuration in such a way that the TagTrans will not function. The use of the dip switch and the Reader utility may be then need to restore the TagTrans back to its default settings.

1.1 Scanner Versions

This Programming Manual is valid for all TagTrans® versions.
FlexiScan® is mechanically different, but same in firmware and software.

1.2 System Requirements

Recommended Operating System:

- Windows 2000 and higher

Recommended any of the following Bluetooth stacks for Windows:

- IVT BlueSoleil www.bluesoleil.com
- Broadcom (Widcom) www.broadcom.com
- Microsoft Windows XP www.microsoft.com
- Toshiba Bluetooth stack http://aps2.toshiba-tro.de/bluetooth

Recommended Bluetooth Adapter

- BT USB Adapter UD100 www.sena.com

1.3 Package List

- TagTrans
- Battery (rechargeable)
- Charger with international plugs
- Hardware and Firmware Configuration Table
- CD-ROM including Software and Manuals
- BT adapter if ordered

1.4 Hardware Preparation

1.4.1 Charging the Battery

Use only the charger provided to charge the battery.

The *charger status LED* will emit a solid orange light while charging, and turns green when the battery is fully charged.

If the TagTrans is attached to the battery during the charging process the *LED on the TagTrans* shines green.

1.4.2 Installing the Battery

Fit the **TagTrans** and the battery unit together by aligning the white points, see the illustration below.

Warning: Incorrect alignment of the TagTrans and battery may result in damage to the casing of either the battery or the TagTrans. Do not use excessive force to close the housings.

2 Power Management

The TagTrans device is a battery powered unit, the battery and the main housing being separate parts. Once the battery is connected to the main housing the TagTrans is powered on and several operating modes are possible.

In order to preserve battery life the unit will go to sleep (Sleep mode) after a period of being idle, the idle period is configurable. The unit can be woken up by inverting the unit to a heads down position and held for <>2 seconds until the units powers on.

In this section these operating modes are described individually.

2.1 Connection Establishment

Once the TagTrans unit has been powered on, by default, the unit tries to establish a Bluetooth connection with a host. There are three possible ways that the device can be powered on.

1. The battery is attached to the main housing.
2. From sleep mode: perform manual waking movement “for at least 2 seconds
3. Full enterprise or idle/standby mode: (by forcing a disconnection from the host)

2.1.1 Status Indicator (LED)

Under normal conditions, if the battery is fully charged, the battery condition indicator LED shines **green** - If this LED shines **orange**, the battery conditions are moderate to low. If the LED flashes **red** the battery should be changed or recharged.
After a **RESET** beep indicates a successful initialization of the TagTrans.

2.1.2 Timer duration

A timer **PwrTmr** is loaded - after a RESET - with a default value of 180s (set **t1**). During this period the TagTrans will attempt to establish a Bluetooth connection to a host. If the timer is zero, the TagTrans changes into **sleep mode**. If a connection has been established the equipment changes into the **full enterprise mode**.
With the “waking movement” the **PwrTmr** is loaded with 60s (set **t2**).

2.1.3 Power Consumption

During the connection process only the BT-module is supplied with power The HF-module remains inactive – power consumption is approx. 50… 150 mA (corresponds about 50 hours of battery life).

2.2 Full enterprise mode

In the full enterprise mode all modules are active.

The Communication between HF-module and host is now open. Each byte from the host is passed directly to the HF-module and/or vice versa i.e. from TagTrans to the host..

Also the so-called „%-instructions “are accepted by TagTrans - see section TagTrans Instructions.

There are two possibilities to establish full enterprise mode:

1. From the **connection establishment**: Host-side connection.
2. From the **standby**: Short manual waking movement or send a character to the TagTrans.

2.2.1 Status Indicator (LED)

The **battery condition indicator LED** shines either **green** or **orange**. A second LED - the **connection indicator LED**, shines **blue** when a connection to a host exists, also when a connection is established a 1 second audible sound is emitted. When data is **transmitted,** the connection indicator will briefly shine **violet**.

2.2.2 Timer duration

When a connection is established an additional timer **RdTmr** is loaded with 20s (set t3). If communication between host and HF-module takes place the **RdTmr** is reset to 20 sec and the **Pwrtmr** is reset to 60 sec.

If a manual waking movement is performed “ the ,**RdTmr** is reset to 20 sec and the **Pwrtmr** is reset to 60 sec

If the timer RdTmr runs down, TagTrans changes into the readiness mode. RdTmr value must always be smaller than PwrTmr.

2.2.3 Power Consumption

In the **full enterprise** the unit power consumption peaks, as all modules are active - approx. **200… 250 mA** (corresponds about 22 hour of battery life).

2.3 Idle / Standby Mode

In this mode the connection to the host is maintained, however the HF-module is deactivated. A feature of this mode is the **fast change into** the **full enterprise mode** the TagTrans uses less power than during the initial connection process. There is one option to get into this mode:

1. The device must have previously been in **full enterprise**: The timer **RdTmr** runs down to zero and the unit’s switches to Idle mode..

2.3.1 Status Indicator (LED)

The **battery status indicator** LED shines **green** or **orange** and the **connecting indicator** LED **flashes blue** in standby mode.

2.3.2 Timer duration

If the timer **PwrTmr** runs down to zero the TagTrans changes into the sleep mode and the connection to the host is broken. Either communication from the host or performing the “waking movement“ changes the TagTrans back into the full enterprise mode.

2.3.3 Power Consumption

By deactivating the HF-module and already having the connection between host and TagTrans established. Standby mode has the smallest power consumption compared to other modes - approx. **40… 100mA** (approximately 72 hours of battery life).

2.4 Sleep Mode

In the sleep mode power consumption is switched off.
There are two options to get into this mode:

1. In **full enterprise** the host can send the TagTrans an instruction „%off “(see section TagTrans Instructions)
2. From Idle/standby mode the timer PwrTmr times out.

2.4.1 Status Indicator (LED)

Both the battery status indicator LED and the connecting indicator LED are extinguished.

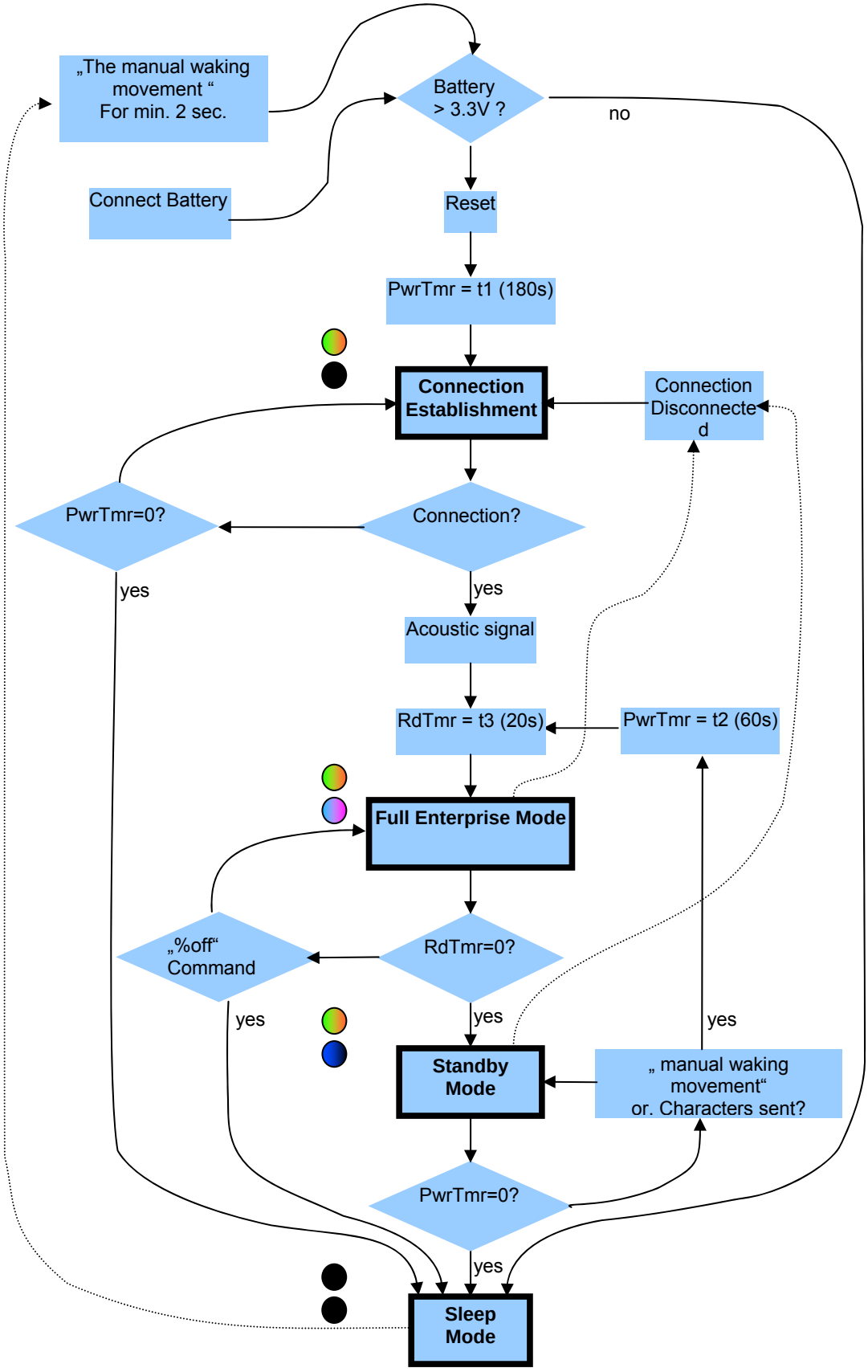
2.4.2 Timer duration.

Until the waking movement is performed“(approx. 2 seconds long), the TagTrans remains in the sleep mode.

2.4.3 Power Consumption

In this mode there is practically no current flow, TagTrans in this mode uses practically no power from the battery. However by the nature of the device there will over time, be power “leakage” and eventually the battery will become discharged and require recharging.

2.5 State diagram



3 TagTrans - Instruction

TagTrans ordinarily routes host-sided incoming characters to the HF / UHF-Module.
Changes and inquiries of the TagTrans unit can be made, in order to do this a simple syntax was introduced.

3.1 General Syntax

The syntax begins with a ' %' - Indication and ends with a 'Return' (0Dh, 0Ah). The indications should be smaller than 254 characters in length.

%command	*[para1 para2...] CrLF* {0Dh, **0Ah**}
e.g. %help	

3.1.1 Command „%help“

Outputs a list of possible instructions.

Input: **%help**	Output instruction list Output: *%TagTrans HW 2.2 SW 0.12.4 HF 3AMS Build Time Apr 29 2010 11:48:13* *%get to ... get timeouts* *toff actual time until power off* *troff actual time until reader off* *t1 timeout power off* *t2 retrigger time power off* *t3 retrigger time reader off* *%set t1 xxxxx ... set timeout power off to xxxxx seconds* *%set t2 xxxxx ... set retrigger time power off to xxxxx seconds* *%set t3 xxxxx ... set retrigger time reader off to xxxxx seconds* *%set factdef [rf/tt/bt] ... set TagTrans to factory default* *%get bat/bth ... get battery level/thresholds* *%set pin xxxx ... set pin (max 16 digits)* *%set name xxxx ... add string to friendly name (max 32 digits)* *%get acon ... get state of auto connect* *%set acon on/off ... set auto connect on/off* *%del pair ... delete pairing* *%set msg on/off ... set message on/off* *%beep volume[0-3] frequency[10Hz] duration[0.1s]* *%blink help ... lists blink commands* *%get br ... get actual baudrate* *%set br xxx ... set baudrate to xxx* *47 -> 9600, 23 -> 19200, 11 -> 38400,* *7 -> 57600, 3 -> 115200* *%ax help ... lists 3axes motion sensor commands* *%off [nowake] ... power down*

3.2 Instruction Set

When the host sends a ' %' charatcer, the normal data flow is interrupted. Each further character is stored in a Buffer (max. 254 characters), also a timer is activated and loaded with approx. 8 seconds. Each subsequent character initializes the timer with 8 seconds. If the timer times out this mode is interrupted and each stored character (inclusive ' %' - character) is transmitted to the HF-module.

If a CrLF (0Dh, 0Ah) is sent, before the timer timeout, the stored character string is treated as an instruction and the timer deactivates.

If the instruction is invalid the buffer will be deleted and the character string is „%syntax error {CrLf}“ is sent to the host.

3.3 Hardware Commands

The firmware supports all hardware combinations, which can be adjusted with the internal command set.

3.3.1 Command „set rf“

This Command is used to select between HF and UHF module.

%set rf 1	Communication HF Module und TagTrans
%set rf 2	Communication UHF Module und TagTrans

3.3.2 Command „set img“

If a barcode Imager (PL4407) is installed, this command activates/deactivates all associated functions.

%set img 1	Activate the Imager functions
%set img 0	Deactivate the Imager functions

3.3.3 Command „set mb“

The „MultiBlock" functionality is with this command activated.

%set mb 1	Activate MultiBlock function
%set mb 0	Deactivate MultiBlock function

3.4 General Commands

3.4.1 Command „reset“

%reset	Reset TagTrans
	Output: %reset\r\n

3.4.2 Command „set br usb“

Especially for TagTrans with USB.

%set br usb	Set USB Baudrate

3.4.3 Command „set/get ID“

%set id	Set TagTrans ID
	Output: %ok\r\n
%get id	Show TagTrans ID
	Output: %reverse 'rv_func'

3.4.4 “Command „set msg on/off”

%set msg on	Enable TagTrans Messages
%set msg off	Disable TagTrans Messages

3.4.5 Command „ver“

%ver	Show whole version
	Output: %TagTrans HW 'ver_hw' SW 'ver_fw' "LF"^"HF"^"UHF" ""^"3AMS" ""^"USB" ""^"AP" Build Time 'build_time'\r\n

3.4.6 Command „ver bl“

%ver bl	Show Boot loader Version
	Output: %"0"^"1"\r\n

3.4.7 Command „ver fw“

%ver fw	Show Firmware Version
	Output: %'ver_fw'\r\n

3.4.8 Command „ver hw“

%ver hw	Show Hardware Version
	Output: %"2.1"^"2.2"\r\n

3.4.9 Command „bootloader“

%boot loader	Reset TagTrans and go to bootloader mode
	Output: %bootloader\r\n

3.4.10 Command „beep“

%beep	Start acoustic signal

3.4.11 Command „blink“

%blink	Start blinking

3.4.12 Command „blink help“

%blink help	List all blink commands

3.4.13 Command „blink set count“

%blink set count	Set the number of blinking

3.4.14 Command „blink set speed“

%blink set speed	Set blinking speed

3.4.15 Command „blink set ths“

%blink set ths	Set threshold between ON and OFF time

3.4.16 Command „blink store count“

%blink store count	Set & store the number of blinking

3.4.17 Command „blink store speed“

%blink store speed	Set & store blinking speed

3.4.18 Command „blink store ths“

%blink store ths	Set & store threshold between ON and OFF time

3.5 Power-Management Commands

TagTrans offers the possibility to adapt power consumption based on the respective application and physical requirements. This section describes all the required instructions to enable various power management options.

3.5.1 Command „set t1/t2/t3“

This instruction changes the values, with which the timers **PwrTmr** and **RdTmr** are loaded, see section Power Management.

%set t1 *x*	Set Power Off Timeout on start-up, x is indicated in decimal and in seconds
%set t2 *x*	Set Power Off Timeout, x is indicated in decimal and in seconds
%set t3 *x*	Set Reader Off Timeout, x is indicated in decimal and in seconds

3.5.2 Command „get to“

Return the current state of the individual timers and the appropriate values back.

%get to	Show time out state
	Output: toff: 'Off Timer' troff: 'Reader Off Timer' t1: 't1' t2: 't2' t3: 't3'\r\n *toff: x troff: y t1: a t2: b t3: c* *x* time up to Sleep Mode - PwrTmr *y* time up to readiness mode - RdTmr *a,b,c* the appropriate value t1, t2, t3.

3.5.3 Command „get bat“

Return the current state from the battery.

%get bat	Show battery status
	Output: *%bat 'vbat'\r\n* The battery value is calculated then, as follows: U.bat = *x* * 0,0245 [V]

3.5.4 Command „get bth“

Return the battery thresholds.

%get bth	Show battery thresholds
	Output: *%bth* 'ths1' 'ths2' 'ths3'\r\n

3.5.5 Command „off“

TagTrans changes immediately into sleep mode, the connection to the host is separated and TagTrans can only manually “wake up“.

%off	
	Output: *%power off\r\n*

3.5.6 Command „off nowake“

Power off TagTrans without WakeUp.

%off nowake	
	Output: *%power off\r\n*

3.5.7 Command „set demo“

This command was introduced for the purpose of demonstration of the TagTrans with no need for a host device to be present. The module connection establishment is void - see section Power Management. Nevertheless TagTrans tries to get a connection with a host in the background. As long as no real connection exists, a connection is simulated to the TagTrans.

%set demo 1	Activate the Demo-Mode
%set demo 0	Deactivate the Demo-Mode

3.6 Bluetooth Commands

The Bluetooth module provides the connection between the host and TagTrans. It works independently and can be configured only over a direct connection to TagTrans.

The coding for example can only be changed, if the Bluetooth connection to a host is established, once a connection is established it is possible to send the appropriate instructions to the TagTrans.

Any Bluetooth configuration changes remain until either changed by the user or the TagTrans device is reset back to the factory settings at which point all changes to the Bluetooth module will be lost.

3.6.1 Command „set name“

This command allows the user to specify the name the TagTrans will recognised as, when, TagTrans is registered in a Bluetooth network.

%set name *str*	*str* is a character string, which is transmitted to the Bluetooth module

If a name is not required a “Return” is sent down as the first indication instead of the character string

3.6.2 Command „set pin“

The connection between host and TagTrans is secured with a PIN number, which is defaulted to “1234“. The following command allows the user to change the PIN

%set pin *str*	str is a character string, which is transmitted to the Bluetooth module

If a PIN number is not required or desirable, a ' CrLF' as the first character can be sent instead of a string and the user would not be prompted for the PIN during the Bluetooth connection process.

3.6.3 Command „set acon on / set acon off / get acon“

As soon as the TagTrans is in the enterprise mode, the Bluetooth module tries, as a standard, to develop a connection sequentially. This command sets the “automatic connection”. It can be activated and/or deactivated.

%set acon on	Activates the automatic connection
%set acon off	Deactivates the automatic connection
%get acon	Returns the current condition

3.6.4 Command „sendbt“

%sendbt	Send iWRAP commands to WT11 Module
	Output: %ok\r\n

3.6.5 Command „del pair“

If TagTrans is already connected with a host, the two are “paired”, i.e. that the addresses of the partner are stored. In order to break this paired connection, the following instruction can be used:

%del pair	Breaks the paired connection

3.6.6 Command „set factdef bt“

%set factdef bt	Set BT Module to factory default

3.7 HF und UHF Module Command

Communication between HF-module and/or UHF module and TagTrans takes place serially. Sometimes it is necessary to adapt the communication settings.

3.7.1 Command „set/get br“

This command is used to setup the Baud rate of the TagTrans.

%set br *x*	*x* 3…115200 *x* 7…57600 *x* 11…38400 *x* 23…19200 *x* 47…9600
%get br	Returns the actual value Output: %br x

3.7.2 Command „get rf“

%get rf	Show Reader Device
	Output: %Reader = "MICROENGINE"^"UHF"^"IMAGER"\r\n

3.7.3 Command „poll [on/off]“ (only for UHF Module)

In order to ensure a continuous inquiry of the serial number of the UHF tags, TagTrans sends a polling command to the UHF module. Since in the readiness enterprise mode the UHF module is not supplied with power, this command must be sent with each start of the full enterprise. In order to automate this procedure, this command allows optional selection of the parameters to be turned on or off.

%poll	Send Polling command
%poll on	Enable Polling Transponder on start-up
%poll off	Disable Polling Transponder on start-up Output: %OK\r\n Command was successfully dispatched %poll 1\r\n %poll 0\r\n *%no response* UHF-Module does not response

3.7.4 Command „set factdef rf“

%set factdef rf	Set RF Reader to factory default

3.7.5 Command „set factdef tt“

%set factdef tt	Restore TagTrans to factory default

3.7.6 Command „get/set reverse“

%get reverse	Show current Reverse function
%set reverse	Set Reverse Functions for UID

3.7.7 Command „mb [cmd]/[set/get auto/key/tag]“ / „md [cmd]“ / „mt [cmd]“

„mb“ stands for **MultiBlock** and serves among other things the formatted output of tag memory content.

The parameter **„[cmd]“** is the sequence (or chain of command), with which TagTrans e.g. has to read out a Tag - see section MultiBlock

It is possible to automate this sequence. Here the HF-module is placed into the so-called „continuous mode “and the amount of same Tag UIDs are counted. Reaches this a value of 3, than the stored sequence will be run. You should note the RF module in this mode is not **host-laterally controllable** - also no characters are routed directly from HF module to the host.

It is necessary for some commands within the sequence to send some parameters to the TagTrans for the corresponding Tags. Parameters like the **length of a block** in byte(s) and / or **the first and the last block**.

Specially created sequences can use the "md..." command (**Debug MultiBlock**) to check their correctness.

The measurement or issue of the required times for a series takes place using "**mt**..." command (**MultiBlock Timing**).

%mb	Starts the **saved** sequence
%mb [cmd]	Execute MB command, Starts the **transferred** sequence Output: *{according to the sequence}*
%md	Starts the **saved** sequence and outputs **Debugging information**
%md [cmd]	Execute MB command and debug, Starts the **transferred** sequence and outputs **Debugging information** Output: *Pos func id in/out if level sub* *….* *exit code: ddd @pos: hh. t.all: t.tts t.calc: t.tts* *ddd* – failure code – see table *hh* – last sequence position t.all *t.tt* – entire expenditure of time (inclusive Output) in seconds t.calc *t.tt* – pure computing/waiting period in seconds
%mt	Starts the **saved** sequence and outputs **timing information** and/or a number of various communication errors
%mt [cmd]	Starts the **transferred** sequence and outputs **timing-Information** and/or a number of various communication errors Output: *{according the sequence}* *t.all: t.tts Err:?dd;Cdd;Fdd;Idd;Ndd;Odd;Rdd;Xdd;* *dd* is the number of errors in decimal *t.tt* – entire expenditure of time (inclusive Output) in seconds
%mb set key	Store last MB command as MB key, Saves the last executed sequence No Output
%mb get key	Show MB key, Outputs the saved sequence Output: *[cmd]*
%mb set tag *ll ss ee*	Set Transponder parameters, Transfers TagTrans the parameter for the corresponding Tags *ll* – the length of a block in byte in decimal *ss* – the first block in decimal (or hexadecimal with Hss) *ee* – the last block in decimal (or hexadecimal with Hee) No Output.
%mb get tag	Show Transponder parameters, Outputs the saved parameter Output: *%mb len:ll first:ss last:ee*
%mb set auto *x*	Disable/Enable MB Auto execute, Automatically sequence execution 0=off / 1=on No Output
%mb get auto	Show MB Auto execute status Output: %mb auto=x 0=off / 1=on
%ms	Store MB command as key 0=off / 1=on No Output
%mb set cal	Disable/Enable MB Antenna calibration mode No Output

3.8 Imager Command

To ease the integration of the Imager into an application, TagTrans makes some functions and instructions available.

3.8.1 Command „set img“

If a barcode Imager (PL4407) is installed, this command activates/deactivates all associated functions.

%set img 1	Activate the Imager functions
%set img 0	Deactivate the Imager functions

3.8.2 Command „ic help“

Output a list of possible instructions.

%ic help	Output instruction list Output: *%ic [init/off] ..init imager / turn off imager* *%ic wakeup [on/off/trig] ..set/trigger wakeup* *%ic set/get prefix/suffix ..use ascii or \xx(hex), \r, \n, * *%ic hc start ..direct communication mode* *%ic hc set/get to [sec] ..set/get hc-timeout (2s..200s)* *%ic hc init [on/off] ..set hc on start-up* *%ic toggle [/off/rf/img] ..start/toggle* *%ic toggle set/get to [dsec] ..toggle timeout (2ds..100ds)* *%ic toggle init [on/off] ..toggle mode on start up* *%ic msg [on/off] ..set imager message on/off* *%ic get pin ..get hardware pin* *%ic 1 ..start dec* *%ic 2 ..stop dec* *%ic 3 ..aim off* *%ic 4 ..aim on* *%ic 5 ..led off* *%ic 6 ..led on* *%ic 7 ..set default* *%ic 8 pp aa [bb cc] (hex)..param send* *%ic 9 pp aa [bb cc] (hex)..param send&store* *%ic 10 pp (hex)..param request*

3.8.3 Command „ic init“

Initializes the Imager-Module. This happens automatically with each Reset procedure.

%ic init	Reset Imager Output: *%init OK*

3.8.4 Command „ic off“

The Imager module is no longer supplied with power. However the Imager commands are still accessible.

%ic off	Deactivates Imager Module
	No Output

3.8.5 Command „ic *x*“

In order to save the final user communication with the Imager (SSI protocol), the most important instructions are simplified with this command.

%*ic* 0	Starts Decode session
%ic 1	Stops Decode session
%ic 2	Activate cross hair
%ic 3	Deactivate cross hair
%ic 4	Activate LED (if available).
%ic 5	Deactivate LED
%ic 6	Activate Sleep Mode

All settings of the Imager module can be questioned or changed with the following command. In the SE4407 documentation all parameters and their values are described in detail.

%ic 8	Change of a parameter (in hexadecimal) (it is not stored!)
%ic 10	Queries of a parameter (in hexadecimal)

The following commands should **not** be exported continuously (e.g., in a loop):

%ic 7	Resets settings of the imager module.
%ic 9	Change and store of a parameter (in hexadecimal)
	Output: „ic 10 xx“ *%para=xx val=yy* *xx* – Parameter *yy* – related value (see SE4407 Documentation)

Error list Output

%ic error 'ic_err'\r\n	
%ic error **1** \r\n	timeout
%ic error **2** \r\n	general failure
%ic error **3** \r\n	no ack
%ic error **4** \r\n	imager not enable
%ic error **5** \r\n	imager not ready or not powered
%ic error **6** \r\n	syntax error
%ic error **7** \r\n	receive buffer error
%ic error **8** \r\n	checksum error
%ic error **9** \r\n	wrong parameter

3.8.5.1 Example

„%ic 8 E3 01“	… MicroPDF417 – Recognition activate (not store)
„%ic 0“	… Start the decode session (according to setting Imager module)
„%ic 1“	… Stop the decode session
„%ic 3“	… Deactivate crosshair
„%ic 6“	… Activate Sleep Mode

3.8.6 Command „ic wakeup on/ ic wakeup off/ ic wakeup trig“

This command orders the Wakeup of the Imager module.

„ic wakeup on“	Wakeup on continuously, Deactivate automatically Sleep Mode (attention: current consumption)
„ic wakeup off“	Wakeup off continuously, Activates Sleep Mode
„ic wakeup trig“	Trigger wakeup pin, Activates shortly the Imager Module
	No Output

3.8.7 Command „ic set prefix/ ic set suffix/ ic get prefix /ic get suffix “

After a successful decode process the data will transfer directly into ASCII. To make such a decode process recognizable, prefix or suffix-sign chains can be added.

%ic set prefix *text*	Set Prefix change.
%ic get prefix *text*	Get Prefix output.
%ic set suffix *text*	Set Suffix change.
%ic get suffix *text*	Get Suffix output.
	Output: With parameter „get …“ outputs the respective sign chain

Note:
Special signs can be indicated with a Backslash.
Recognised signs are: "\r", "\n", "\\", "\00" the number must be two digits and in Hexadecimal.
The maximum length is: 10 signs.

3.8.7.1 Example

„%ic set prefix \FFdecoder:“	…set prefix text -> {0xFF}decoder:
„%ic set postfix end\r\n”	…set suffix text -> end{CrLf}

3.8.8 Command „ic hc start / ic hc init on / ic hc init off / ic hc set to / ic hc get to“

With this command the direct (binary) communication is allowed with the Imager module. After an adjustable Time-out the direct communication is broken off.

%ic hc start	Start binary communication
%ic hc init on	Start binary communication on start-up Automatic execution of the start command after a Wakeup.
%ic hc init off	Do not start binary communication on start-up No automatically execution of the start command after a Wakeup.
%ic hc set to *x*	setting the timeout = *x* (in seconds; max 100) Set time out for binary communication
%ic hc get to	Query of the timeout Get time out for binary communication
%ic hc to off	Activate time out for binary communication
%ic hc to on	Deactivate time out for binary communication
	Output: „%ic hc get to“: *%hc timeout: x s* *x* – time in seconds

3.8.9 Command „ic toggle/ ic toggle init on/ init off / set to/ get to/ img / rf/ off“

This command, after an adjustable timeout, puts the Imager module in the Sleep mode or begins the decode process. As soon as a sign is transferred, this time-out is reset. If a successful decode process is recognised by the Imager module, the result is transferred, like the %ic 0 command, to the Host and the module is put immediately in the Sleep mode.

%ic toggle	activates the Toggle-Mode / changes between „Sleep-Mode“ and „Active-Mode“
%ic toggle img	change to Active-Mode, Toggle to Imager Decoding
%ic toggle rf	change to Sleep-Mode and Toggle to RF
%ic toggle off	deactivates the toggle-mode
%ic toggle set to *x*	setting the timeout = *x* (in 0.1s) for toggle mode
%ic toggle get to	outputs the timeout (in 0.1s) for toggle mode
%ic toggle init on	toggle-mode after a Wakeup - start-up
%ic toggle init off	no Toggle-Mode after a Wakeup, turn toggle mode off
	Output: Output like with the command „ic 0“. At the parameter „get to“ the timeout is outputted in 0.1s.

3.8.9.1 Example

„%ic toggle set to 8“	setting the timeout in 0.8s
“%ic toggle”	activates the Toggle-Mode
„%ic toggle rf“	changes to Sleep-Mode
„%ic toggle off“	deactivates the Toggle-Mode

3.8.10 Command „ic msg on/off“

Activates / Deactivates Imager error messages.

%ic msg on	error message activated.
%ic msg off	error message deactivated.
	no Output.

3.8.11 Command „ic get pin“

Displays the status of every pin of the imager module or the TagTrans.

%ic get pin	Show hardware pin states
	Output: %ic vcc=x %ic aim/wakeup=x %ic pwrdwn=x %ic decode=x %ic trigger=x %ic rts=x %ic cts=x %ic download=x x – binary value (0 or 1)

3.9 Motion Sensor Commands

The 3-axes motion sensor determines movements or accelerations in every direction.

Basically 2 movement types can be detected automatically:

1. **"free fall":** e.g. **Pivot the TagTrans** device from below to above or from top to bottom.
2. **"Punch or kick":** (simple or double) a **double "tap" of the TagTrans** with the fingers within half a second or. e.g. **360 ° rotation around the longitudinal axis** within a tenth second

These movements may be combined with one of the following (events):

- **"Wake up the TagTrans"**
- **"Reset all timer"**
- **"TagTrans in sleep mode enabling"**
- **"Run of stored MultiBlock command"**

There are 2 different configuration modes:

"Active mode": scenario: a movement is detected and executes an action (event).

"WakeUp mode": scenario: a movement is detected and TagTrans being awakened from sleep mode.

Note: All thresholds refer to the x-times acceleration of gravity g and can be between 0 and 15. A value of 1 corresponds to roughly 0.5 * g with a relative error of 15%.

3.9.1 Command „axr"

The sensor includes an 8 – bit register, this will be used to control its behaviour or to get acceleration data.
The register address consists of 7 bit.

With the command „%axr" the register can be read out.

%axr	Read out Register
	Output: 'reg_adress'->'reg_value'\r\n

3.9.2 Command „axw"

With the command „%axw" on the register can be written. (Attention: not recommended for users!)

%axw	Write on register
	Output: 'reg_adress'<-'reg_value'\r\n
	Failure Output: %ax error\r\n %ax error com\r\n

3.9.3 Command „ax init"

The sensor will be initialized.

%ax init	Initialization of the sensor
	Output: %ax ok\r\n

3.9.4 Command „ax stop"

The sensor will be switched off – Sleep Mode. (to reduce power consumption)

%ax stop	Sensor Stop
	Output: %ax ok\r\n

3.9.5 Command „ax help”

List all “3-axes motion sensor” commands.

%ax help	List sensor commands
	Output: 3axis motion sensor command list %ax config wakeup/active ... select wakeup/active config %ax get config ... show configuration %ax set event C X N ... select event n after x motion c=0 -> channel 1 c=1 -> channel 2 x=0 -> no n=0 -> no event/wake up x=1 -> free fall n=1 -> turn off TagTrans x=2 -> click n=2 -> retrigger timer n=3 -> execute MultiBlock %ax filter [val] ... set high-pass filter [0..4] Free fall detect settings... %ax ff ths [val] ... set threshold 2..15 %ax ff [x/y/z][0/</>] ... start event if 0 -> ignore axe x/y/z > -> active if accel. of axe x/y/z is greater than ths < -> active if accel. of axe x/y/z is smaller than ths %ax ff dur [csec] ... set free fall duration 0..255csec Click detect settings... %ax clk [x/y/z][0/1/2/3] ... none/single/double/both clicks %ax clk ths [x/y/z] [val] ... set threshold 2..15 %ax clk tlim [msec] ... set time limit for one click 0..127msec %ax clk tlat [msec] ... set latency after click 0..255msec %ax clk twin [msec] ... set time window for two clicks 0..255msec %ax chk ... starts testing mode

3.9.6 Command „ax config wakeup”

With this command the TagTrans will be transferred in the WakeUp Mode.

%ax config wakeup	Configuration Wakeup Mode
	No Output

3.9.7 Command „ax config active”

With this command the TagTrans will be transferred in the active – configuration mode – default after every new start.

%ax config active	Configuration Active Mode
	No Output

3.9.8 Command „ax get config”

Shows the momentary configuration mode and list the corresponding configuration.

%ax get config	
	Output: Configuration mode: active Channel 1: retrigger timer if click Channel 2: no event/wake up if none High-pass filter cut-off freq.: 2Hz Free fall settings: Threshold: 2 Duration: 10 csec Axes: X> Y> Z> Click settings: Threshold: X=2 Y=2 Z=2 Time limit: 32 msec Time latency: 100 msec Time window: 100 msec Axes: X both, Y both, Z both

3.9.8.1 Example

Input:	
%ax config wakeup **%ax get config**	
	Output: Configuration mode: wakeup Channel 1: no event/wake up if click Channel 2: no event/wake up if none High-pass filter cut-off freq.: 2Hz Free fall settings: Threshold: 2 Duration: 10 csec Axes: X> Y> Z> Click settings: Threshold: X=2 Y=2 Z=2 Time limit: 32 msec Time latency: 100 msec Time window: 100 msec Axes: X both, Y both, Z both

3.9.9 Command „ax set event”

With this command automatic recognized movement are combined with any operation. Two different movements can recognize simultaneously. (Channel 1 / Channel 2)

%ax set event *a b c*

a … **0 or 1** -> 1. or 2. Movement recognition
b … **0 to 2** -> 0 = no movement, 1 = free fall, 2 = punch/kick
c … **0 to 3** -> 0 = wake up, 1 = timer reset, 2 = TagTrans switch off, 3 = MultiBlock start

%ax set event	set Event Trigger (event <-> trigger)
	No Output

3.9.10 Command „ax filter”

The High pass filter will be set to filter out static acceleration. (e.g. Standard acceleration of gravity).

%ax set event *a*

b … **0 to 4** -> 0 = static, 1 = 0,25 Hz, 2 = 0,5 Hz, 3 = 1 Hz, 4 = 2 Hz

%ax filter	Set High pass filter
	No Output
	Failure Output: %ax error\r\n

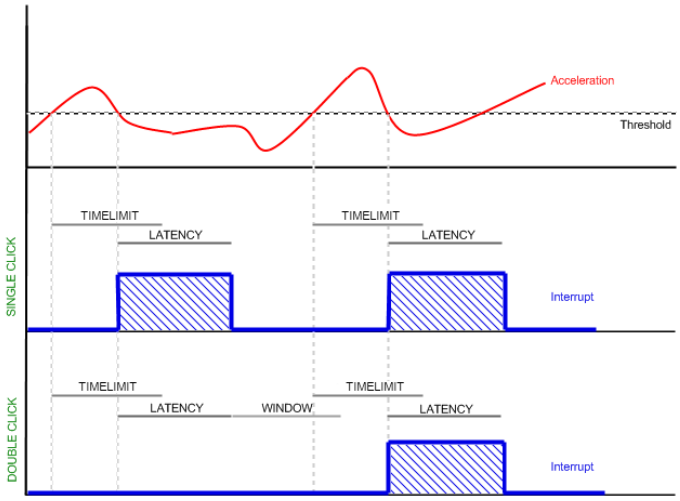
3.9.11 Command „ ax ff / ax ff ths / ax ff dur”

For the free fall axis, threshold and duration have to be set.

%ax ff	Axis – set the axis for free fall (greater or smaller than the threshold)
%ax ff ths	Threshold – set the threshold of the free fall
%ax ff dur	Duration – Set the duration within or without the threshold of the free fall.
	No Output
	Failure Output: %ax error\r\n

3.9.12 Command „ax clk / ax clk ths / ax clk tlat / ax clk tlim / ax clk twin“

The sensor can distinguish between two different sorts of punch / kick. – single or double.



The illustration shows one double kick and one single kick.

%ax clk	Set axis x, y or z
%ax clk ths	Set the threshold for x/y/z axis – every axis can defined individually
%ax clk tlat	Delay the action beyond the motion detection
%ax clk tlim	Set time limit for the punch /kick detection
%ax clk twin	Set maximum permissible time window for dual punch/kick
	No Output
	Failure Output: %ax error\r\n

3.9.13 Command „ax chk”

Switch to test mode – to check primarily the set motion detection. A deep signal is heard for the first motion detection and a high level for the second.
Additionally, all 120 milliseconds the current acceleration of the axes are showed. You can here e.g. the data emitted through a terminal program save to a file and process them further.

%ax chk	Change to test mode
	Output: %'Output for 3axis'\r

3.9.13.1 Example

%ax chk	
	Output: %3ax testing mode %type 'h' for help

3.9.14 Command „ax set default”

Setting the sensor to factory default.

%ax set default	Set Motion Sensor to factory default
	Output: %ax ok\r\n

3.10 MultiBlock

In this section the functionality of the multi block sequence and the possible commands within a sequence are explained.

3.10.1 Decoding of the sequence

- The command detection occurs character by character.
- If a command is recognised, this is executed immediately and the output occurs immediately.
- With the character **"."** a sequence is stopped.
- A **{return} control character** breaks the sequence to avoid an overrun.
- Every error which originates during the command detection leads to an abnormal termination. An exception displays the "If instruction" - however only if no syntax error is given.
- The maximum length of a sequence amounts to 250 characters.
- It is case sensitive.

3.10.2 Memory Management

- Every **block** which is **read or is described** is stored in a **static memory**.
- If a saved block is read, this is read out by the memory.
- At each new start of the sequence, the memory is deleted.

3.10.3 Communication with HF-Module

- At every start of a sequence, it is "held on" the first received UID of a tag.
- Therefore the false reading or writing of another tag is avoided. If required, a command 'Release of tag' offers the possibility to release the Tag. Then a new UID would be possible in the same sequence.
- If another UID is recognised during a sequence, this will cause a communication error. Therefore any false reading or writing of another tag is avoided. If needed, a command 'Release Tag' offers the possibility "to release" the Tag. Then is would be possible to read a new UID in the same sequence.
- If a block should be partially described, this is read before. (R&W)
- Some communication errors lead to the immediate abnormal termination, other communication errors will permit repeated attempts.
- The communication occurs in ASCII characters.

3.10.4 Instruction List

The table shows all the possible commands which can be used **within a sequence**.

Command	Command Name	Explanation	Section	Parameter	Condition	Textmemory
"{text}"	Text	Displays {text}	3.10.4.1	-	Yes	-
U	UID	Passes the actual UID of the tag	3.10.4.2	Yes	Yes	*Yes*
R	Read	Reads the Tag block by block	3.10.4.3	Yes	Yes	Yes
W	Write	Writes on the Tag block by block	3.10.4.4	Yes	Yes	-
PR	Pointer Read	Reads the Tag byte by byte	3.10.4.5	Yes	Yes	Yes
PW	Pointer Write	Writes on the Tag byte by byte	3.10.4.6	Yes	Yes	-
T	Trigger	Resets the reader time-out counter	3.10.4.7	-	-	-
Q	Release Tag	Forces the removing from the field	3.10.4.8	-	Yes	-
b	Beep	Enable an acoustic signal	3.10.4.9	Yes	-	-
I	Imager Decode	Sends to the Imager a Decode command.	3.10.4.10	-	Yes	Yes
? : / ;	If, Then, Else, End If	Checks a condition: [...]?{condition}:{true}/{false};[...]	3.10.4.11	-	-	-
.	End	Quits a sequence.	3.10.4.12	-	-	-

3.10.4.1 Text – Command

The text command allows the sending of character strings to the host. Special ASCII character can be passed to the host with a backslash '\' and a following number.

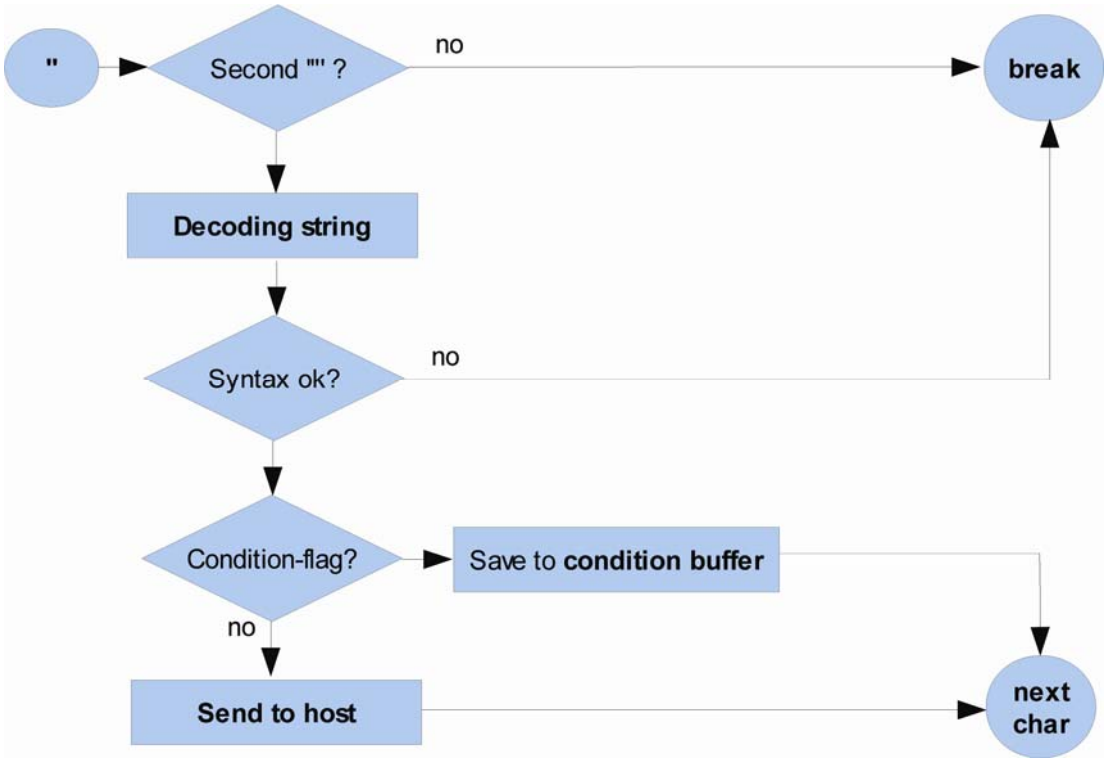
3.10.4.1.1 Syntax

"{text}"

3.10.4.1.2 Possible special characters

\Hxx	xx = Hexadecimal, 00..FF
\xxx	xxx = Decimal number, 000..255
\r	Return
\n	new line
\t	Tabulator
\"	quotation mark
\\	Backslash

3.10.4.1.3 State diagram



3.10.4.2 UID – Command

Sends or saves the UID of a tag. The UID is displayed by the HF module as a Nibble.

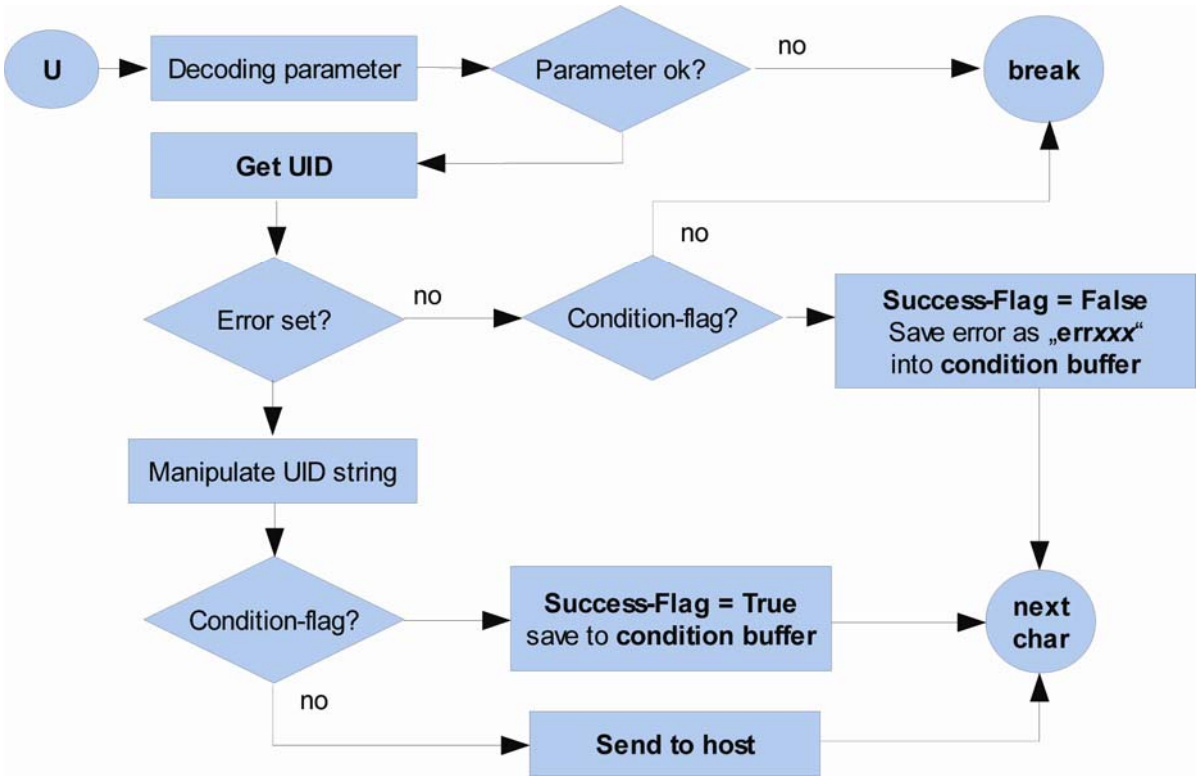
3.10.4.2.1 Syntax

U[]	Complete UID
U[a]	The *x*. sign from the UID (from 0 away)
U[a-b]	The *x*. to *y*. sign from the UID (from 0 away)
U[x]	Output is stored in the text memory

3.10.4.2.2 Description

1. Parameter is checked for syntax and validity and is saved.
2. UID is read, if it has not been read already.
3. If a communication error appears (e.g., no tag in the field) there are 2 possibilities:
   1. If the command is within a condition the error code is saved in the form of "errxxx" as a character string (and xxx is like the error code). The Success flag it is set to "False".
   2. If the command is outside a condition, the sequence is exited.
4. The character string UID is manipulated according to the parameter.
5. If the command UID is within a condition, the character string is saved and the Success flag is set to "True".
6. If the command UID is outside a condition, the character string is displayed directly.

3.10.4.2.3 State diagram



3.10.4.3 Reading block by block

Sends or saves one or several blocks of a tag and the **output** can be changed to suit the application requirements..

3.10.4.3.1 Syntax

Output in Byte	Output in Nibble	Output in decimal (16 Bit)
R[]	R{}	R()
R[a]	R{a}	R(a)
R[a-b]	R{a-b}	R(a-b)
R[a-b,w,x,y,z]	R{a-b,w,x,y,z}	R(a-b,w,x)
R[a-b,“{text}“]	R{a-b,“{text}“}	R(a-b,“{text}“)
R[a-b,w,x,y,z,“{text}“]	R{a-b,w,x,y,z,“{text}“}	R(a-b,w,x,“{text}“)

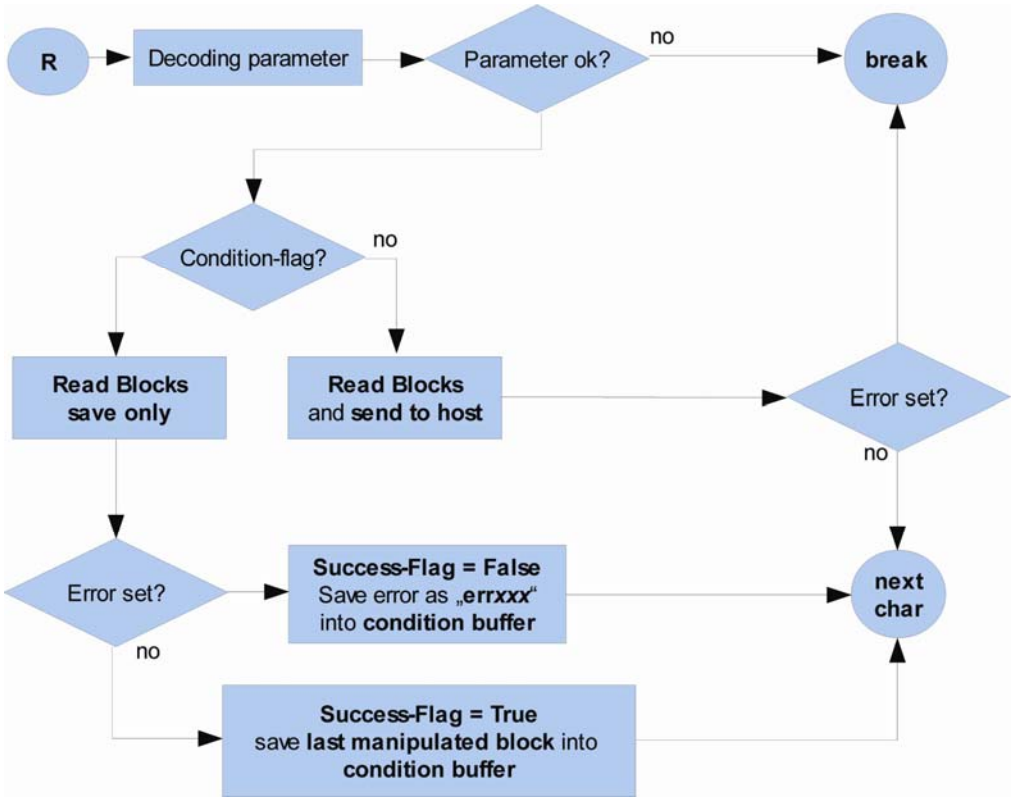
3.10.4.3.2 Parameter Description

Parameter	Explanation
Empty	Reading the 0. block
A	Reading the **a**. block
a-b	Reading the **a.** up to **b.** block
w	**w.** Byte displayed **first**; **if not specified:** display **all Bytes** (the first two Bytes as decimal)
x	**x.** Byte displayed **second**; (with Decimal treat **x**. Byte as a **Low byte** and **w**. Byte as a **high byte**) if not specified: displayed **w**. byte. (with Decimal treat **w**. Byte as a **Low byte**)
y	**y.** Byte displayed **third**; if not specified: displayed **w**. up to **x**. Byte.
z	**z.** Byte displayed **fourth**; if not specified: displayed **w**. up to **y**. Byte.
"*{text}*"	Separates every block with the character string **{text}** The same rules similar to the text command are valid - see section Text - Command In addition, the **text position** of the single manipulated blocks and the **block number** can be displayed. **&T** ... mark text positions **&C** or **&C%%** mark the position of the block number in hexadecimal **&c** or **&c%%** or **&c%%%** mark the position of block number in decimal **&&** corresponds to the character **&** e.g.: "block number: &C%% && content: &T \r\n" → block number: 0A & content: 00FF01FE

3.10.4.3.3 Description

1. Parameter is checked for syntax and validity, than saved.
2. If the command is **within a condition** the function **'Read block'** is called **without output**.
   If the command is **outside a condition** the function is called **with output**.
3. If a communication error occurs (e.g., error excess) there are 2 possibilities:
   1. If the command is within a condition, the error code is saved in the form of "errxxx", where xxx is the error code, as a character string. The **Success flag** is set to "**False**".
   2. If the command is beyond a condition, the sequence is exited.
4. **The last read block is preserved in the buffer.** This is already manipulated according to the parameter, however, **none** is included **"{text}" separation**.
5. If the „Read block command is within a condition, the character string is saved in the buffer and the **Success flag** is set to **"True"**.

3.10.4.3.4 State diagram



3.10.4.4 Writing block by block

Describes and **saves** one or several blocks on one Tag and the kind of the **input** can be different.

3.10.4.4.1 Syntax

Input in Byte	**Input in Nibble**	**Input in decimal** (16 Bit)
W[]	W{}	
W[*a*]	W{*a*}	
W[*a*-*b*]	W{*a*-*b*}	
		W(*a*-*b*,*w*,*x*)
W[*a*-*b*,“*{text}*“]	W{*a*-*b*,“*{text}*“}	W(*a*-*b*,“*{text}*“)
W[*a*-*b*,*w*,“*{text}*“]	W{*a*-*b*,*w*,“*{text}*“}	W(*a*-*b*,*w*,*x*,“*{text}*“)

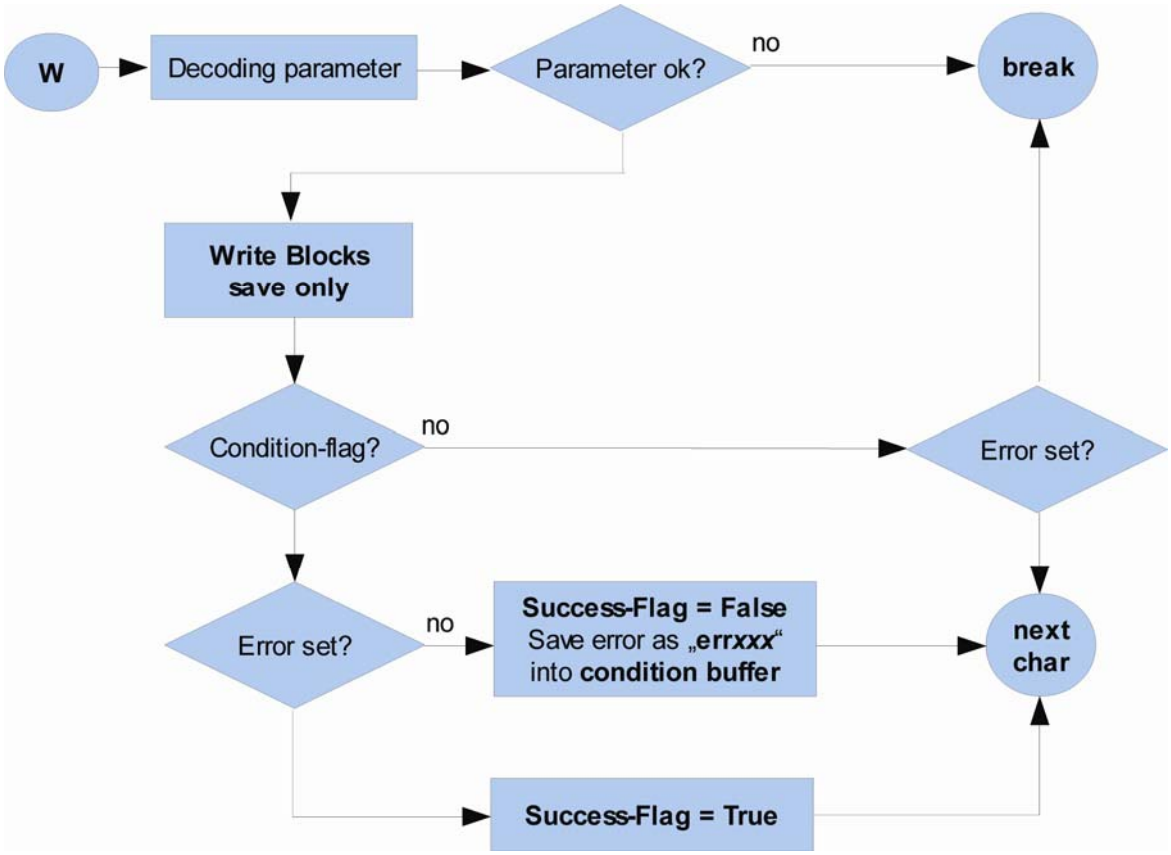
3.10.4.4.2 Parameter Description

Parameter	Explanation
Empty	Writing the 0. Blocks
a	Writing the **a.** Blocks
a-b	Writing the **a.** to **b.** Blocks
w	**w.** Byte to use on the **first position** (decimal: High-Byte) **if not specified:** use the **first byte** (decimal: Low-Byte)
x	only with decimal Input: **x.** Byte to use with the **second position**; (Low-Byte) **if not specified:** use **only w.** Byte as a position (Low Byte)
"*{text}*"	The character string **{text}** contains the information to be described. With *Byte-Input* [ ] : The same rules similar to the text command are valid. **– see section** Text – Command Max. 4 Bytes (block length from one Tag) with *Nibble-Input*{ } : The characters 'A' to 'F' must be capitalised. Max. 8 bytes or the length must be an even number. The number may amount to max. 65535. Hexadecimal are marked with an 'H' before: max. HFFFF **If not specified:** If the memory has already read in the desired block, the information is read by the memory – only with Byte and Nibble-Input. Only sensibly together with **„release of tag".**

3.10.4.4.3 Description

1. Parameter is checked for syntax and validity and is saved.
2. If a communication error appears (e.g., error excess) there are 2 possibilities:
    1. Is the command within a condition, the error code is saved in form from "errxxx" and xxx is like the error code as a character string. **Success flag** is put on "**False**".
    2. If the command is beyond a condition, the sequence is exited.
3. Is the „Write block" command within a condition, the **success flag** is put on "**True**".

3.10.4.4.4 State diagram



3.10.4.5 Byte-wise reading

Allows the byte wise readout of the memory contents of a tag. The maximum memory is calculated from the block length in bytes and from the number of the readable or writeable blocks. These parameters are adjusted with the command **„%mb set tag“**, see section Command „mb“
On this occasion, it should be considered that the real reading of the tag is still occurring in a block wise fashion and is also saved accordingly. However, the output block is treated independently with the advantage of being able to generate a **self-defined memory construction**.

3.10.4.5.1 Syntax

Output in Byte	**Output in Nibble**	**Output in Decimal** (16 Bit)
PR[]	PR{}	PR()
PR[*a*]	PR{*a*}	PR(*a*)
PR[*a*-*b*]	PR{*a*-*b*}	PR(*a*-*b*)

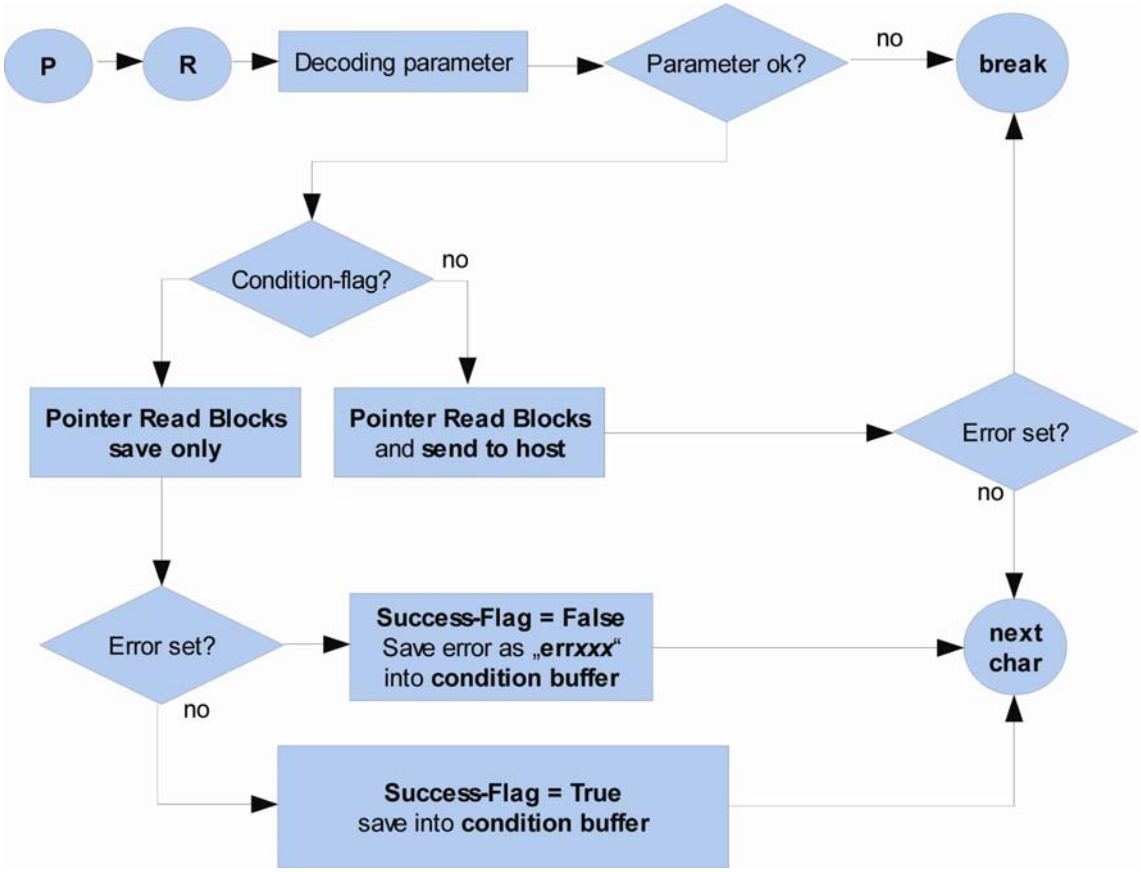
3.10.4.5.2 Parameter Description

Parameter	**Description**
empty	Read out all (see „%mb get tag“ section Command „mb“
a	Read the **a.** Bytes (by Decimal handle the **a.** Byte as **Low-Byte**)
a-b	Reading the **a.** to **b.** Bytes (by Decimal handle the **a.** Byte as **Low-Byte** and the **b.** Byte as **High-Byte**)

3.10.4.5.3 Description

1. Parameter is checked for syntax and validity and is saved.
2. If the command is within a condition the function **'Pointer Read'** is called without output. If it is outside a condition the function is called with output.
3. If a communication error occurs (e.g., error excess) there are 2 possibilities:
    1. If the command is within a condition, the error code is saved in form from "errxxx" where xxx is the error code as a string. **Success flag** is set to "**False**".
    2. If the command is outside a condition, the sequence is exited.
4. If the „**Pointer Read"** is within a condition, the character string is saved in the buffer and the **Success flag** is set to "**True**".

3.10.4.5.4 State diagram



3.10.4.6 Byte wise writing

Allows the byte wise writing on the memory content of a tag. The maximum memory is calculated from the block length in byte and from the number of the readable or writeable blocks. These parameters are with the command „%mb set tag" and are adjustable – see section Text – Command
It should be noted that real writing on the tag is still done in a block wise manner and is also saved accordingly. However, the input is treated block independently which has the advantage of allowing **self-defined memory construction**.

3.10.4.6.1 Syntax

Input in Byte	**Input in Nibble**	**Input in Decimal (16 Bit)**
PW[]	*PW{}*	
PW["{text}"]	*PW{"{text}"}*	*PW("{text}")*
PW[a,"{text}"]	*PW{a,"{text}"}*	*PW(a,"{text}")*

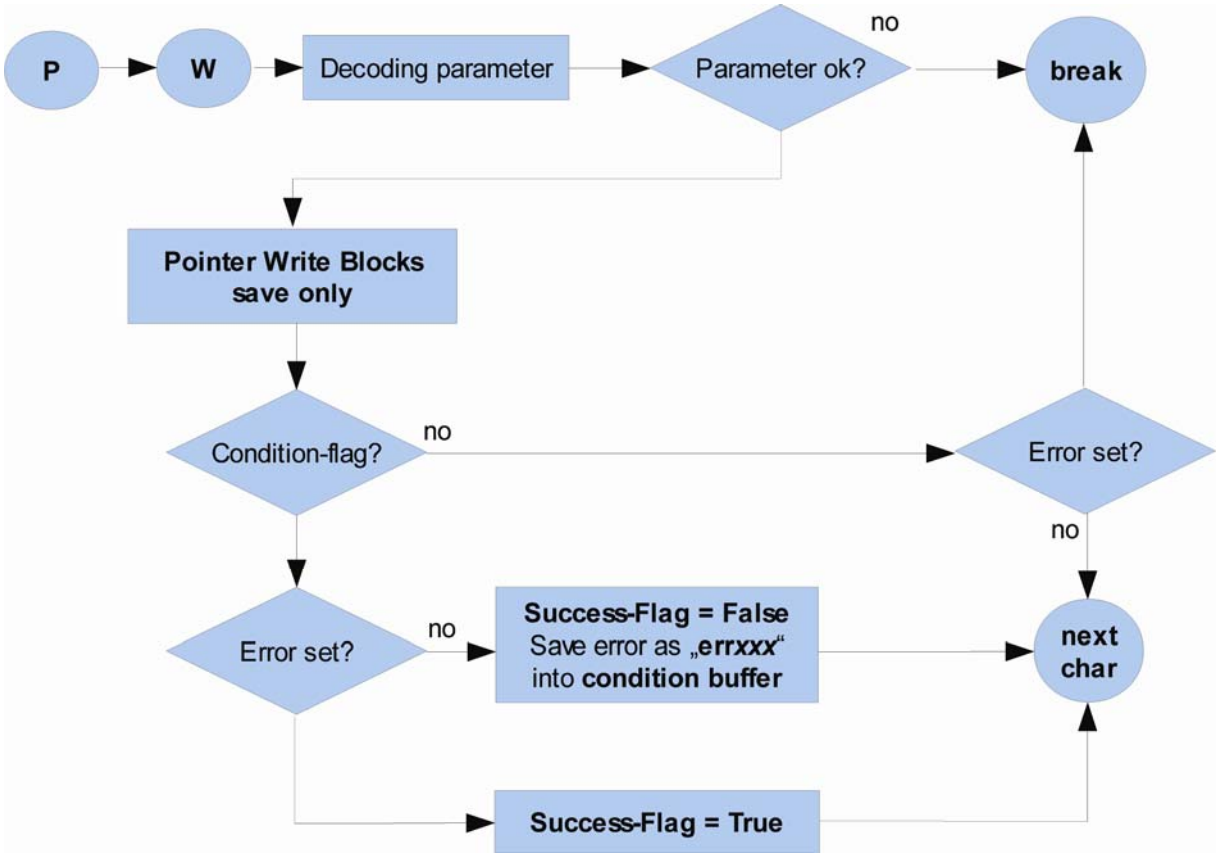
3.10.4.6.2 Parameter Description

Parameter	Description
empty	Everything writing with 0 (see „%mb get tag“Command „mb“)
a	Writing from **a.** Byte (if not specified → from the first Byte)
“*{text}*“	The character string {text} contains the information to be described. From the length of the character string the number of the bytes to be described is calculated. With *Byte-Input* [ ] : The same rules are valid similar to the text command - see section Text – Command! With *Nibble-Input* { } : The characters 'A' to 'F' must be capitalised. Max. 8 bytes or the length must be an even number. With *Decimal-Input* ( ) : The number may amount to max. 65535. Hexadecimals are marked with an 'H' before: max. HFFFF

3.10.4.6.3 Description

1. Parameter is checked for syntax and validity and is saved.
2. If a communication error occurs (e.g., error excess) there are 2 possibilities:
    1. If the command is within a condition, the error code is saved in form from "errxxx" where xxx is the error code, as a character string. **Success flag** is set to "**False**".
    2. If the command is outside a condition, the sequence is exited.
3. If the "Pointer Write" command is within a condition, the Success flag is set to "True".

3.10.4.6.4 State diagram



3.10.4.7 Trigger Timeout

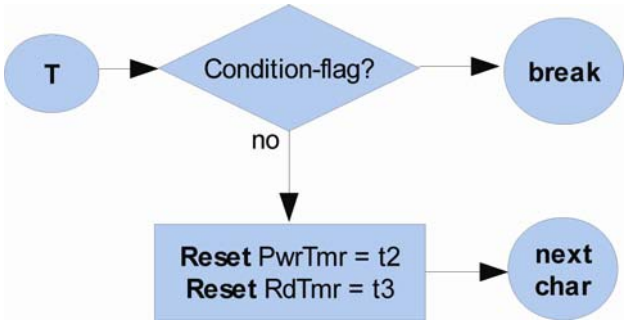
If multiblock is applied automatically, the elapsed-time metres are in full operating mode and are no longer reset

independently. The user can determine, when or under which condition a back-spacing of the elapsed-time metres should occur.

3.10.4.7.1 Syntax

T

3.10.4.7.2 State diagram



3.10.4.8 Tag Release

To avoid a repeated readout or describing a tag – e.g., in the automated operating mode –The command „Tag Release“ is recommended. This command can be used in a condition and allows the user during a sequence to fix the errors independently.

3.10.4.8.1 Syntax

Q

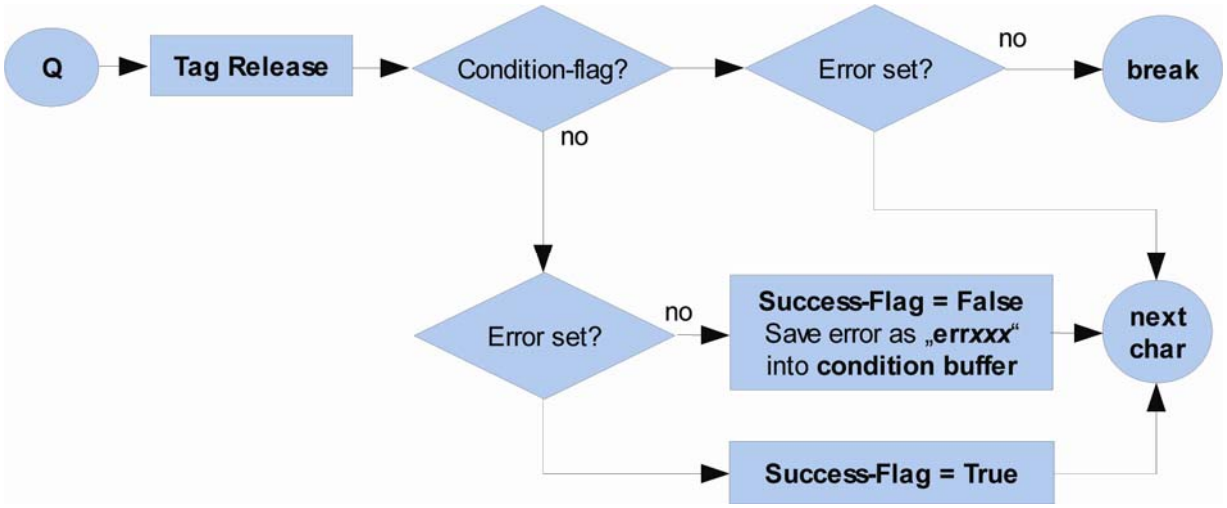
3.10.4.8.2 Description

The function „Tag Release“ puts the HF module in "continuous mode", and waits to another tag (UID) in the field, or Tag no longer in the field.
If a communication error occurs (e.g., time excess of 3 seconds) there are two possibilities:

1. If the command within a condition, the error code is saved in form from "errxxx" where xxx is the error code, as a character string. Success flag is set to "False".
2. Is the command outside a condition, the sequence is exited.

If the „Tag Release“ command is within a condition, the Success flag is set to "True".

3.10.4.8.3 State diagram



3.10.4.9 Beep

The user has the possibility to set an audible signal – e.g., after a successful reading or writing process. Besides the audio signal, the tone, duration and volume of the signal is configurable..

3.10.4.9.1 Syntax

b[]
b[*a*]
b[*a*,*b*]
b[*a*,*b*,*c*]
b[*a*,*b*,*c*,*d*]

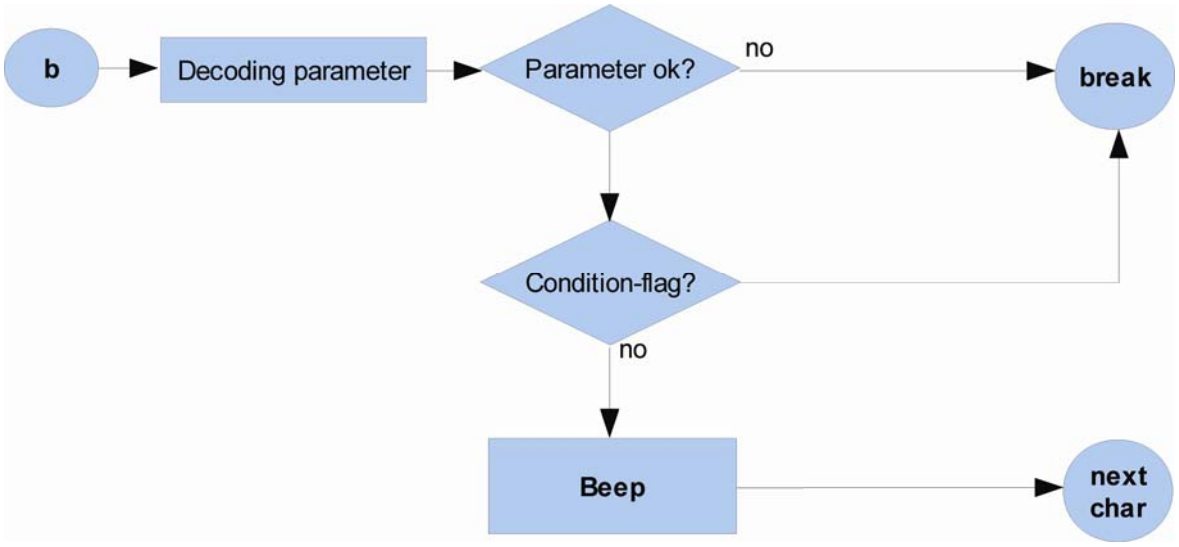
3.10.4.9.2 Parameter Description

Parameter	**Description**
empty	frequency is 300Hz; duration is 100ms; volume is 3; continuation occurs immediately
a	**a** corresponds of the frequency f. ton = **a***10Hz (e.g. 44 corresponds 440Hz) *Duration is 100ms; volume is 3*; continuation occurs immediately
b	**b** corresponds of the duration t.ton = **b***100ms (e.g. 10 correspond 1 second) *Volume is 3; continuation occurs immediately*
c	**c** corresponds of the frequency 0 is minimum and 3 is Maximum *Continuation occurs immediately*
d	**d** can be 0 or 1. With 0 the sequence is continued immediately and with 1 the duration of the signal is waited. Ahead a reading or writing process the wait of the signal end are recommended, so d=1.

3.10.4.9.3 Description

1. Parameter is checked for syntax and validity and is saved.
2. The "Beep" command is executed according to the parameters

3.10.4.9.4 State diagram



3.10.4.10 Imager Decode – Command

If an Imager module is installed, it is possible, within the sequence, to send a Decode to control and then display the received character string or complete a comparison.

3.10.4.10.1Syntax

Output in Byte
I[]
I[*a*]
I[*a*-*b*]

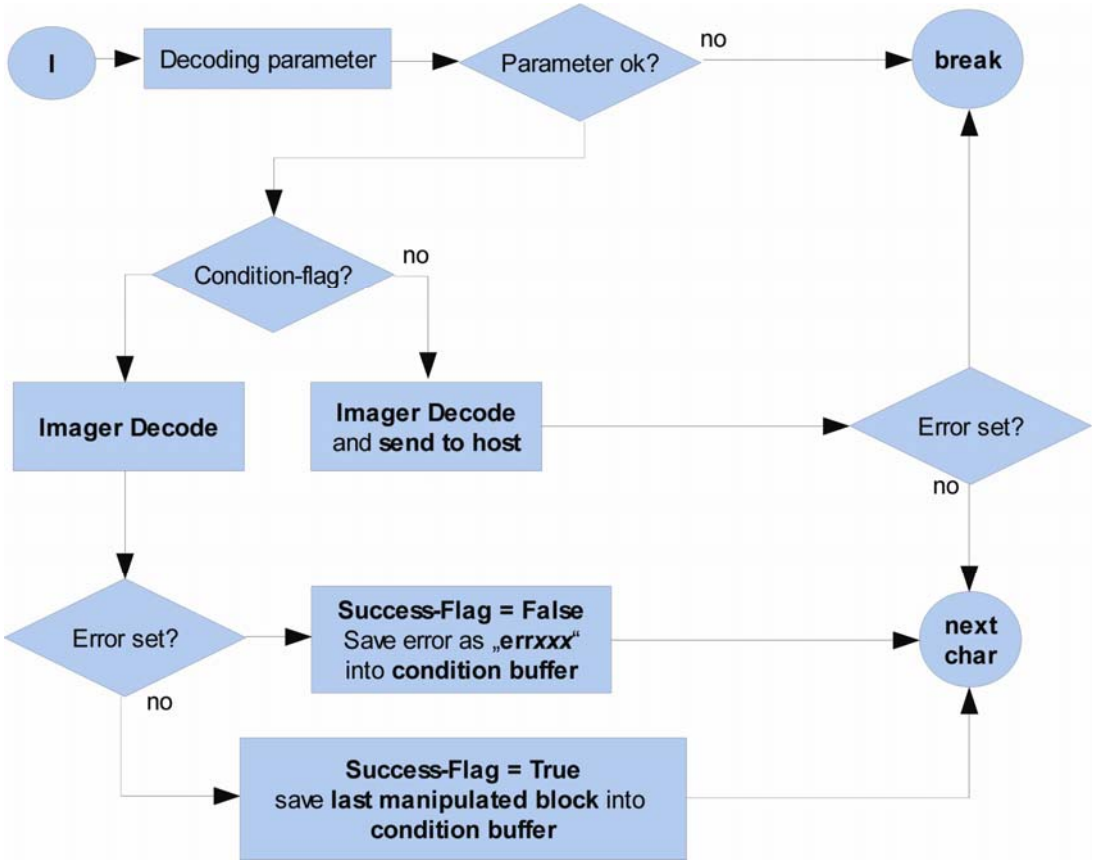
3.10.4.10.2 Parameter Description

Parameter	Description
empty	Read out all
a	Reading the **a.** Bytes
a-b	Reading from the **a.** to **b.** Bytes

3.10.4.10.3 Description

1. Parameter is checked for syntax, validity and is saved.
2. If the command is **within a condition** the function **'Imager Decode'** is called without output. If it is **outside a condition** the function is called **with output** .
3. If a communication error occurs (e.g., time excess) there are 2 possibilities:
    1. If the command is within a condition, the error code is saved in form from "errxxx" where xxx is the error code, as a character string. **Success flag** is set to "**False**".
    2. If the command is outside a condition, the sequence is exited.
4. If the "Imager Decode" command is within a condition, the character string is saved in the buffer and the **Success flag** is set to "**True**".

3.10.4.10.4Sate diagram



3.10.4.11 Conditions

Often it takes to be checked memory contents or UIDs. With four characters this is possible with multiblock:

?	If	
:	then	output = true
/	else	output = false
;	end if	

3.10.4.11.1 ? .. if

As soon as the "if" character? is recognised, the "Condition flag" is set on and signals to all other following commands that they are in a condition. Two properties are same in one condition with every command:

1. No output occurs to the host.
2. No abnormal termination of the sequence by communication error or timeouts.

The table in the section 2.8.4 Instruction List shows the commands which can be used within a condition. Up to the command „Release tag“ all commands can be compared, with the help of the operators, basically to numbers or character strings.

Possible operators within a condition are:

character	description	application
<	less	Numbers
<=	equal to or less than	Numbers
>	greater	Numbers
>=	not less than	Numbers
=	equal	Numbers & character strings
!=	not equal to	Numbers & character strings

Some commands can also be applied without operators – these put the "**Success flag**",
which fixes the success, e.g. of a communication with "true" (=successfully) or "false" (= not successfully).

It is possible only for one comparison in a condition.

3.10.4.11.2: .. Then

The **":" - Character** forms the **end of a condition** and resets the "Condition flag". If an operator was applied and two variables have been given, these variables are checked and compared for validity. Then the result is either "true" or "false".

If no operator was applied, a check is done to see whether the "Success flag" was set and this then transmitted.

If the result is **"false"**, then the else is checked next **"/" - character** (=else).

If the result is **"true"**, the **next character** is evaluated normally.

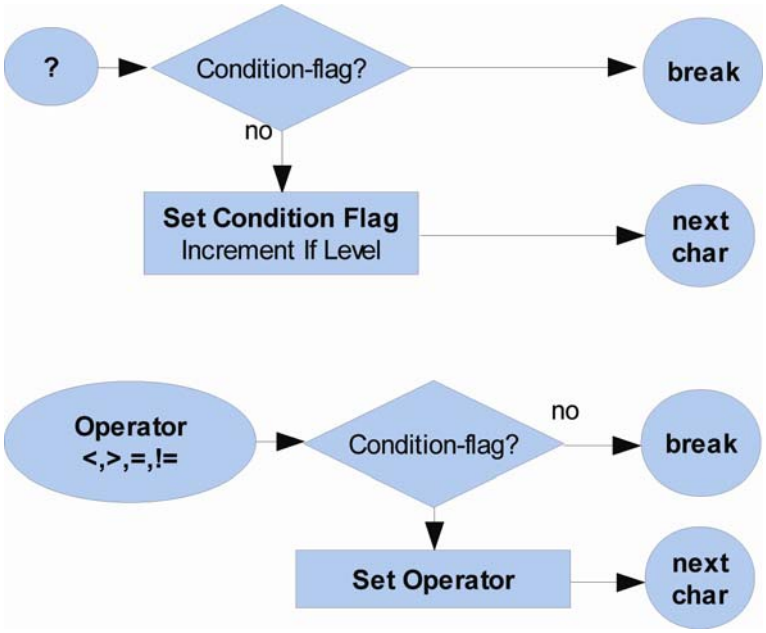
3.10.4.11.3/ .. Else

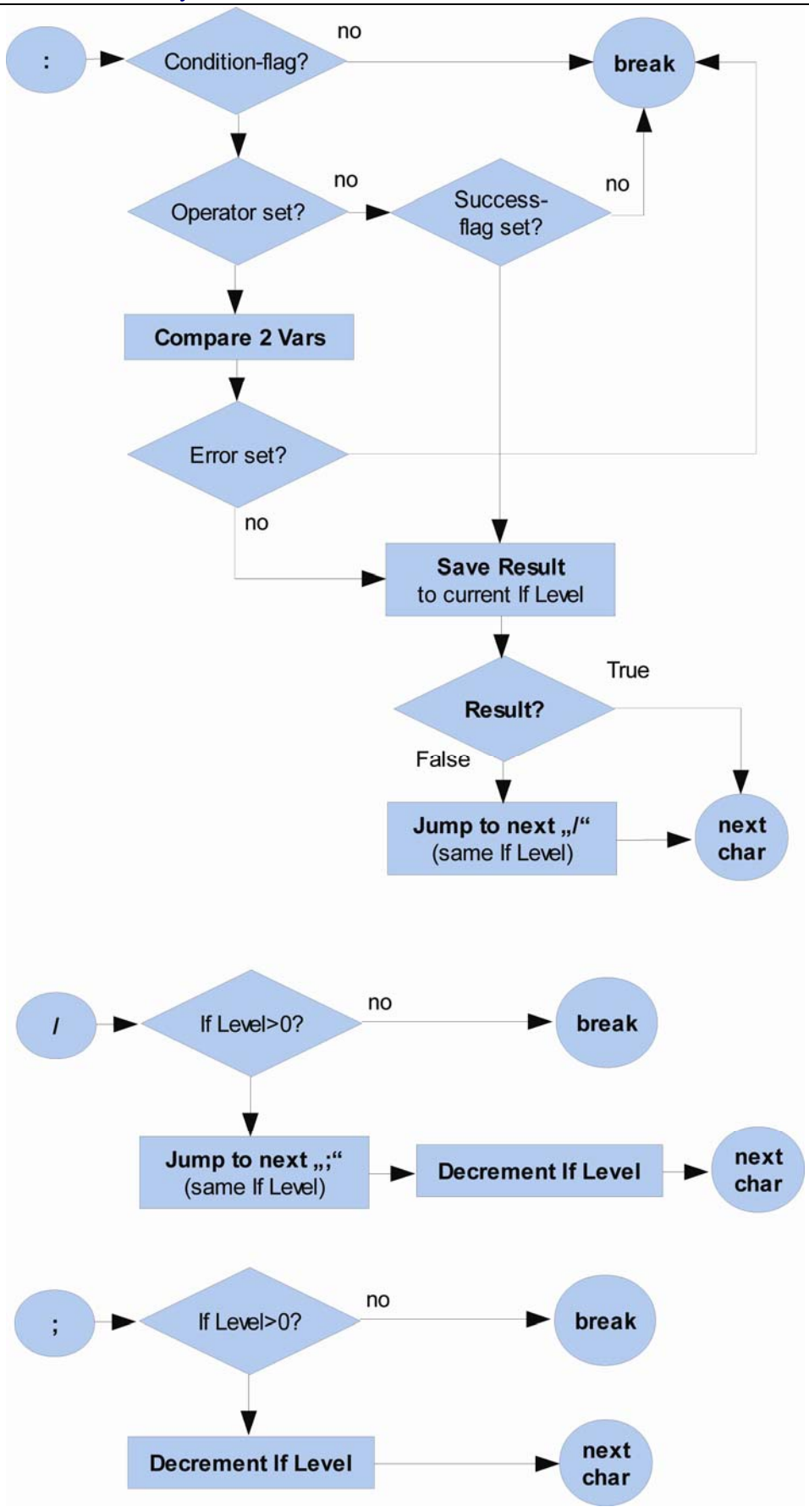
The **"/" - character** is relevant only for a "**true**" result and signals the Jump up to the **";" character** (end If). If no commands are planned for a "**false**" result, the **"/" character** must form the end for this command string.

3.10.4.11.4; .. End If

This character forms **the end of an "If" instruction.**

3.10.4.11.5 State diagram




:
Condition-flag?
no
break
Operator set?
no
Success-flag set?
no
Compare 2 Vars
Error set?
no
Save Result to current If Level
Result?
True
False
Jump to next „/“ (same If Level)
next char
/
If Level>0?
no
break
Jump to next „;“ (same If Level)
Decrement If Level
next char
;
If Level>0?
no
break
Decrement If Level
next char

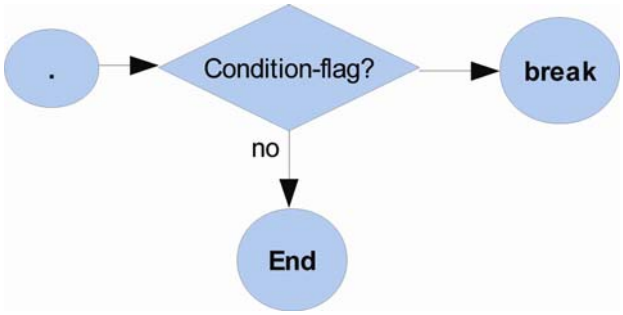
3.10.4.12 End – Command

The final command quits the sequence and can be used everywhere, except within a condition.

3.10.4.12.1Syntax

.

3.10.4.12.2 State diagram



3.10.5 Error Codes

In this section a complete list of possible error codes are listed which can occur within a sequence.

Error code	Description
50	unknown command
51	no value
52	value expected
53	value out of range
54	unexpected end
55	if - condition error
56	if - level overflow
57	if - level is zero
58	if - level - other failure
59	if - condition expected
60	if - condition - operator already set
61	if - condition - can not compare
62	if - condition - wrong operator
63	if - condition - can not exec command
64	hex value expected
65	if - condition - unknown operator
66	text expected
67	converting text failed
68	parameter - fast already set
69	parameter - brace open expected
70	parameter - wrong brace type

71	parameter - too many parameters
72	parameter - unknown parameter
73	parameter - wrong output type
74	parameter - need more parameters
75	parameter - text-parameter already set
76	block length - out of range
77	converting nibble string failed
78	converting nibble to byte failed
79	converting byte to nibble failed
100	micro engine error - X
101	micro engine error - F
102	micro engine error - C
103	micro engine error - N
104	micro engine error - Q
105	micro engine error - U
106	micro engine error - R
107	micro engine error - O
108	wrong mode
109	timeout reset
110	timeout get uid
111	timeout select
112	timeout read
113	timeout write
114	timeout stop continuous mode
115	write failed
116	pointer read wrong parameter
117	pointer read overflow
118	pointer write wrong parameter
119	pointer write overflow
120	wrong uid
121	release tag command - failed
150	beep command - wrong parameter
160	imager command - failed

4 Appendix A

Glossary

5 Technical Support

Contact Information

Support: Please send your questions by mail to sa@datatronic.eu.

Worldwide, Technical Support is available through our office located in Europe:

DATATRONIC IDentsysteme GmbH

Dreisteinstraße 47

AT 2372 Gießhübl

AUSTRIA

Tel.: 0043 (0) 2236 / 377668-0

Fax: 0043 (0) 2236 / 377668-11

DATATRONIC IDsystems Ltd

1 Medway Ave Oakley, Hampshire RG23 7DP, UK

UNITED KINGDOM

Tel. 0044 (0) 1256 533 -560

Fax 0044 (0) 1256 533 -560-581

mail@datatronic.eu

http://www.datatronic.eu

TagTrans ProgrammingManual E03.doc