



voyantic

# Voyantic **Tag**surance UHF Tester

Manual

rev.1.6.0

**Compatible with:**

**Software version 1.8.0**

**Firmware version 1.6.0**

# 1 Important Information

**PLEASE READ THE COMPLETE USER GUIDE CAREFULLY BEFORE USING THE VOYANTIC TAGSURANCE™ UHF TESTER SYSTEM.**

Voyantic Ltd. operates a policy of ongoing development. Voyantic Ltd. reserves the right to make changes and improvements to any of the products described in this user guide without prior notice.

THE CONTENTS OF THIS USER GUIDE ARE PROVIDED "AS IS". EXCEPT AS REQUIRED BY APPLICABLE LAW, NO WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, ARE MADE IN RELATION TO THE ACCURACY, RELIABILITY OR CONTENTS OF THIS USER GUIDE. VOYANTIC LTD. RESERVES THE RIGHT TO REVISE THIS USER GUIDE OR WITHDRAW IT AT ANY TIME WITHOUT PRIOR NOTICE.

The General Terms and Conditions of Voyantic Ltd. shall apply. These can be downloaded at: <http://www.voyantic.com/termsandconditions.pdf>

THE MEASUREMENT UNIT ENCLOSURE DOES NOT CONTAIN ANY USER SERVICEABLE PARTS AND SHALL NOT BE OPENED. Voyantic Ltd. will inspect the seals of the equipment in case of warranty and maintenance procedures. If the seals are broken and the device therefore suspected to be tampered with, the regular warranty and the extended warranty under the maintenance program will be void.

The Voyantic Tagsurance UHF Tester makes use of radio frequencies. If the device RF-port is connected to a radiating element, such as an antenna outside a shielded environment, local permissions or approvals may be needed. VOYANTIC LTD. DOES NOT WARRANT ANY TYPE OF APPROVAL FOR VOYANTIC TAGSURANCE UHF TESTER. VOYANTIC LTD. SHALL UNDER NO CIRCUMSTANCES BE LIABLE OF ANY USE OF VOYANTIC TAGSURANCE UHF TESTER.

Use of any additional software requires a valid license. Any copyrights, patents and other intellectual property rights (including the right to change and further develop) in and to the Voyantic Tagsurance UHF Tester (including any related documentation and other materials delivered by Voyantic Ltd.) shall belong to Voyantic Ltd.

## 2 Table of Contents

<b>1</b>	<b>IMPORTANT INFORMATION</b>	<b>2</b>
<b>2</b>	<b>TABLE OF CONTENTS</b>	<b>3</b>
<b>3</b>	<b>PRODUCT OVERVIEW</b>	<b>4</b>
3.1	OPERATING PRINCIPLE	4
3.2	FUNCTIONS AND LICENSE OPTIONS	4
3.3	TECHNICAL SPECIFICATIONS	4
<b>4</b>	<b>INSTALLATION</b>	<b>5</b>
4.1	UNPACKING INFORMATION	5
4.2	INSTALLING THE TAGSURANCE GRAPHICAL USER INTERFACE	6
4.3	SETTING UP A MEASUREMENT SETUP	7
4.4	SIGNAL CONNECTIONS	8
4.5	INTEGRATION AND CONNECTIVITY OPTIONS	12
<b>5</b>	<b>OPERATION USING THE TAGSURANCE GRAPHICAL USER INTERFACE</b>	<b>13</b>
5.1	OVERVIEW	13
5.2	SYSTEM SETTINGS	15
5.3	ANTENNA SETUP CHECK	17
5.4	PERFORMANCE TEST CASE DEFINITION	18
5.5	ENCODE OPTIONS	24
5.6	MEASUREMENT PERFORMANCE AND MONITORING	30
5.7	ERROR HANDLING	36
<b>6</b>	<b>OPERATION USING THE SERIAL COMMAND INTERFACE</b>	<b>38</b>
6.1	COMMUNICATION PROTOCOL	38
6.2	SYSTEM CONFIGURATION COMMANDS	39
6.3	INLINE MEASUREMENT COMMANDS	40
6.4	OTHER MEASUREMENT COMMANDS	45
6.5	TAG ENCODING, LOCKING, AND KILLING COMMANDS	47
6.6	COMBINING ENCODING WITH PERFORMANCE TESTING	54
6.7	ERROR HANDLING	54
<b>7</b>	<b>UPDATING THE DEVICE FIRMWARE AND LICENSE</b>	<b>55</b>
<b>APPENDIX A:</b>	<b>TCP REMOTE ACCESS INTERFACE FOR TAGSURANCE GUI</b>	<b>A1</b>
A.1	OVERVIEW	A1
A.2	TCP/IP CONNECTION ESTABLISHMENT	A2
A.3	COMMANDS DESCRIPTION	A3
A.4	APPLICATION EXAMPLE	A20
A.5	HEARTBEAT	A21
<b>APPENDIX B:</b>	<b>UDP TEST DATA BROADCASTING INTERFACE</b>	<b>B1</b>
<b>APPENDIX C:</b>	<b>ENABLING TAGSURANCE GUI OPTIONS BY USING PROGRAM CALL ARGUMENTS</b>	<b>C1</b>
<b>APPENDIX D:</b>	<b>SUMMARY OF SERIAL INTERFACE COMMANDS</b>	<b>D1</b>
<b>APPENDIX E:</b>	<b>SUMMARY OF ERROR INDICATORS</b>	<b>E1</b>

## 3 Product Overview

### 3.1 Operating Principle

The testing of the tag samples is always carried in the active state using the selected carrier frequency and transmitted power. The tag is given a command according to the ISO 18000-6C protocol and it is detected if a valid response is sent back by the tag. Active mode is always preferred as it tests both the performance and the functionality by challenging the tag in a similar way as an RFID reader providing applicable results.

The measurement unit is optimized to operate over the extended frequency range of 800 MHz to 1100 MHz. The reason for the large range is to match and exceed the operating range of the tags. RFID inlay tags typically operate on some bandwidth on a range from 850 MHz to around 1050 MHz. This way the tag is more tolerant to material induced detuning, and has an optimally stable performance over the entire 865 MHz to 960 MHz range even when attached on various materials. As the broad operating range is important for the tag, it is also important to verify the bandwidth for every tag in production. This is why the 800 MHz to 1100 MHz operating range is needed.

### 3.2 Functions and License Options

OPTIONS	STANDARD FEATURES	OPTIONAL FEATURES
Frequency range	860 – 960 MHz	800 – 1100 MHz
Points test mode	x	
Threshold sweep	(only ref curve)	x
Read test		x
Write test		x
Sensitivity test		x
Encode, kill, lock		x

### 3.3 Technical Specifications

#### Absolute maximum values

RF input power	+30 dBm
External power voltage range	17 -20 VDC

#### Specification

RF output power range	-10 dBm to +25 dBm
RF output power setting accuracy	+/- 1 dB
RF output power setting nominal step size	0.25 dB
Carrier frequency accuracy	10 ppm
Carrier frequency resolution	100 kHz

## 4 Installation

### 4.1 Unpacking Information

The Tagsurance UHF Tester system includes the following items:

1. USB memory stick containing the measurement software package
2. Tagsurance unit
3. Delivery Kit (optional)
  - 18VDC Power supply
  - RS232-cable
  - Carrying case
  - D15 I/O screw terminal adapter
  - Mounting brackets
4. Starter Kit (optional)
  - Delivery Kit
  - Snoop Pro coupling element
  - Antenna cable
  - Reference plate
  - Reference tag



**Fig 1 Delivery package containing Starter Kit**

## **4.2 *Installing the Tagsurance Graphical User Interface***

The Tagsurance GUI is a software for Windows XP/Vista/7/8. It is designed for the use of the Tagsurance UHF Tester hardware unit. The main operations of the program are: test case generation, test case execution, and test data gathering from the Tester unit to an output file.

### **PREPARATION**

If you are using Windows 8, it is recommended to turn off the Windows fast startup during the installation or alternatively manually fully reboot the PC after installation to enable proper completion of the driver installations.

### **INSTALLATION**

To install the Tagsurance Graphical User Interface:

1. From the USB Stick browse and run: .../Tagsurance GUI Installer v..../setup.exe
2. Select the destination folder paths for the installation files
  - Note that when used, the program creates and updates files under its own installation folder. The program and the user must have the privileges for this for correct operation. Specific Windows system folders have restricted access for programs to write to. This includes the Program files folder in Windows Vista/7/8.
3. Read and accept the license agreements and follow the instructions on the screen to install the product
4. The summary of the contents to-be-installed/modified will be presented for acceptance before starting the installation
5. After successful installation, the installer prompts the user to restart the computer
6. The program can be run from the Start menu

## 4.3 Setting Up a Measurement Setup

Setting up the basic test setup requires:

1. The Tagsurance Unit with a power supply
2. Antenna
3. Antenna cable
4. RS-232 serial data cable
5. A computer with Tagsurance GUI installed
6. Serial adapter (if a native serial port is not available)
7. I/O beeper connector and pedal trigger (optional for manual testing)

Assembly instructions:

1. Connect antenna to the antenna port in the device front panel (RF 50Ω)
2. Connect serial cable from the computer to the serial port in the device back panel (RS-232)
3. Connect power supply to the Power in port in the device back panel (18VDC)
4. Connect I/O beeper connector to the device I/O port and pedal trigger to the plug in the beeper component (only applicable for the Manual Test Station)
5. Start the Tagsurance GUI application to perform measurements



**Fig 2 Basic measurement setup with Manual Test Station Kit**

4.4    *Signal Connections*

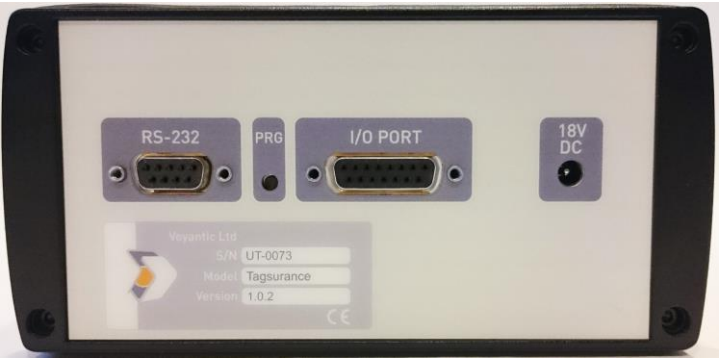
4.4.1    **The Tagsurance UHF Tester Front and Back Panel**



**Fig 3 Voyantic Tagsurance UHF Tester front panel**

**FRONT PANEL**

<b>TAGSURANCE</b>	Indicates that the power is on and the device is working properly
<b>READY</b>	Indicates that the device is ready for new commands or trigger
<b>BUSY</b>	Indicates that the device is busy
<b>PASS</b>	Indicates a passed inline test
<b>FAIL</b>	Indicates a failed inline test
<b>RF 50Ω</b>	Antenna connector



**Fig 4 Voyantic Tagsurance UHF Tester back panel**

**BACK PANEL**

<b>RS-232</b>	Serial cable connector	Type: DB9, female
<b>I/O PORT</b>	Extension port connector	Type: DB15, female
<b>18VDC</b>	Power supply voltage connector	



## 4.4.2 I/O Signalling

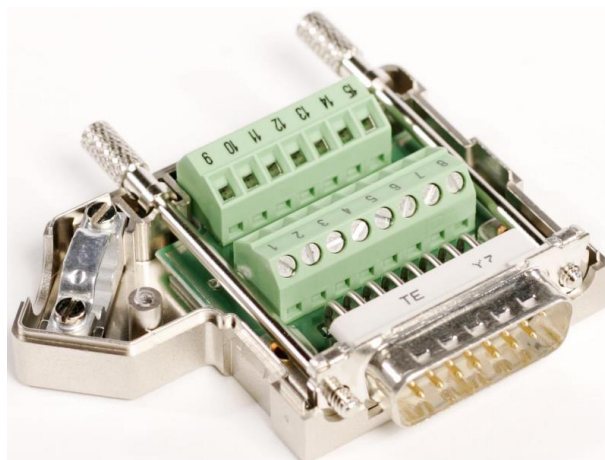


Fig 5 External I/O connector

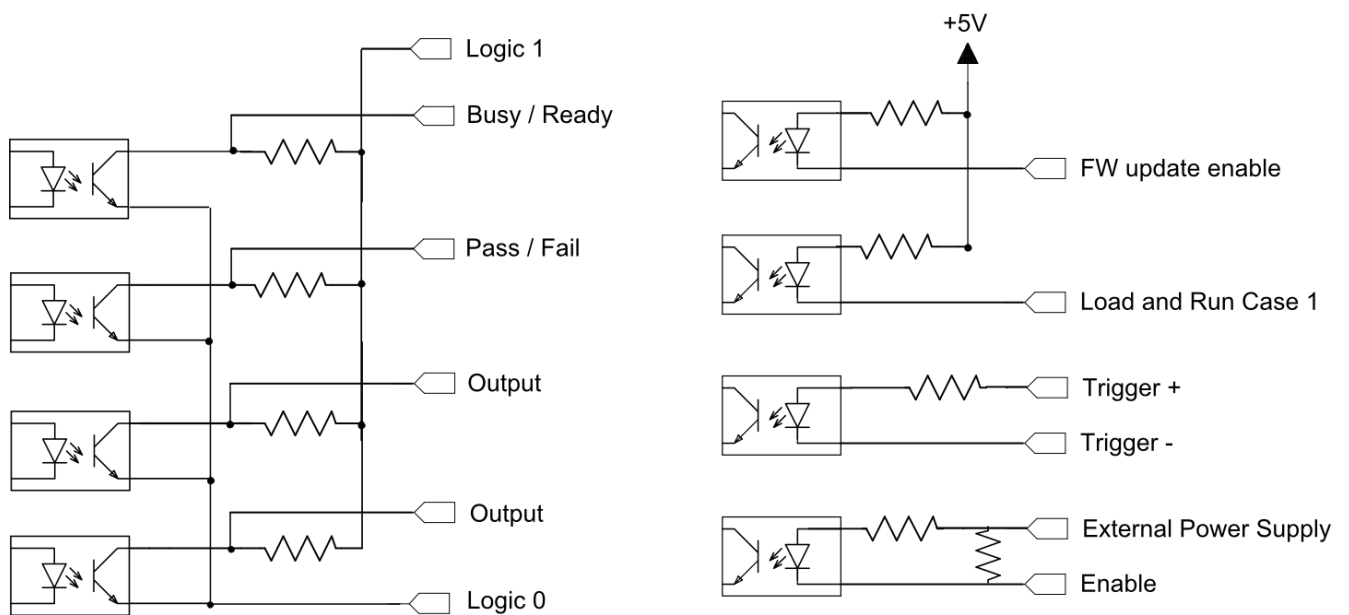
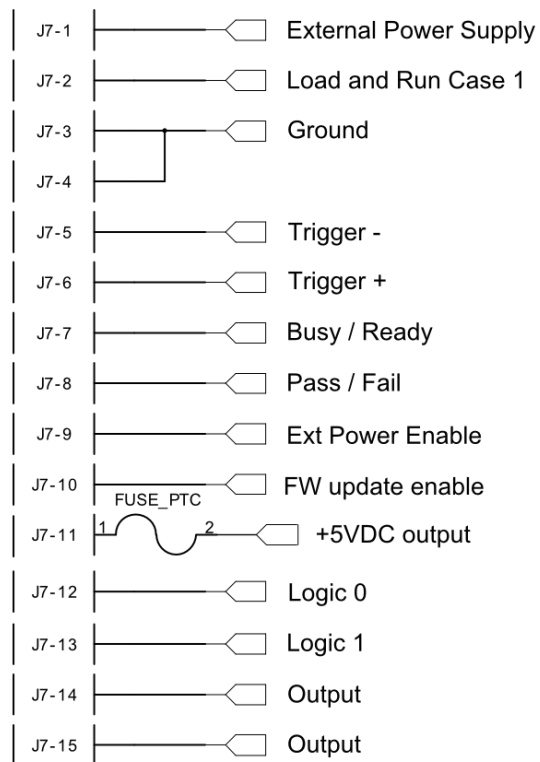
Pin	Function
1	External power supply input (+18VDC)
2	Load and run case number 1 input (active low <sup>1,2</sup> )
3	Ground
4	Ground
5	External trigger input -
6	External trigger input +
7	Busy/ready indicator output (busy=HIGH)
8	Pass/fail indicator output (default: pass=HIGH)
9	External power supply enable input (active low)
10	Firmware update enable input (active low <sup>1</sup> )
11	+5VDC output (maximum current: 100mA)
12	Logic 0-level reference input
13	Logic 1-level reference input (max voltage: 24V)
14	Output: (reserved for future use)
15	Output: (reserved for future use)

<sup>1</sup> Signal takes effect when Tagsurance test result is powered up

<sup>2</sup> Does not work with Tagsurance GUI

The high/low levels of the output pins 7, 8, 14 and 15 are set by the pins 12 and 13. The voltage level of the pin 13 must be higher than the voltage of pin 12.

The trigger signal is a 5-24V differential signal between pins 5 and 6. If a unipolar signal is preferred, pin 5 can be grounded to either pin 3 or pin 4.



**Fig 6 I/O schematic drawing**

The following figure explains the signalling with different setting options:

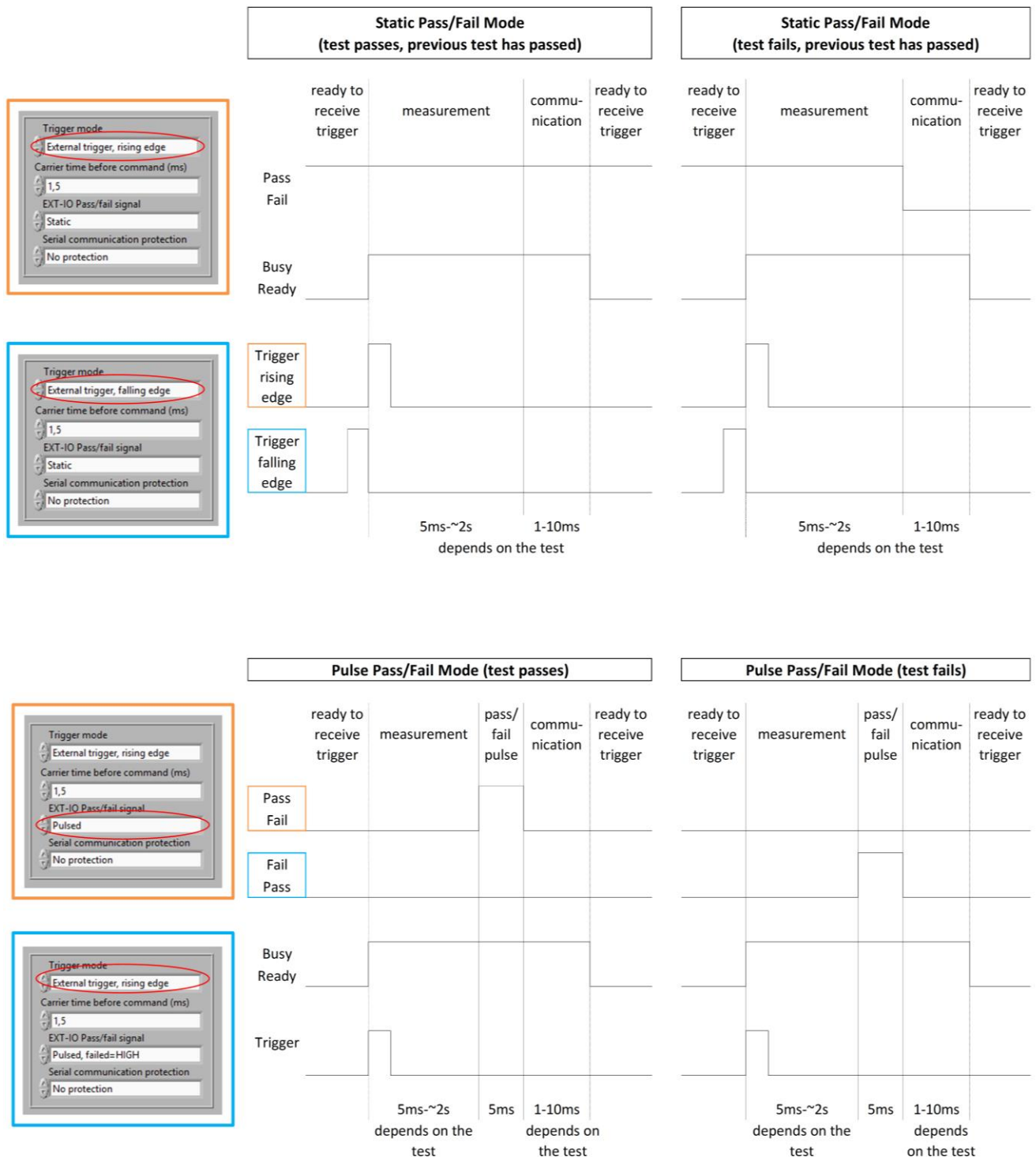


Fig 7 Signal timing diagrams

## 4.5 Integration and Connectivity Options

The Tagsurance test system provides multiple connectivity options allowing various alternatives for test station build-up and machine integration (Fig 8).

The Tagsurance Tester Unit provides two connectivity options: serial port (RS-232) and EXT-IO connector (DB-15). **Serial port (RS-232)** connection is the basic interface providing the most comprehensive control over the device operation. Application of serial communication interface is described in chapter 6. **EXT-IO-port** is the interface for machine connections which provides trigger input, busy/ready-indication, pass/fail indication, and external power supply connection. These and other EXT-IO signals are described in chapter 4.4.

The Tagsurance GUI reserves serial port connection, but adds three alternative interfaces: graphical user interface, TCP remote access interface, and UDP result broadcasting interface. **Tagsurance GUI** provides an intuitive interface for performing and monitoring production testing and encoding. GUI operation is described in chapter 5. **TCP remote access interface** is a feature of Tagsurance GUI which allows remote control of the test and encoding operations. By implementing remote access command set, the user can easily add control of testing flow as a part of their own testing solution. **UDP broadcasting interface** is a one-direction communication interface used to report test data from the Tagsurance GUI to a remote target. This allows, for example, integration of testing to external encoding platforms or special data logging practices. TCP interface is described in Appendix A and UDP broadcasting interface in Appendix B.

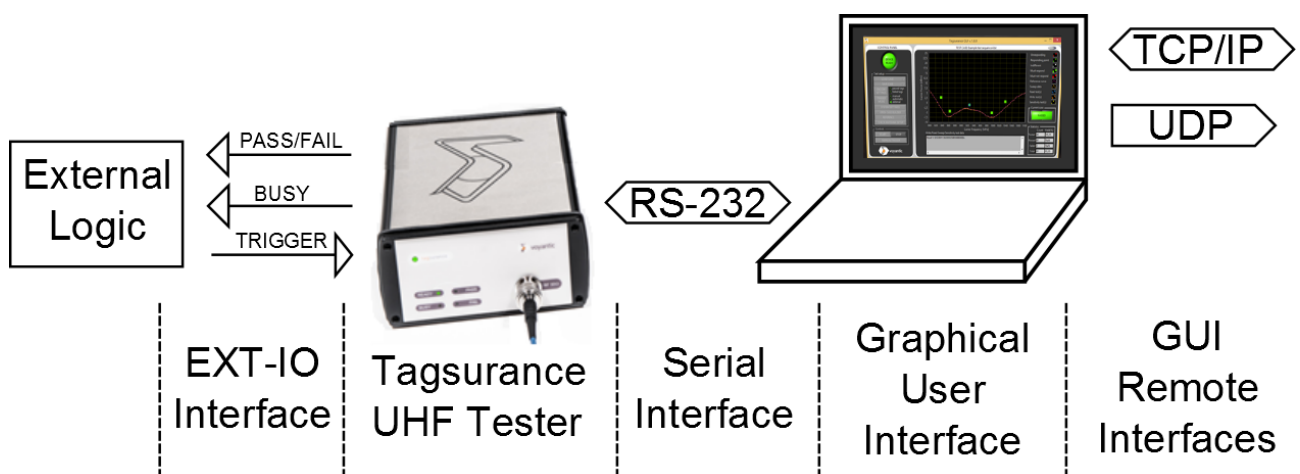


Fig 8 Tagsurance test system machine integration and connectivity options

## 5 Operation Using the Tagsurance Graphical User Interface

### 5.1 Overview

To start the application, run: 'tagsurance\_gui.exe', or use the shortcut in the Windows Start menu. The program will automatically recognize the Tagsurance test unit connected to the PC and open the startup view once the serial port connection is initialized. In case there are multiple devices connected to the computer, the user will be prompted to choose the one to be used. An error message is given if no devices can be found or the firmware version does not match with the GUI. After successful initialization, the measurement system is ready to use and the device ready indicator will be lit (Fig 9). The device information can be viewed by taking the cursor on the device ready indicator.

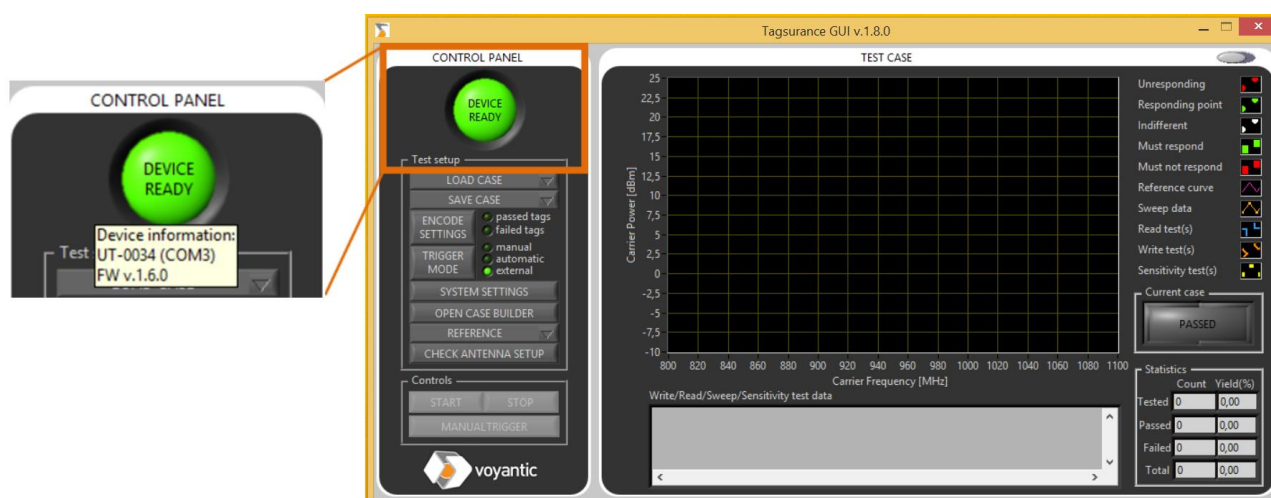


Fig 9 Application software front panel: startup view

The application interface has two main sections: control panel and test case.

**Control panel** includes all the necessary controls for defining test settings and performing measurements. Through these controls (see table below), it is possible to configure system settings (device, remote port, broadcasting), handle performance test cases and encoding options (load, save, modify), perform antenna setup check (with reference tag), and control measurements (start, stop, trigger). System configuration is discussed in Chapter 5.2, antenna setup check procedure in Chapter 5.3, performance test case definition in Chapter 5.4, encoding options in Chapter 5.5, and measurement process and monitoring in Chapter 5.6.1.

**Test case section** summarizes current test case parameters and the defined encoding tasks, and, once testing has been started, provides an interface for monitoring measurement results. Two alternative views are provided: basic view and statistical view. In the basic view, data from all tests and encoding tasks is shown in detail for the latest test performed. In the statistical view, data from certain type of performance tests (sensitivity tests), is presented on a timeline (as a function of tag test index) and as a histogram. Display items are explained in the table below and a detailed description of different views is provided in Chapter 5.6.2 and 5.6.3.

## CONTROL PANEL

<b>Load case</b> (menu)	Load and activate test case from a file or from the device memory <sup>1</sup>
<b>Save case</b> (menu)	Save a test case to a file or to the device memory <sup>1</sup>
<b>Encode settings</b> <sup>2</sup>	Open encode settings dialog (chapter 5.5)
<b>Trigger Mode</b> <sup>2</sup>	Change trigger mode - Manual: a new measurement is triggered by clicking “Manual trigger” -button - Automatic: a new measurement is auto-triggered after completing previous one - External: a new measurement is triggered by providing trigger signal to EXT-IO port (in the device back panel). Signal polarity is defined from system settings.
<b>System settings</b>	Open system settings dialog (see chapter 5.2)
<b>Open case builder</b>	Open interactive Test Case Builder (see chapter 5.4)
<b>Start case</b>	Run active test case
<b>Stop case</b>	Stop running test case
<b>Reference</b> (menu)	Choose reference curve operation (load from file, save to file, measure <sup>3</sup> , or hide)
<b>Check antenna setup</b>	Open view for comparing current setup to a specific reference curve (chapter 5.3) <sup>4</sup>
<b>Manual trigger</b>	Triggers a new measurement (only in manual trigger mode <sup>2</sup> )

<sup>1</sup> There are 5 memory banks available in the device memory for saving test cases (1 case/bank).

<sup>2</sup> Settings have to be defined prior to starting the test. Changes are not possible while a test is running.

<sup>3</sup> The frequency range applied depends on the concurrent license options. Step size is 5MHz.

<sup>4</sup> The reference curve must be loaded (from Reference menu) before entering antenna setup check mode.

## TEST CASE (Basic view)

<b>Graph</b>	Before starting: reference curve (optional), test specifications After starting the test and triggering: reference curve (optional), sweep test results, point test specifications and results, sensitivity test results
<b>Write/Read/Sweep/Sensitivity test data</b>	Before starting the test: test specifications for write/read/sweep/sensitivity test After starting the test and triggering: write/read/sensitivity test data
<b>Current Case</b>	Indication of whether the latest measurement results meet test case criteria
<b>Statistics</b>	Tested: Count of the tested tags, performance test yield Passed: Count of the tags that passed performance test, yield of the encoding executed for passed tags (NaN if no encoding has been executed) Failed: Count of the tags that failed performance test, yield of the encoding/kill executed for failed tags (NaN if no encoding/kill has been executed) Total: Total count of the tags that passed performance test and were successfully encoded (if encoding has been defined), total yield of performance test passed and successfully encoded tags

## 5.2 System Settings

### 5.2.1 Overview

The device settings and the operation of the Tagsurance GUI can be configured from system settings dialog (Fig 10). These settings influence the measurement performance and have to be defined before starting a measurement. Some of the features will be stored to device memory, and only have to be defined once. Such features are trigger type and polarity (for EXT-IO trigger), carrier before command, and EXT-IO Pass/fail signal form. The rest have to be reprogrammed after GUI startup, or defined in tagsurance\_gui.exe program call arguments (see appendix C).

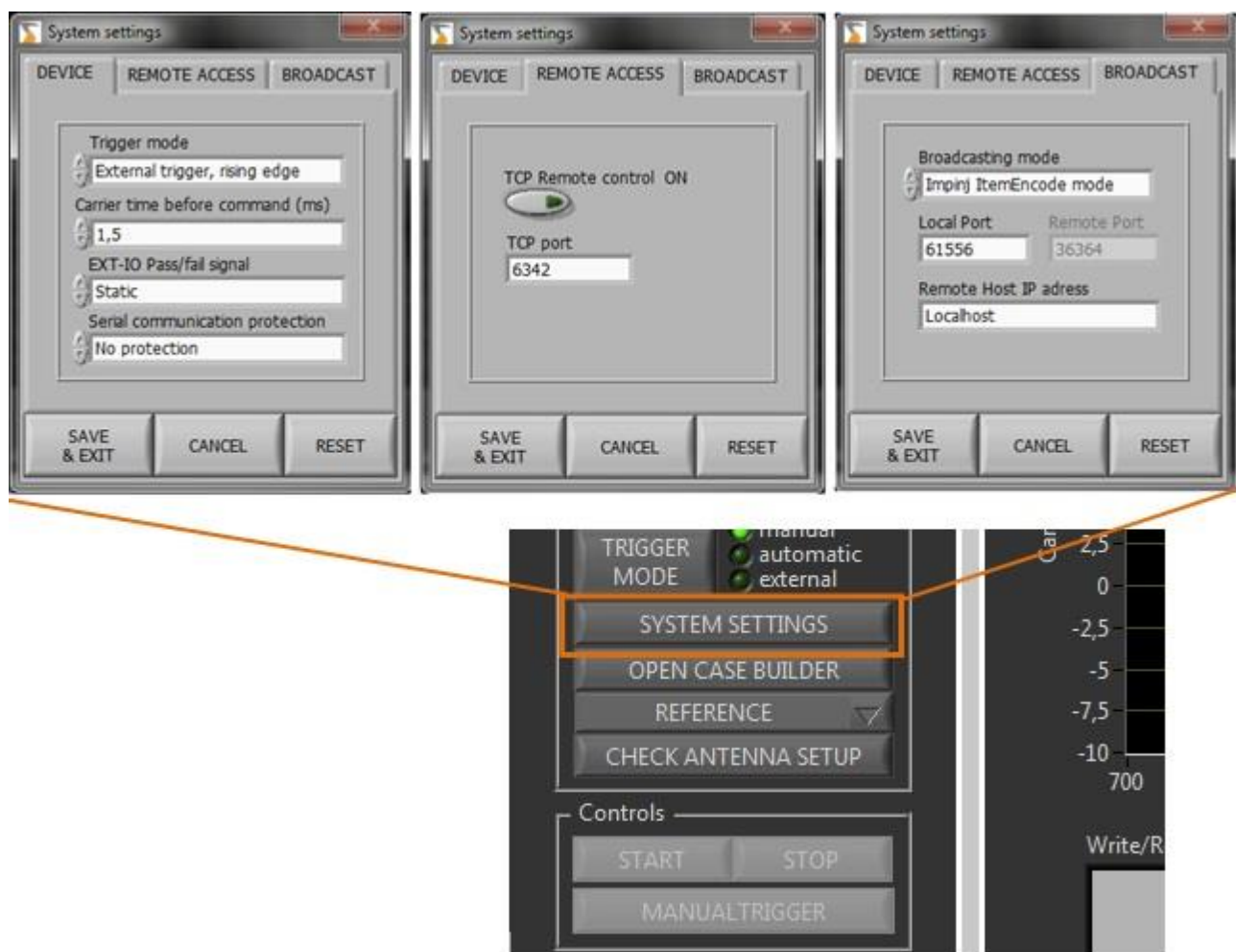


Fig 10 System settings menu

## 5.2.2 Device Settings

The device settings are used to configure the Tagsurance measurement unit signal input and output characteristics. Trigger mode, carrier before command, and I/O-port pass/fail signal settings are programmed to device memory and preserved there. Serial communication protection settings are not stored to device memory and will have to be reconfigured after each GUI startup, or defined in Tagsurance\_gui.exe program call (see appendix C).

**Trigger mode** defines the triggering options. In internal trigger mode, trigger is sent through the serial port. In external trigger mode, the signal is provided to the EXT-I/O port (pins 5 and 6), and measurement is triggered on the rising or falling edge of the signal, depending on the polarity setting. Default value: external trigger, rising edge.

**Carrier time before command** is the set time the Tagsurance Tester Unit will hold carrier high before sending a command to the tag. Default value: 1,5ms, which corresponds to the minimum time defined in Class 1 Gen 2 UHF RFID protocol standard (ISO 18000-6C).

**I/O port pass/fail signal** defines the form and polarity of the pass signal applied to the I/O port (Pin 8). In the static signal mode the signal remains in the state defined by the last test until a test with different result is performed. The pulse signal mode gives a 5ms pulse when the test is passed. By default, a passed test result is indicated by logic high (or a positive pulse). “**Failed = HIGH**” option changes this indication (pin 8) to fail resulting in a logic high (or a positive pulse). Signal waveforms are presented in a graph in Fig 7. Default value: static.

**Serial communication protection** setting enables/disables framing and checksum verification for measurement data transmission. When checksum is enabled, the Tagsurance will calculate an 8-bit sum over the test result data bytes and send that with results enabling GUI to detect data corruption, and missing bytes. When framesync is enabled, the Tagsurance test device will send a 4-bytes framesync sequence (0x23 0x23 0x23 0x21) with the test result data so that the GUI can recognize missing bytes and resynchronize data reception. Protection improves tolerance for serial data transmission issues. This is not a common issue, but may occur e.g. in a high interference environment or due to faulty operation of serial port adapter. Checksum transmission takes ~260µs and framesync transmission ~1ms. This affects minimum trigger interval, but not the case execution time. Default value: No protection.

## 5.2.3 Remote Access and Result Broadcast Options

The Tagsurance GUI can communicate with 3<sup>rd</sup> party applications through remote access interface (TCP) and result broadcast interface (UDP). TCP interface allows bidirectional communication between GUI and 3<sup>rd</sup> party master software. UDP interface is a one-directional interface used to broadcast data to 3<sup>rd</sup> party software. Options have to be enabled after GUI startup, or be defined in GUI program call (see Appendix C).

**TCP remote control** enables/disables remote control interface of the Tagsurance GUI. Having enabled this, the GUI operations can be controlled through TCP port, defined in **TCP port** field. Detailed instructions on how to apply TCP remote control interface are provided in Appendix A. Default value: disabled, port: 6342.

**Broadcasting mode** enables/disables test result broadcasting. Three broadcast modes are available, which all have to be accompanied with information on local transmission port, target IP address, and target port. Detailed information on the different modes is provided in Appendix B. Default value: Broadcast disabled.



### 5.3 Antenna Setup Check

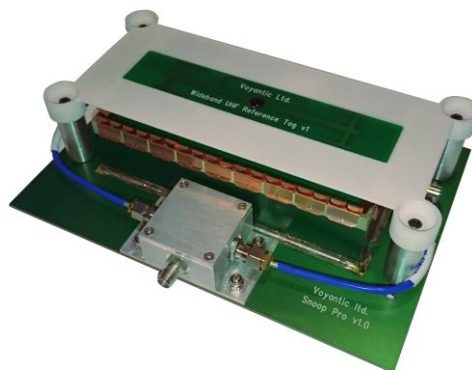
Installation of production test antenna, cabling, attachment of antenna shielding plate, and environmental factors may affect the characteristics of the carrier presented to the tag under test and induce bias to the measurement results. In order to minimize this, it is recommended to check the test setup by using a reference result measured with a reference tag.

#### GENERATION OF REFERENCE CURVE (with the Voyantic Reference Tag and plate)

1. Remove the shielding plate from the Snoop Pro Antenna
2. Attach the Voyantic Reference Tag by using the Voyantic Snoop antenna jig
3. Open the Tagsurance GUI
4. From the controls menu, click: "Reference", and choose: "Measure"
5. Wait until the measurement is finished
6. From the controls menu, click: "Reference", and choose: "Save"
7. When asked, define file path for the reference result data and click: "OK"
8. Now the reference result has been saved and can be loaded for use from the controls menu, by clicking: "Reference" and choosing: "Load"

#### ANTENNA SETUP CHECK AGAINST REFERENCE RESULT (with the Voyantic Reference Tag and plate)

1. Remove the shielding plate from the Snoop Pro Antenna
2. Attach the Voyantic Reference Tag by using the Voyantic Snoop antenna jig
3. Open the Tagsurance GUI.
4. From the controls menu, click: "Reference", and choose: "Load"
5. When asked, define file path for the reference result data and click: "OK"
6. From the controls menu, click: "Check antenna setup"
7. Visible on the screen are the reference curve (purple line), and the corresponding sweep curve measured with current setup which is updated every 2 seconds (yellow, dotted line). The amount of bias can be analyzed by measuring the difference between the curves.
8. Once the current setup adjustment is ready, click: "Pause" to stop updating the current setup curve. Then click: "Save setup" to create a file including reference curve data and current measurement result.
9. From the controls menu, click: "Stop", to return to basic view
10. Now the setup checking is done



**Fig 11 The Voyantic Reference Tag and plate on the Snoop Pro coupling element**

## 5.4 Performance Test Case Definition

### 5.4.1 Generic Guidelines and Important Limitations

#### 1. Frequency and Power Range

- The standard frequency range is: 860–960MHz, but it can be extended to 800–1100 MHz
- Power range is: -10 to +25dBm

#### 2. Amount of the Tasks Included in the Case

In order to be feasible, the case size must meet the following requirements:

- Test case definition shall not take more than 510 bytes, i.e.:  
$$(3 \times N_{\text{test\_points}} + 3) + 7 \times N_{\text{read\_tasks}} + \sum (8 + 2 \times N_{\text{words},j}) + 7 \times N_{\text{sweep\_tasks}} \leq 510,$$
  
where  $j=0 \dots N_{\text{write\_tasks}}$
- Test result data size shall not exceed 100 bytes, i.e.:  
$$\text{ceil}(N_{\text{test\_points}} / 8) + \sum (1 + 2 \times N_{\text{words},l}) + N_{\text{write\_tasks}} + \sum ((f_{\text{end},k} / f_{\text{start},k}) / f_{\text{step},k} + 1) \leq 100,$$
  
where  $l = 0 \dots N_{\text{read\_tasks}}$ ,  $j = 0 \dots N_{\text{read\_tasks}}$ , and  $k = N_{\text{sweep\_tasks}}$

#### 3. Time Consumption

Case execution time depends on the amount and type of tasks included. This can be estimated by using the equations below. Results are expressed in milliseconds (ms).

- Point test takes about:  $3,6 + \text{carrier\_time}$
- Read test takes about:  $(16,7 + \text{carrier\_time} + 0,5 \times N_{\text{word\_count}}) \times N_{\text{repetitions}}$
- Write test takes about:  $(35,4 + \text{carrier\_time} + 0,6 \times N_{\text{word\_count}}) \times N_{\text{repetitions}}$
- Sweep test time consumption depends on the tag and the sweep parameters.  
A rough estimate is given by:  $((f_{\text{end},k} / f_{\text{start},k}) / f_{\text{step},k} + 1) \times 5 \times 5\text{ms}$ .
- Sensitivity test takes about:  $2,7 + \text{carrier\_time} + (N_{\text{iterations}} - 1) \times (3,8 + \text{carrier\_time})$ ,  
where  $N_{\text{iterations}} = \text{ceil}(\log_2((\text{highest test level} - \text{lowest test level}) / \text{uncertainty})) \in [1,9]$

### 5.4.2 Creating Test Cases with the Test Case Builder

The Test Case Builder (a subprogram in the Tagsurance GUI) is an interactive tool that allows efficient generation of feasible performance test cases for production testing (Fig 12). The Test Case Builder is launched from the Tagsurance GUI control panel by clicking: “Open Case Builder”, and the changes made will be automatically occupied when returning to the main application, if not cleared before. The test case generated can also be saved as a file.

Tasks are added/modified/removed simply by using mouse buttons (see table below) while the tool automatically takes care of all license requirements and input/output data size limitations as well as estimates the maximum execution time for the test case. Clicking on the graph with left mouse button graph opens a dialog where the test type and parameters (Fig 13 and Fig 14) can be specified. The defined tasks are displayed as symbols on the graph (at corresponding frequency/power location), and they can be modified by dragging or by right clicking the respective symbol. Case preview field shows the complete test case in text format (exact case file format).

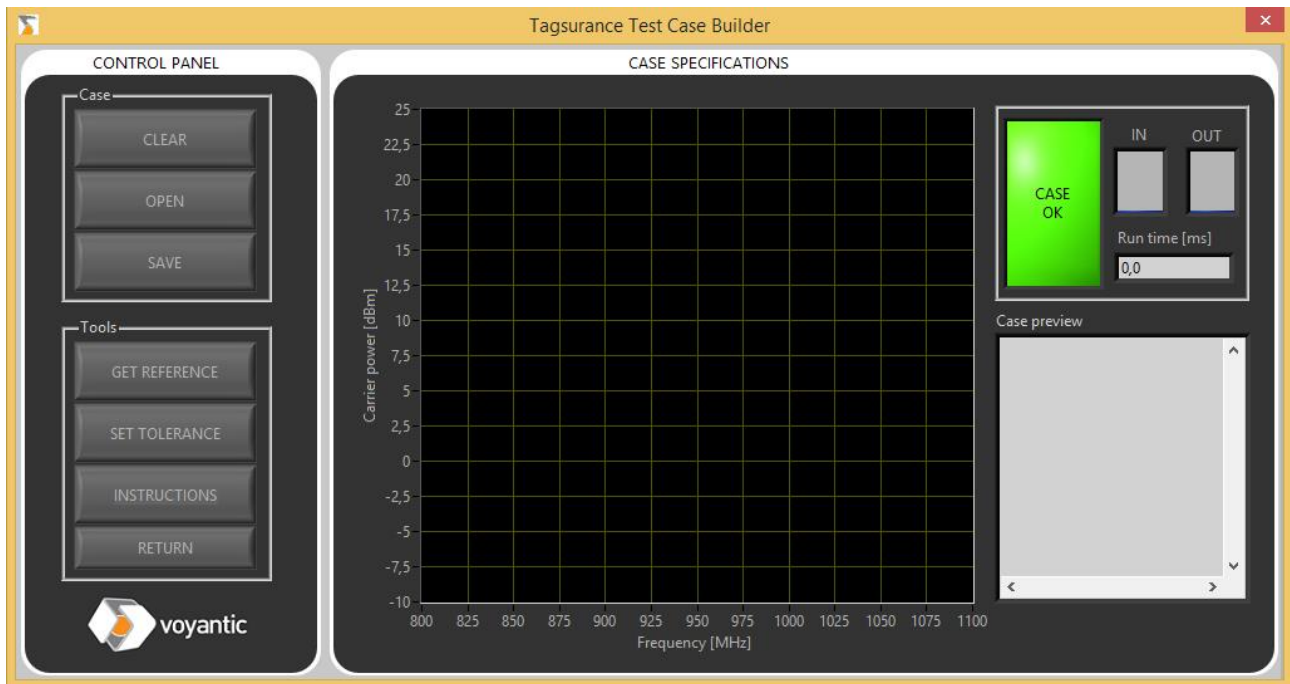


Fig 12 Test Case Builder is an interactive tool for generating test case files for the Voyantic Tagsurance UHF Tester

## CONTROLS

### BUTTONS

<b>Clear</b>	Clear current case data
<b>Save</b>	Save the current case to a file
<b>Open</b>	Open a case from a file
<b>Get reference</b>	Initiates response threshold test and draws the curve on the graph window
<b>Set tolerance</b>	Adjust tolerance conditions to define how many points test can be allowed to fail until the complete test case for a single tag is indicated to fail
<b>Instructions</b>	Open a help window for quick instructions
<b>Return</b>	Close the case builder application while keeping the current case in memory
<b>'x'</b>	Close the case builder application while keeping the current case in memory

### MOUSE

<b>Left click graph</b>	Add new task <sup>1,2</sup>
<b>Left drag and drop<sup>3</sup></b>	Moves the task appointed by the cursor Frequency and power specifications are modified accordingly
<b>Right click task<sup>3,4</sup></b>	Starts a dialog for removing or editing the task appointed by the cursor

<sup>1</sup> Test Case Builder does not allow tasks to be added which are not covered by the license terms. An error message is given if the task cannot be added.

<sup>2</sup> Execution order of tests in GUI is: points, write tests, read tests, sweep, and sensitivity tests.

<sup>3</sup> Sensitivity tests can be chosen by clicking on the lower edge of the vertical symbol bar, and sweep test by clicking the left-most end of the horizontal symbol bar.

<sup>4</sup> If there are multiple points near to the location clicked, GUI will choose the nearest. If the user cannot easily choose the test wanted, x- and y-scale can be modified to allow better visibility.

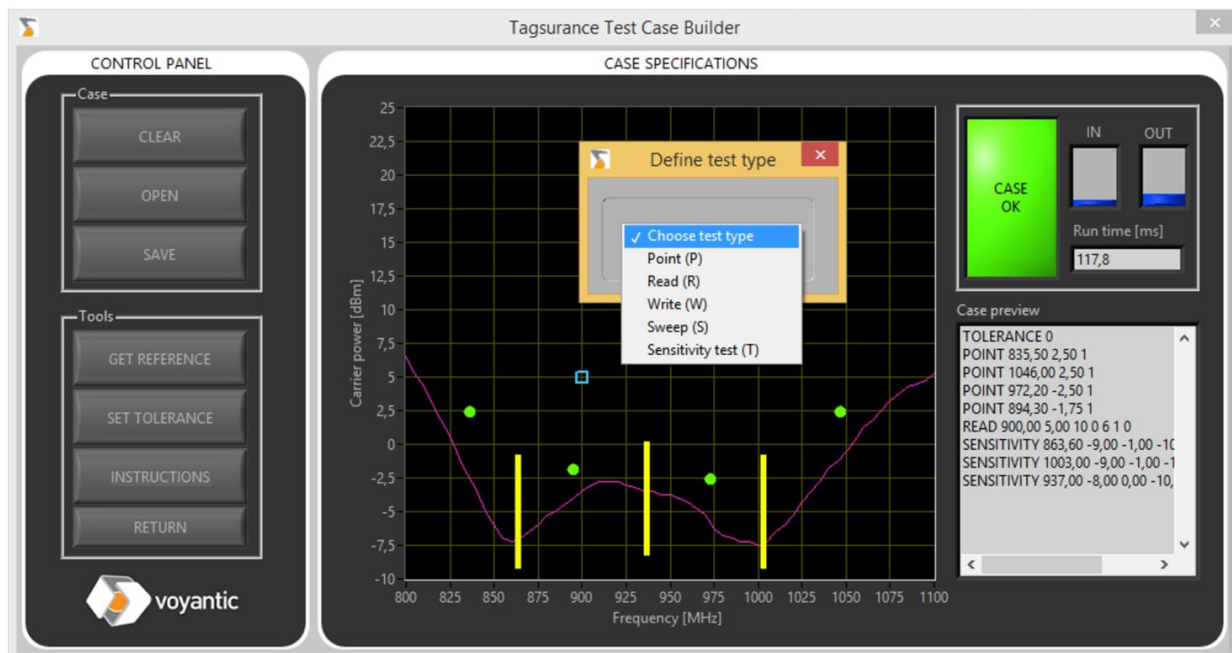


Fig 13 Test Case Builder view during test case creation

## INDICATORS

### Graph

Presented are test task indicators, and reference curve.

Frequency and power range can be adjusted by double-clicking the outmost values of the axes.

### LED

Green if case size is ok, red if the case is full

### IN

Size of the input data vector. Maximum is 510 bytes

### OUT

Size of the output data vector. Maximum is 100 bytes

### Run time<sup>1</sup>

An estimate of the maximum time the current test case takes to perform (ms)

### Case preview<sup>2,3</sup>

Contents of the automatically generated case file describing the current case

<sup>1</sup> Run time for write and sweep depends on tag and IC type. Accurate definition requires empirical testing.

<sup>2</sup> This corresponds to case text file format and syntax.

<sup>3</sup> Execution order of tests in GUI is: points, write tests, read tests, sweep, and sensitivity tests.

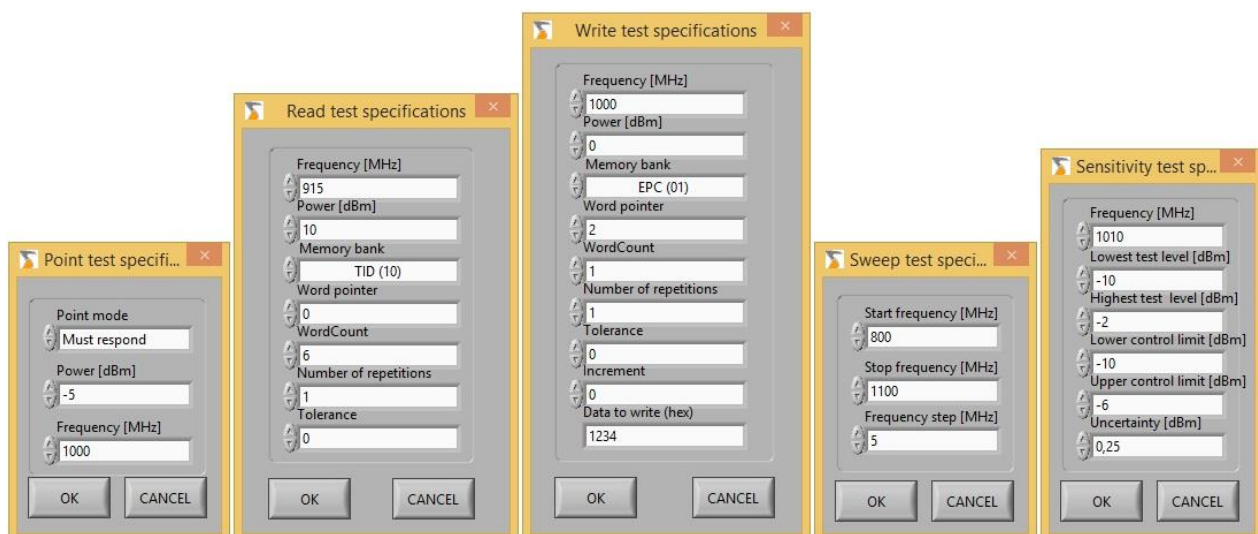


Fig 14 Specification of the test parameters is made through a dialogue

## TEST PARAMETERS

<b>Points Test</b>	<b>Point mode</b>	Defines if the tag tested should respond (must respond), should not respond (must not respond) to a command at the current frequency and power level, or if the result should not impact the overall pass/fail of the test sequence (indifferent)
	<b>Frequency (MHz)</b>	Defines the carrier frequency
	<b>Power (dBm)</b>	Defines the carrier power
<b>Read</b>	<b>Frequency (MHz)</b>	Defines the carrier frequency
	<b>Power (dBm)</b>	Defines the carrier power
	<b>Memory bank</b>	Defines the memory bank to be read (00, 01, 10 or 11)
	<b>Word pointer</b>	Defines the address within the memory bank (the word location from the beginning) where reading should start
	<b>Word count</b>	Defines the number of words read from the memory bank
	<b>Repetitions</b>	Defines how many times read command is executed
	<b>Tolerance</b>	Defines the amount of times the command may fail while the read task still is considered passed
<b>Write</b>	<b>Frequency (MHz)</b>	Defines the carrier frequency
	<b>Power (dBm)</b>	Defines the carrier power
	<b>Memory bank</b>	Defines the memory bank to be written (00, 01, 10 or 11)
	<b>Word pointer</b>	Defines the address within the memory bank (the word location from the beginning) where writing should start
	<b>Word count</b>	Defines the number of words to be written (max. 8 words) Note! Incremental counter is 32-bits wide
	<b>Repetitions</b>	Defines how many times write command is executed
	<b>Tolerance</b>	Defines the amount of times the command may fail while the write task still is considered passed
<b>Sweep</b>	<b>Increment</b>	Defines the increment for the written data per cycle (0 for static data, >0 for incremental data)
	<b>Data</b>	Defines the data to be written (HEX-format, e.g., 00ff)
	<b>Start Frequency</b>	Defines the start frequency in (MHz)
	<b>Stop Frequency</b>	Defines the stop frequency (MHz)
<b>Sensitivity Test</b>	<b>Frequency Step</b>	Defines the frequency step size (MHz)
	<b>Frequency</b>	Defines the carrier frequency (MHz)
	<b>Lowest test level</b>	Defines the lower bound for the iteration algorithm (dBm)
	<b>Highest test level</b>	Defines the upper bound for the iteration algorithm (dBm)
	<b>Lower control limit</b>	Defines the lower pass criterion (dBm)
	<b>Upper control limit</b>	Defines the upper pass criterion (dBm)
	<b>Uncertainty criterion</b>	Defines the uncertainty limit for the algorithm (dBm)

### 5.4.3 Creating Test Case Files Manually

Test case files can be created manually by writing a text file according to the following format.

#### Point Test:

TOLERANCE [\tab] [Tolerance]

POINT [\tab] [Frequency(MHz)] [\tab] [Power (dBm)] [\tab] [Point mode]

Where:

- [Tolerance] defines the number of points that may fail while the test is still not failed
- [Frequency(MHz)] defines the carrier frequency
- [Power (dBm)] defines the carrier power
- [Point mode] defines if the tag tested should respond (01), should not respond (10) to a command at the current frequency and power level, or if the result is indifferent and should not impact the overall pass/fail of the test sequence (00).

#### Read Test:

READ [\tab] [Frequency (MHz)] [\tab] [Power(dBm)] [Memory bank (00,01,10,11)]  
[\tab] [Word pointer(1-128, EBV)] [\tab] [Word count] [\tab] [Repetitions] [\tab] [Tolerance]

Where:

- [Frequency (MHz)] defines the carrier frequency
- [Power(dBm)] defines the carrier power
- [Memory bank (00, 01, 10, 11)] defines the memory bank to-be-read
- [Word pointer(1-128, EBV)] defines the address where the reading should start within the memory bank defined
- [Word count] defines the number of words read from the memory bank
- [Repetitions] defines the amount of times the read command is executed
- [Tolerance] defines the amount of times the command may fail the read task still is considered passed

#### Write Test:

BLOCKWRITE [\tab] [Frequency (MHz)] [\tab] [Power(dBm)] [\tab] [Memory bank (00,01,10,11)] [\tab]  
[Word pointer(1-128, EBV)] [\tab] [Repetitions] [\tab] [Tolerance] [\tab] [Increment] [\tab] [Word count]  
[\tab] [Data] [\tab]

Where:

- [Frequency (MHz)] defines the carrier frequency
- [Power(dBm)] defines the carrier power
- [Memory bank (00, 01, 10, 11)] defines the memory bank to-be-written
- [Word pointer(1-128, EBV)] defines the address where the writing should start within the memory bank defined
- [Repetitions] defines the amount of times the write command is executed

- [Tolerance] defines how many times the command may fail while the task still is considered passed
- [Increment] defines the increment for the data per cycle (Enter 0 for static data, >0 for incremental data)
- [Word count] defines the number of words to be written (max. word count is 8). Note! Incremental counter is 32-bits wide.
- [Data] defines the data to be written. Defined in HEX-format (e.g., 0x00ff).

#### Sweep Test:

SWEEP [\tab] [Start Frequency (MHz)] [\tab] [Stop Frequency (MHz)] [\tab] [Frequency Step (MHz)]

Where:

- [Start Frequency (MHz)] defines the start frequency
- [Stop Frequency (MHz)] defines the stop frequency
- [Frequency Step (MHz)] defines the frequency step size

#### Sensitivity Test:

SENSITIVITY [\tab] [Frequency (MHz)] [\tab] [Lowest test level (dBm)] [\tab] [Highest test level (dBm)]  
[\tab] [Lower control limit (dBm)] [\tab] [Upper control limit (dBm)] [\tab] [Uncertainty criterion (dBm)]

Where:

- [Frequency (MHz)] defines the carrier frequency
- [Lowest test level (dBm)] defines the lower bound for the iteration algorithm
- [Highest test level (dBm)] defines the upper bound for the iteration algorithm
- [Lower control limit (dBm)] defines the lower pass criterion
- [Upper control limit (dBm)] defines the upper pass criterion
- [Uncertainty criterion (dBm)] defines the uncertainty limit for the algorithm.

#### Example

TOLERANCE	0						
POINT	888,00	0,00	1				
READ	880,10	0,00	10	0	4	1	0
SWEEP	860	960	5				
SENSITIVITY	900	0	10	2	8	0,5	

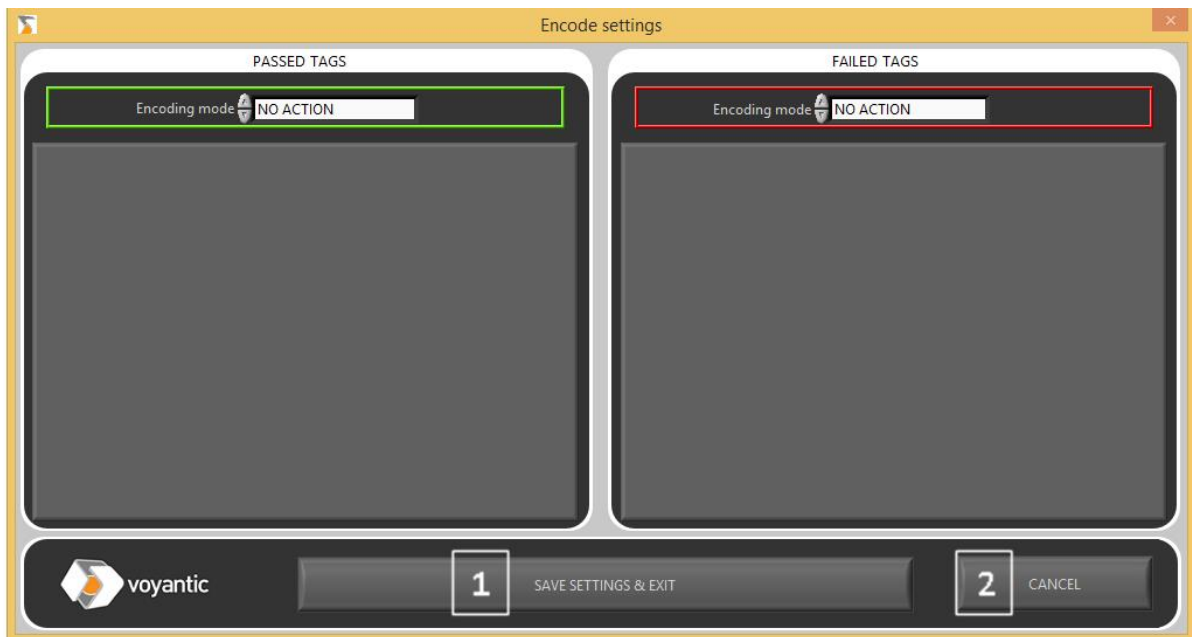
The example above will produce a test case which includes:

- One test point: (888; 0, must respond), which may not fail (tolerance=0)
- One read test: (880,10; 0, TID-bank, bytes: 0-8), which is executed once and is not allowed to fail
- One sweep test: (860 to 960 MHz at 5 MHz steps, start and stop frequencies included)
- One sensitivity test: (900MHz, range: 0-10dBm, LCL 2dBm, UCL 8dBm, 0,5dB uncertainty)

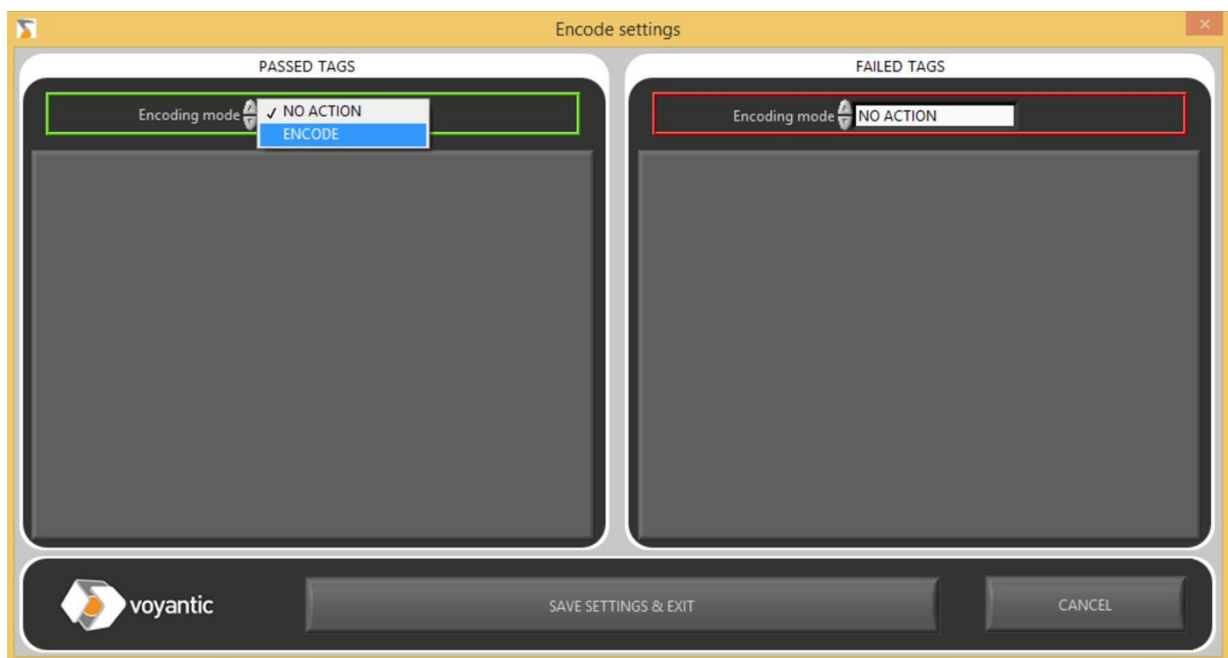
The created test case text file can be loaded from file similarly as any case created with the Test Case Builder.

## 5.5 Encode Options

Encode settings dialog is used to manage encode/kill specifications separately for passed tags (i.e. tags that have passed the performance test) and failed tags (i.e. tags that have failed the performance test). Allowed encoding modes are: no action and encode for passed tags; and no action, encode and kill for failed tags. Settings can be initiated to be used by clicking “Save settings & exit” [1]. If the button is disabled, the specifications are not valid. “Cancel” [2] will undo any changes to specifications. The specifications to be initialized are the ones visible on the dialog screen. Hidden items have no effect. By default, the action for both cases is: no action (Fig 15).



**Fig 15 Encode settings: Basic view**



**Fig 16 Encode settings: Enabling encoding**



Encode specifications are defined similarly for passed and failed tags. The mode of operation is selected by choosing “Encode” from the “Encoding mode” drop-down menu (Fig 16). This will open up a dialog page, where the details of the encode procedure are defined (Fig 17).

On the top of the page there are fields for **frequency**, **power**, and **access password**. Frequency and power specify the carrier parameters during communication. Adequate power level depends on the tag, the chip, and the frequency used. It is recommended to use a power level clearly above the response threshold. If this value is not known, it can be defined empirically by generating and running a performance test case with multiple write tests at different power levels (Fig 18). Access password is the current, earlier set, access password for the tag. By entering the access password, the Tagsurance is set to perform access procedure prior to the actual encoding. Access sequence is skipped if the access password is 0x00000000 (default).



Fig 17 Encode settings: Dialog page

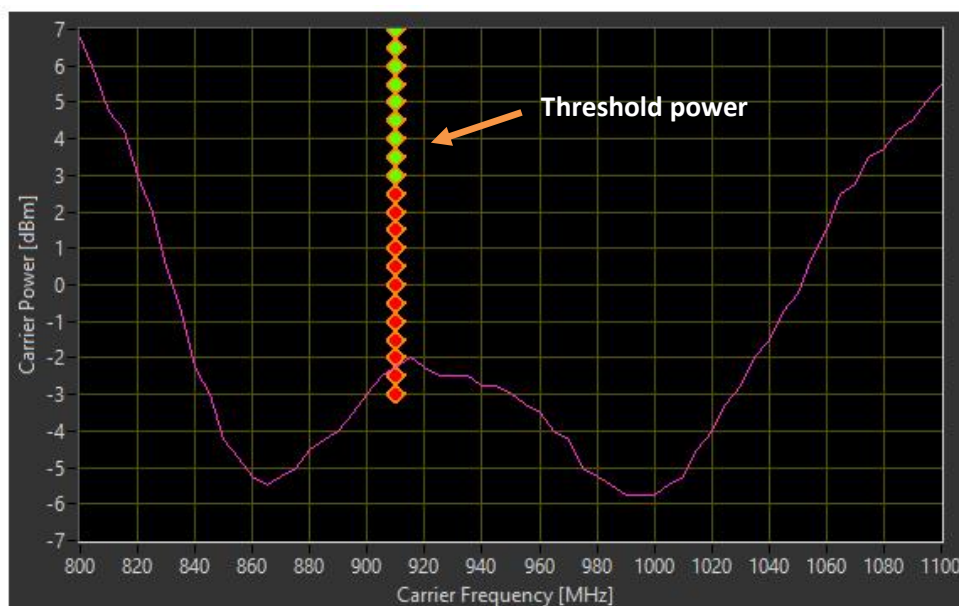
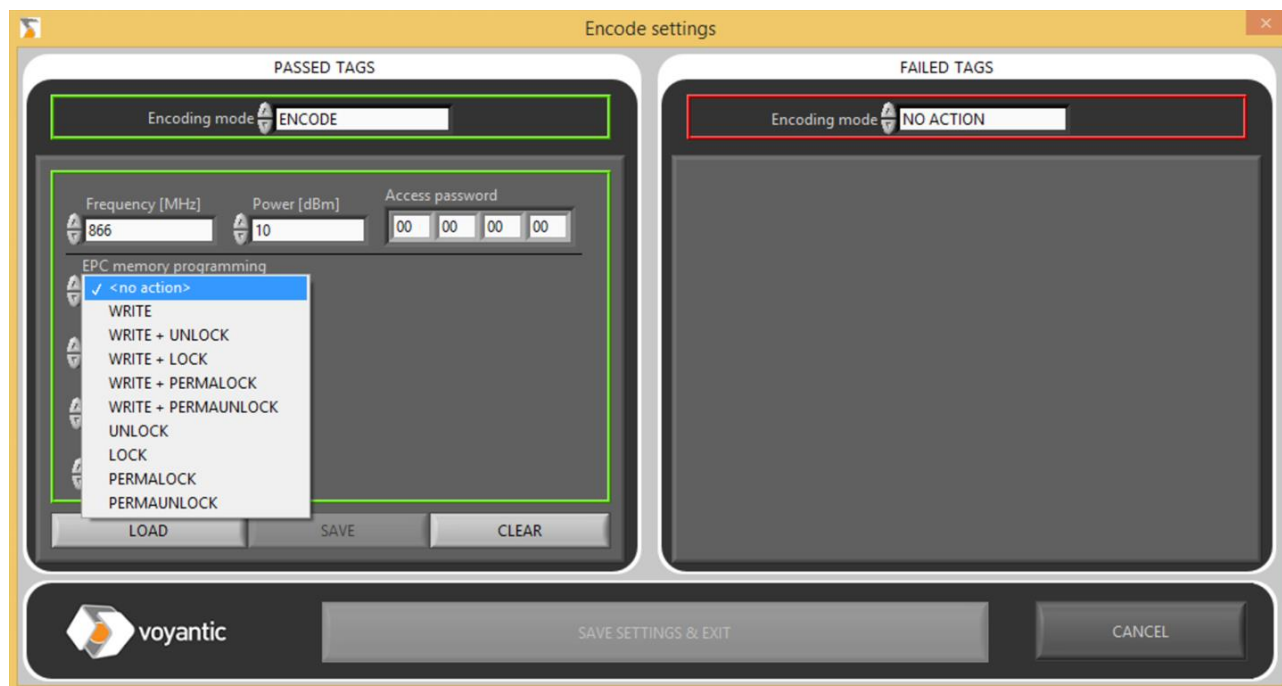


Fig 18 Defining threshold power level for writing

Next on the page are “**EPC memory**”, “**User memory**”, “**Kill-pwd**”, and “**Access-pwd**” **programming** menus. Actions for the different memories are selected by choosing a proper action from the respective menu. It is possible to: write data to memory, set lock bits, or do both (Fig 19).



**Fig 19 Encode settings: Selecting actions for each memory location**

If an action including a lock operation is chosen from any of the menus, encode action will set the lockbits for the tag memory location according to the table below. Unlock and lock only affect the password lock bit, while permalock and permaunlock will also change the permalock bit.

Locking mode	PWD-write(/read)	Permalock
Unlock	0	<no change>
Lock	1	<no change>
Permalock	1	1
Permaunlock	0	1

In case an action including write is chosen from either EPC or User memory programming menus, more fields to enter the data will appear (Fig 20). **Word pointer** field defines the starting address for the write operation (EBV format, 0-127). The data is entered through a dialog opening from “←”(Fig 20, [1]). In the dialog, data can be imported from a text file, copy pasted, or fed manually item-per-item (Fig 21). Data should be provided in HEX format (0-9, A-F, a-f). After the data to be encoded has been defined, **Word count** and **Codes** show the word count programmed per tag and the number of different items in the data list, respectively.



Fig 20 Encode settings: Defining encoding



Fig 21 Encode settings: Adding the data to be encoded

If an action including write is chosen from either Kill-pwd or Access-pwd programming menus (Fig 22), a field for entering the four password bytes will appear (Fig 23). Data should be provided in HEX format (0-9, A-F, a-f), MSB first. The default value for both passwords is 0x00000000.

After the fields have been filled properly, “Save settings & exit” is enabled, and it is possible to initiate encoding. In addition, it is also possible to save specifications to a file and load ready-made cases by clicking “Load” or “Save”. “Clear” will clear the data entered and restore default values.

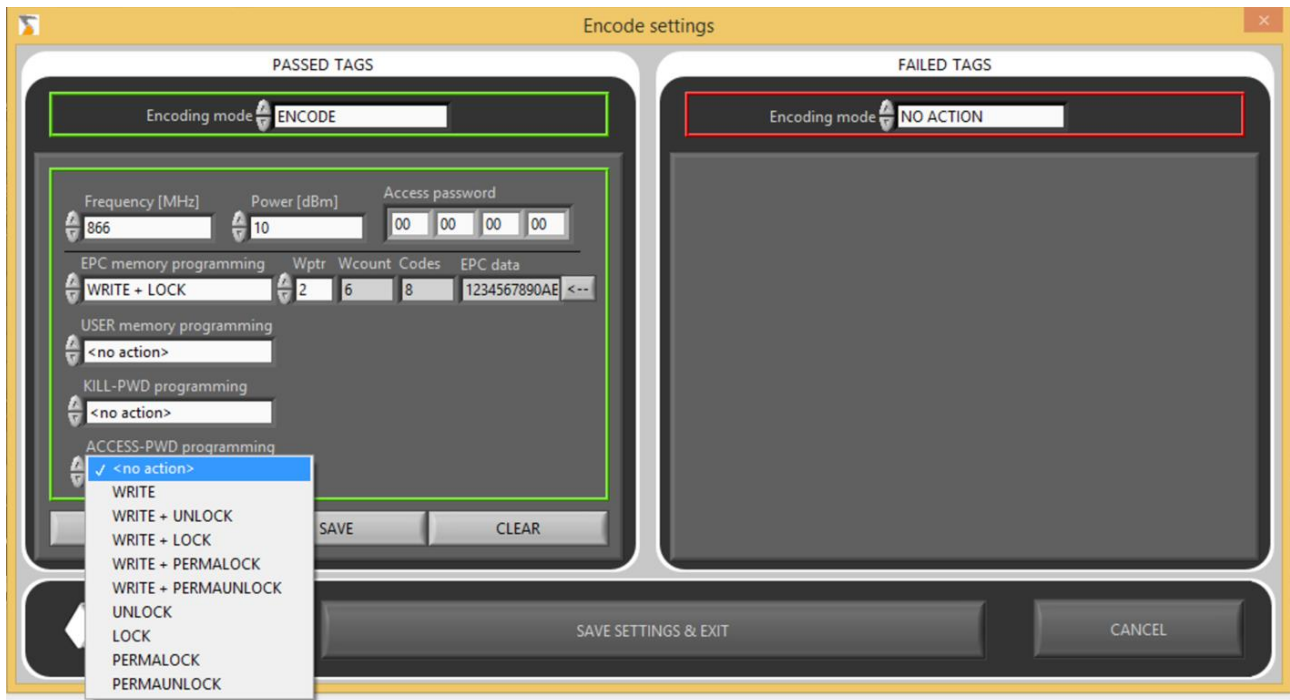


Fig 22 Encode settings: Programming passwords



Fig 23 Encode settings: Entering the new password

Kill operation is only allowed for failed tags, and it can be selected by choosing “Kill” from the “Encoding mode” menu under the header “Failed tags”. This will open a dialog page showing the parameters to be defined for the kill command (Fig 24, Fig 25). **Frequency** and **Power** specify the carrier parameters during communication. As for the encoding, adequate power level for the kill depends on the tag, the chip, and the frequency used.

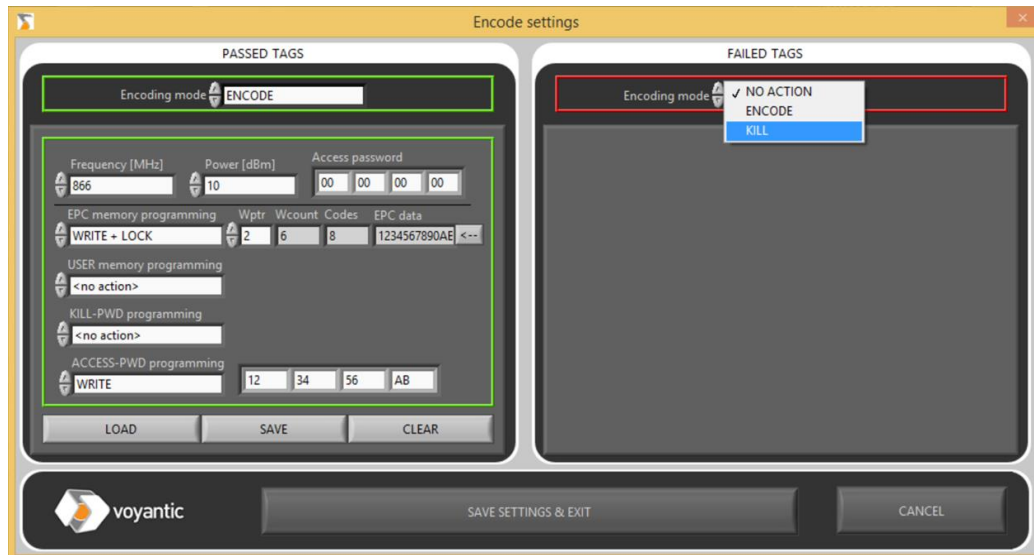


Fig 24 Encode settings: Choosing the kill command for failed tags

If the kill password is known and non-zero, the Tagsurance will conduct the kill procedure normally using the entered kill password. In this case access password is not needed. If the kill password is zero, or unknown, the Tagsurance will first overwrite the current kill password (with 0x11111111), and then conduct the kill procedure. This is necessary, because killing can only be done with a valid non-zero password. If kill password is password locked, rewriting it will require access with a valid access password. In this case it should be entered in the access password field. If the memory is not locked, access password can be left as 0x00000000, which will cause the access procedure to be skipped.



Fig 25 Encode settings: Entering the earlier set kill and access passwords to enable tag killing

## **5.6 Measurement Performance and Monitoring**

### **5.6.1 Measurement Preparation and Basic Execution Process**

Before starting the measurement, it is recommended to:

1. Check the system settings (for instructions, see chapter 5.2)
2. Check the antenna setup (for instructions, see chapter 5.3)
3. Check the EXT-IO connections (for instructions, see chapter 4.4)

When the test setup is ready, measurements can be performed by following the next steps:

1. Load a test case (for instructions, see chapter 5.1)

Test case can be loaded either from a file or the device memory. To select the case to be used, choose the right instance from the 'Load Case' dropdown menu. An error message is generated if the case loaded is not valid. Otherwise, 'Start Case' -button is enabled.

2. Define encoding options (optional, for instructions, see chapter 5.5)

Load a predefined encoding specification and data or create one in the encoding settings.

3. Run a test case

Click 'Start Case' button to run the active case.

4. Apply trigger (not needed in automatic trigger mode)

The device does not accept a new trigger signal while the test case is running. The concurrent test rate is presented on the indicator at the lower right corner of the application window.

5. Stop case

Click 'Stop Case' button to stop the case.



## 5.6.2 Data Output: Basic View

Before the test has been started, the test case determined is shown in the case data section. The graph [1] in the Fig 26 shows three (3) sensitivity tests on point frequency, four (4) test points and a read function determined. The must respond test points are shown as green rectangle pointers, the must not respond test points are shown as red rectangle pointers and the sensitivity tests are shown as yellow lines with the height of defined power range on the defined frequency. The specified write and read tasks are shown in the Write/Read data field together with sensitivity test specifications [2].

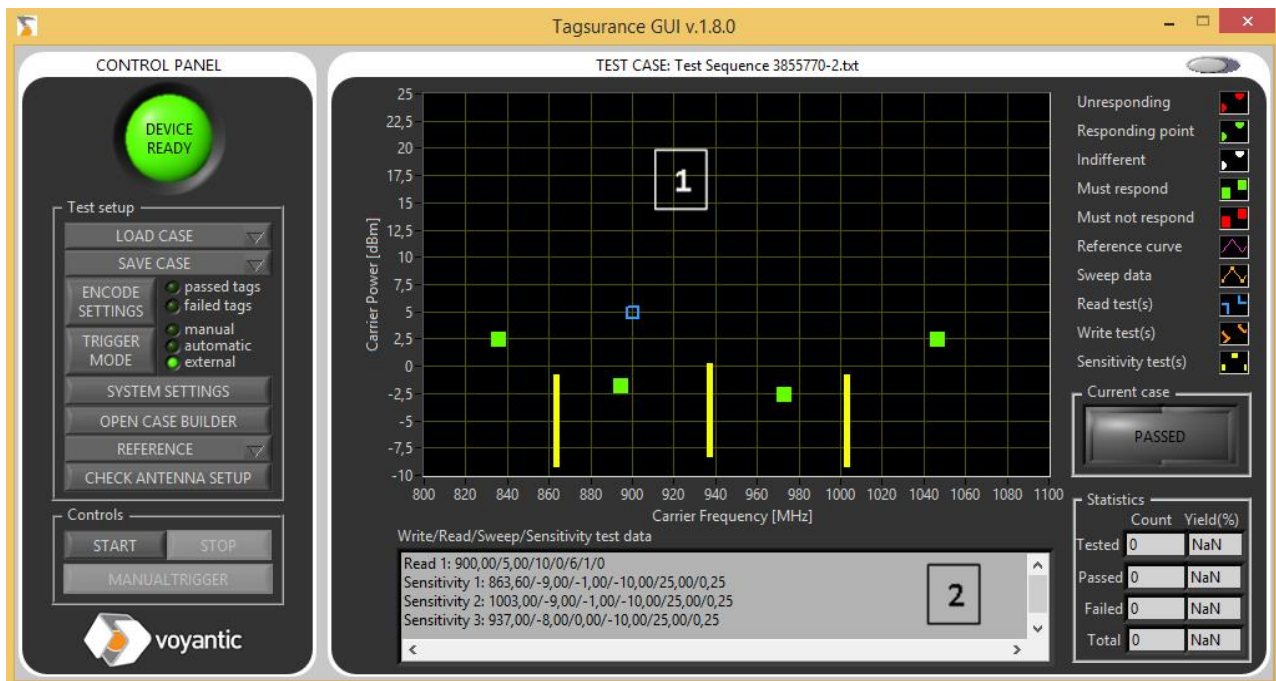


Fig 26 Application software front panel: Before starting the measurements

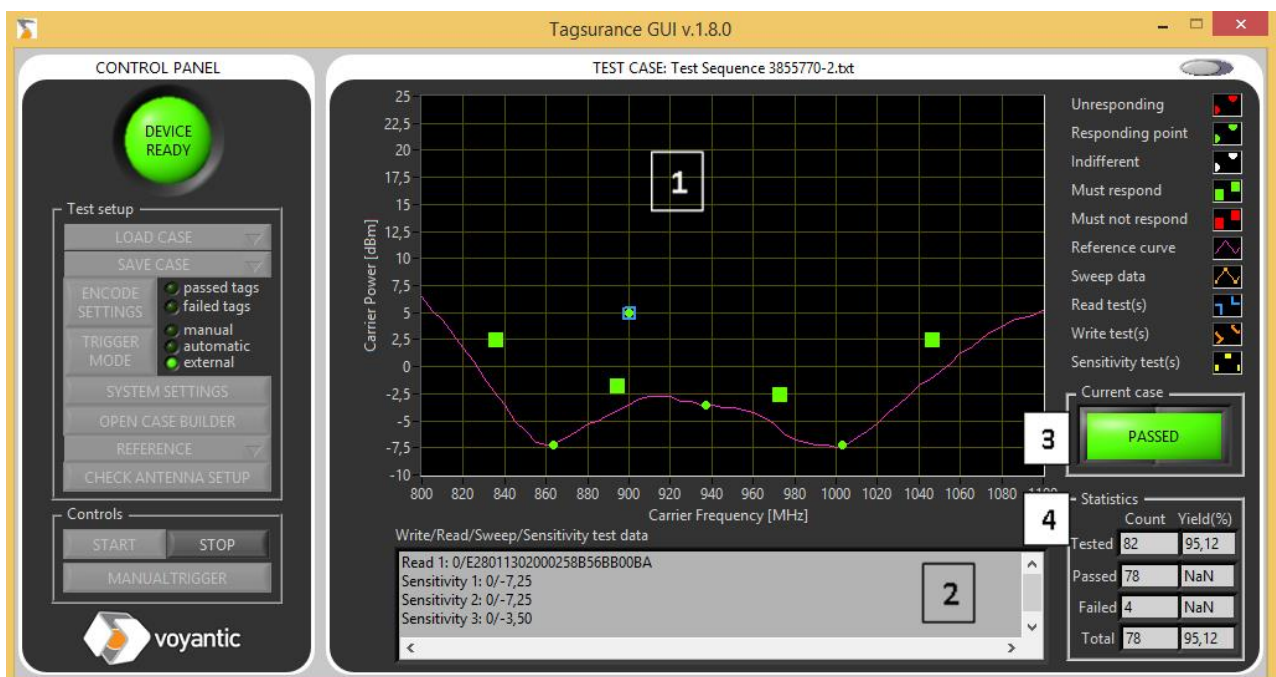


Fig 27 Application software front-panel: During ongoing measurement

Once a test case is started (Fig 27), the test results from each test run are shown on the graph [1] with additional information which is provided in the text field [2].

**Point Test results** are displayed on the graph. By default, results are indicated with color-coded round dots, and point conditions with squares. These settings can be changed from the graph legend menu.

**Read Test results** are presented in the window below the chart in a format: error/data. If the error code (see table below) is different from 0 (i.e. the tolerance condition was not met), the data is presented as 00's. On the graph they are presented by blue squares with green/red dot in the middle (pass/fail).

**Write Test results** are presented in the window below the chart in a format: error/data written. If the error code (see table below) is different from 0 (i.e. the tolerance condition was not met), the task was failed. On the graph they are presented by brown diamonds with green/red dot in the middle (pass/fail).

INTERPRETATION OF READ/WRITE TASK ERROR BYTE			
0x00	0 <sub>10</sub>	No errors	Tolerance condition was met
0x01	1 <sub>10</sub>	Connection failed	Tag did not respond to query
0x02	2 <sub>10</sub>	Command failed	Tag did not respond to the command
0x03	3 <sub>10</sub>	Inventory failed	Tag responds to query, but inventory failed
0x04	4 <sub>10</sub>	Tag error message 1	Other error (not covered by the other tag error codes)
0x34	52 <sub>10</sub>	Tag error message 2	Memory overrun (too high memory address)
0x44	68 <sub>10</sub>	Tag error message 3	Memory locked (memory cannot be read/written)
0xB4	180 <sub>10</sub>	Tag error message 4	Insufficient power (carrier level too low)
0xF4	244 <sub>10</sub>	Tag error message 5	Non-specified (error-specific codes not supported by the tag)

**Sweep Test results** are presented on the graph. By default, the threshold curve measured is presented with x's at the measurement points.

**Sensitivity Test results** are presented in the window below the chart in a format: error/data. If the error code is 0, the test was passed. Non-zero error indicates that the LCL/UCL criterion was not met. On the graph they are presented by a green/red dot (pass/fail).

INTERPRETATION OF SENSITIVITY TEST TASK ERROR BYTE			
0x00	0 <sub>10</sub>	No errors	Tolerance condition was met
0x01	1 <sub>10</sub>	Out of tolerances	Tolerance condition was not met, the threshold power exceeded the upper control limit or was below the lower control limit
0x02	2 <sub>10</sub>	Out of range	UCL and LCL are not defined inside the measurement range and the response power is out of measurement range.

**Encoding process outcome** is indicated in the test data field after the read data and test results. In case of an error during encoding, the exact step of the communication, where the error has occurred, is given together with the error code.



**Current case** field [3] indicates whether the tested tag has met the specified test conditions and was encoded successfully.

**Statistics** field [4] presents a summary of all the tested cases giving the count and yield for the tested cases and the amount of tested tags per second. This data are visible in real-time during the testing. The statistics give the count of tested tags and the performance test yield. On the second row the amount of tags that passed the performance test is given together with the yield of the encoding of the passed tags. The same information for the tags that have failed the performance test is given on the third row. On the last row there is the total amount of tags that have passed the performance test and have been successfully encoded after that (if the encoding has been specified), together with the final combined yield of testing and encoding.

Statistics		
	Count	Yield(%)
Tested	12	83.33
Passed	10	100.00
Failed	2	100.00
Total	10	83.33

Fig 28 Statistics on test, encoding and overall yield

The default plot settings can be modified and the data extracted from the menu shown in Fig 29. The plot settings menu is opened by right clicking the legend menu on the right. Axes can be modified by entering the minimum and maximum values of the range. This can be done by double clicking the outmost values in the graph and entering the desired values.

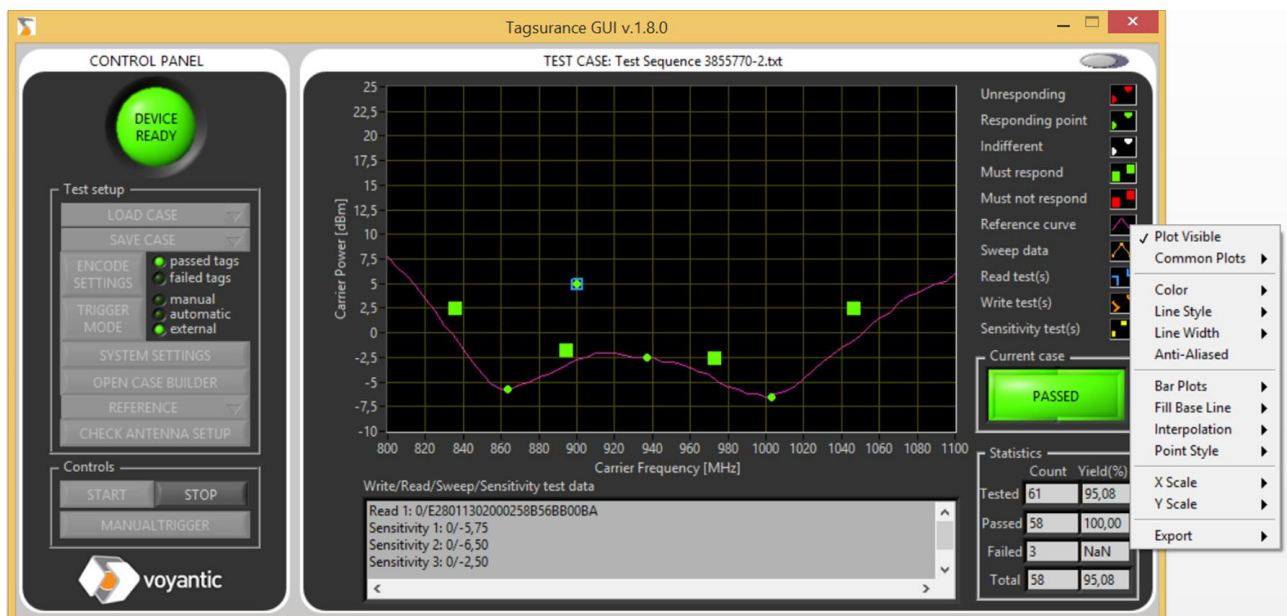


Fig 29 Plot and axis settings

### 5.6.3 Data Output: Statistical View

If sensitivity tests are included in the test case applied, the measured threshold powers on point frequency can be viewed in a statistical view shown in the figure below. Switching to the statistical view is done by clicking the button in the upper right corner of the user interface [Fig 30, 5]. In this view the user is able to determine three different frequencies from the defined sensitivity test frequencies in the test case to be monitored at the same time. The variance data of the overall batch is shown in the left-most graph and the real-time threshold power data for each measured tag is shown in the right-most graph.

The amount of measured samples in the graph can be adjusted [6] and after the test has been stopped, the data can be viewed in sections by scrolling [7]. The upper and lower control limits specified in the case file are shown on the right under each defined frequency [8]. The percentages of the samples with threshold power above the upper control limit and the samples with threshold power below the lower control limit are given next to the limits. Also the average threshold power and the standard deviation of the threshold power for each frequency are calculated.

If the threshold power level exceeds the upper control limit or goes below the lower control limit, the green led indicator on the left side of the frequency indicator will be lit. The indicator will turn back to green with a measurement result between the control limits.

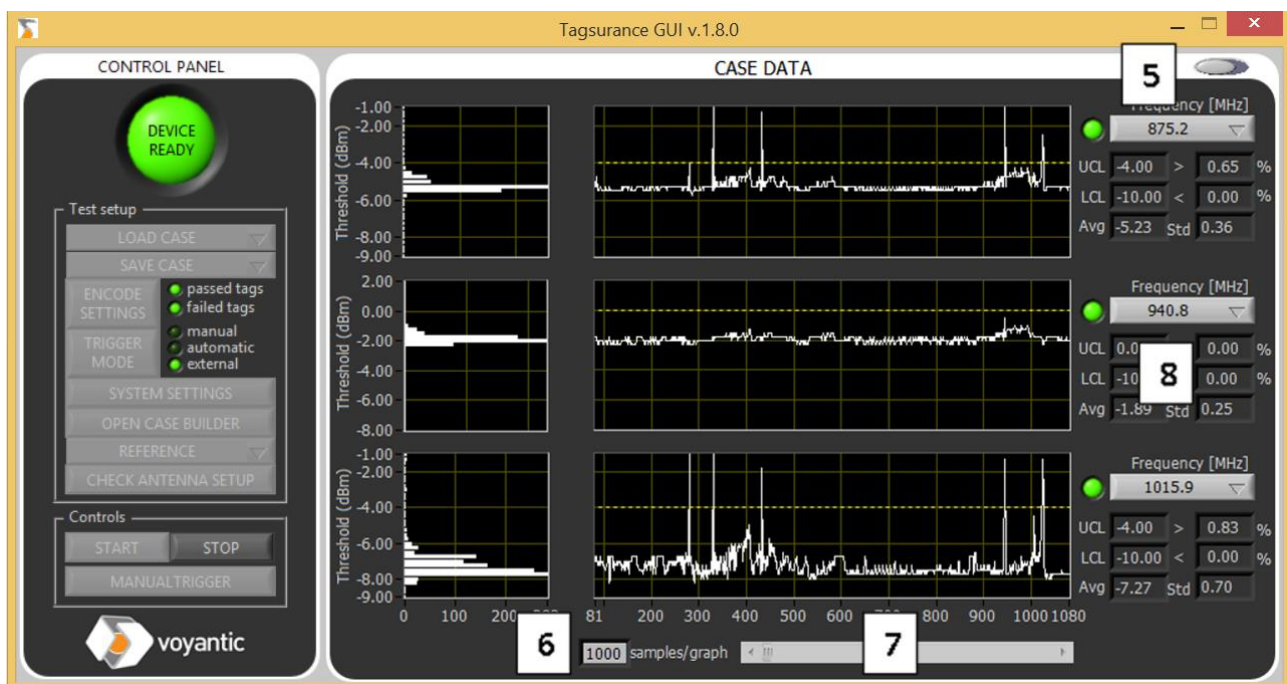


Fig 30 Application software front-panel, statistical view

## 5.6.4 Measurement Log Files

Measurement data are also saved to a log file, which can be used for achieving and performing analysis (an example is presented in Fig 31). The file contains: date, time, serial number of the Tagsurance UHF Tester used in the measurements, task specifications, statistics, and results from each test performed. In addition, the test case file path is documented in the log file, if the case has been loaded from a specific file. Also, the antenna setup check data file path is shown, if the antenna setup check procedure has been performed after the latest GUI startup.

If encoding after testing is defined, a separate log file for the encoding part of the process is generated, too (an example is presented in Fig 32). The fail contains: date, time, encoding specifications, statistics, the exact data encoded on each tag or definition of the error occurred.

The target path for both log files is prompted at the beginning of the test session, but to ensure safe data integrity, temporary files are used for logging the data while the measurement is on. Only after the measurement has ended, the temporary log data is copied to the target file(s). Erroneous operation is indicated by an error message pop-up which advises how to restore results (not in TCP mode).

A	B	C	D	E	F	G	H	I	J
1	Tagsurance	27.11.2014	16:18						
2	Test unit:	UT-0024							
3	Test setup:	C:\Tagsurance\Data\antenna_setup_check_data_20141127.txt							
4	Test case:	C:\Tagsurance\Data\Test case files\test_case_example.txt							
5									
6	Read specifications	Frequency [MHz]	Power [dBm]	Bank	Start adress	Word count	Repetitions	Tolerance	
7	Read 1	866	9,5	TID (10)	0	6	2	0	
8									
9	Sensitivity test specifications	Frequency [MHz]	Lowest tested power [dBm]	Highest tested power [dBm]	Lower control limit [dBm]	Upper control limit [dBm]	Uncertainty criterion [dBm]		
10	Sensitivity 1	832	-10	10	-10	2,5	0,25		
11	Sensitivity 2	974,8	-10	0	-10	-5	0,25		
12									
13	Case specifications (Point tolerance: 0)								
14	Frequency [MHz]		866	780,9	880	951,7	889,1	1013,2	832
15	Power [dBm]		9,5	1	-1,5	-6,25	-6,5	-6,5	974,8
16	Mode	Read 1	must respond	must respond	must respond	must not respond	must not respond	Sensitivity 1	Sensitivity 2
17									
18									
19	Results (Tags tested: 12, Yield: 91,67%)								
20									
21	Time stamp	Pass/Fail	Read data (err/data)	Tested points				Sensitivity 1	Sensitivity 2
22	16:18:21	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
23	16:18:21	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
24	16:18:24	FAIL	0/E2801130200021485989019	0	0	0	0	0 1/5,50	1/0,00
25	16:18:27	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
26	16:18:27	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
27	16:18:32	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
28	16:18:32	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
29	16:18:35	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
30	16:18:36	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
31	16:18:37	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
32	16:18:37	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
33	16:18:39	PASS	0/E2801130200021485989019	1	1	1	0	0 0/-1,25	0/-7,75
34									

Fig 31 Example of test log data output file

	A	B	C	D	E	F	G	H	I	J	K
1	Tagsurance	27.11.2014	16:18								
2											
3	Performance test	Encoding	Frequency [MHz]	Power [dBm]	Access password	EPC memory programming	User memory programming	New kill Password	New access password	Lock bits	KILL
4	PASS	Encode	866		10 0x00000000	2.2.2001	-	-	0x11111111	-----	-
5	FAIL	Encode	866		10 0x00000000	2.2.2001	-	-	-	-----	-
6											
7	Results	Performance test	Passed tags encoding	Failed tags encoding							
8	Count	12	11	1							
9	Yield [%]	91,67	100	100							
10											
11	Time stamp	Performance test	Passed tags encoding	Failed tags encoding	Inventory (+access)	EPC memory programming	User memory programming	Kill password programming	Access password programming	Lock	KILL
12	16:18:21	PASS	PASS	-	0	11111111	-	-	0	-	-
13	16:18:21	PASS	PASS	-	0	11111111	-	-	0	-	-
14	16:18:24	FAIL	-	PASS	0 ffffffff	-	-	-	-	-	-
15	16:18:27	PASS	PASS	-	0	11111111	-	-	0	-	-
16	16:18:27	PASS	PASS	-	0	11111111	-	-	0	-	-
17	16:18:32	PASS	PASS	-	0	11111111	-	-	0	-	-
18	16:18:32	PASS	PASS	-	0	11111111	-	-	0	-	-
19	16:18:35	PASS	PASS	-	0	11111111	-	-	0	-	-
20	16:18:36	PASS	PASS	-	0	11111111	-	-	0	-	-
21	16:18:37	PASS	PASS	-	0	11111111	-	-	0	-	-
22	16:18:37	PASS	PASS	-	0	11111111	-	-	0	-	-
23	16:18:39	PASS	PASS	-	0	11111111	-	-	0	-	-
24											

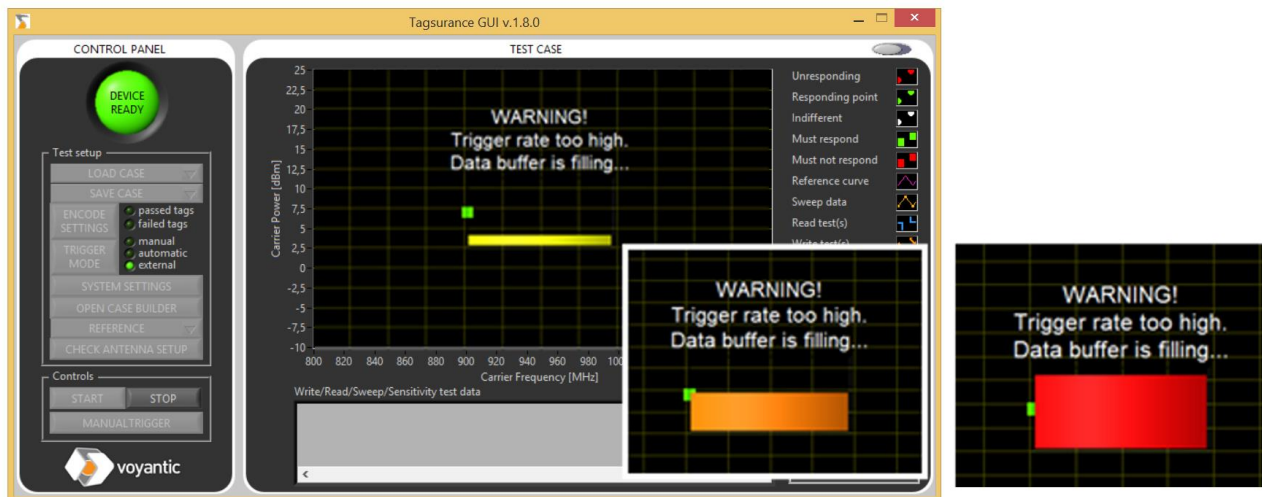
Fig 32 Example of encoding log file

## 5.7 Error Handling

The software is built to recover from many errors that may occur between and during the measurement. However, if something unexpected happens, the software will display an error dialog, which shows the possible error code, explains it and prompts the user for further actions.

INTERPRETATION OF SYSTEM ERROR CODE(S) (0 -> no error, 1 --> error)		
Bit 0	invalid input data sequence	valid range: 1 to 510 bytes
Bit 1	timeout during data reception	maximum delay between data bytes: 1000ms
Bit 2	invalid memory bank	valid range: 0 to 4
Bit 3	invalid word count	valid range: >0 (read test), 1-8 (write test)
Bit 4	invalid command characters	valid values: 'P', 'R', 'W', and 'S'
Bit 5	output data size required too large	valid range: 0 to 100 bytes
Bit 6	invalid frequency	valid range: 860 to 960MHz (extended: 800 to 1100MHz)
Bit 7	Invalid power	valid range: -10 to +25dBm
0xFF	license error	requested task not covered by current license options

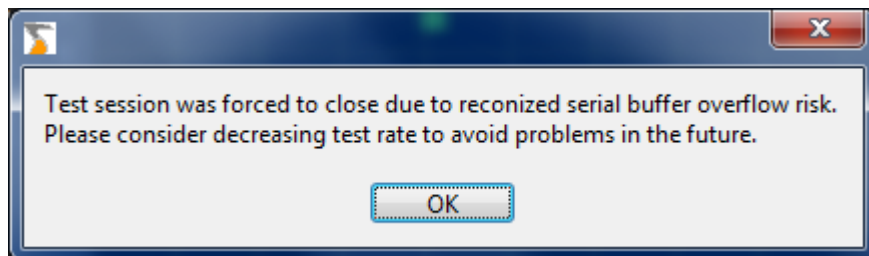
In case of a very short cycle time (~>15-20 tags/second), depending on the performance of the PC running the Tagsurance GUI, it is possible that the test data is sent by the Tagsurance faster than the PC can receive and the Tagsurance GUI process. This will first show as the test data view lagging from the actual results (which can be checked from the front panel LEDs). The I/O signals and the front panel LEDs will always give the pass/fail results in real-time, even if the software would be lagging because of the data buffering.



**Fig 33 Serial buffer filling**

The warning will appear in the user interface when the serial buffer level is 10% filled. The buffer is large enough to contain thousands of tag results, so in this point the buffer filling can be seen as a significant delay in the user interface visual presentation of the results. When the process is slowed down and stopped, for example for adding raw material etc., the software will keep running and emptying the buffer. If the test case is stopped when there are still data in the buffer, this data will be lost.

The yellow bar indicates the buffer fill level and it will first turn to orange and then to red. If the buffer level reaches 90%, the program will close the test session to avoid buffer overflow and the below message will appear.



**Fig 34 Serial buffer overflow risk**

In case of very fast testing, where the serial buffer overflow risk is possible, gathering the data log with Tagsurance GUI is still possible. To enable this, the Tagsurance GUI should be minimized to the Windows task bar. This way the software does not use the computing power for refreshing the visual interface and is able to write the log file without the buffer being filled. The Tagsurance device will always provide pass/fail information in real time through I/O signal port and also visually in front panel with LED lights.

## 6 Operation Using the Serial Command Interface

### 6.1 Communication Protocol

When the Tagsurance UHF Tester is turned on, it first performs initialization and then moves to the IDLE state. The process takes less than one second and the device is then ready to accept commands from the serial port (Fig 35). IDLE state is indicated by setting the ready led in the device from panel on and the busy pin in the external connector low. When interfacing the device directly from the serial port, the READY pin can be used to detect when the initialization is complete.

The serial command interface provides commands that can be used to perform measurements (inline testing, response threshold measurements, and self-checking), or to modify system settings. The different functions can be initiated when the device is in the IDLE mode, and it is also where it returns when the requested function is successfully performed (see Fig 35).

#### RS-232 serial port settings:

Baud rate: 38400kbps, Data: 8 bits, no parity, LSB first, Interpretation: ASCII format

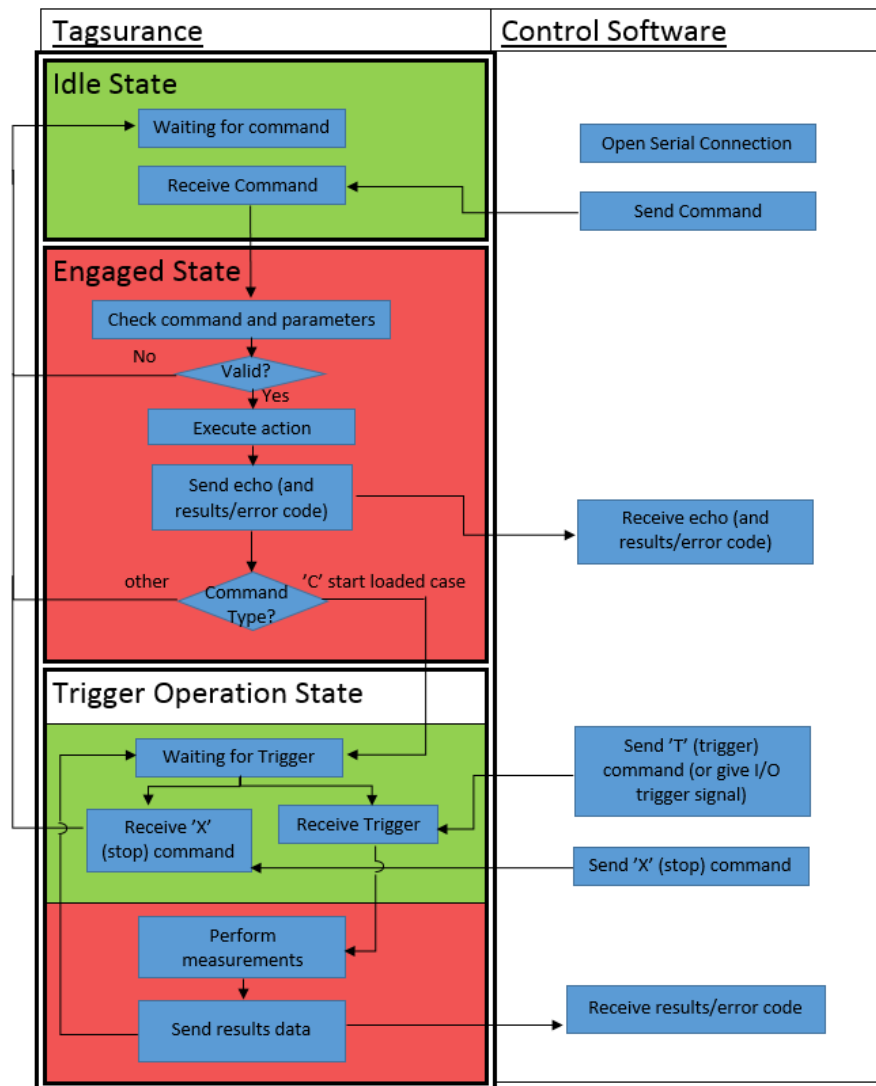


Fig 35 Basic operation cycle of the Voyantic Tagsurance UHF Tester

## 6.2 System Configuration Commands

System settings can be modified through the serial interface.

SYSTEM CONFIGURATION			
Function	Command	Response	Next state
Enable external trigger (default)	'PTE'	(none)	IDLE
Disable external trigger	'PTI'	(none)	IDLE
Set trigger (ext) to rising edge (default)	'PTH'	(none)	IDLE
Set trigger (ext) to falling edge	'PTL'	(none)	IDLE
Enable internal gen2 chip	'POC'	(none)	IDLE
Disable internal gen2 chip (default)	'POA'	(none)	IDLE
Set carrier time (default: 480=[1][224]=2,5ms)	'PC'[time_high][time_low]	(none)	IDLE
Set pass(/fail) signal to static mode (default)	'PIPS'	(none)	IDLE
Set pass(/fail) signal to pulsed mode	'PIPP'	(none)	IDLE
Enable "failed = HIGH" option	'PIPM'	(none)	IDLE
Disable "failed = HIGH" option (default)	'PIPN'	(none)	IDLE
Save settings	'PFS'	(none)	IDLE
Load saved settings	'PFL'	[fbyte <sub>L</sub> ][fbyte <sub>H</sub> ] [time <sub>H</sub> ][time <sub>L</sub> ]	IDLE
Reset factory default settings	'PFR'	(none)	IDLE

INTERPRETATION AND DEFINITION OF CARRIER TIME	
calculation formula	
carrier_time[ms] = time/160 – 0,5ms	
time = (carrier_time[ms] + 0,5ms) * 160	
valid range	
112 - 560 = 0,2ms - 3ms	

INTERPRETATION OF THE FBYTE DATA		
bit	FBYTE <sub>L</sub>	FBYTE <sub>H</sub>
0	trigger mode (0->ext, 1->int, default: ext)	(RFU)
1	trigger edge (0->falling, 1->rising, default: rising)	(RFU)
2	pass signal mode (0->static, 1->pulsed, default: static)	(RFU)
3	failed = HIGH option (0->disabled, 1->enabled, default: disabled)	(RFU)
4	(RFU)	(RFU)
5	(RFU)	(RFU)
6	(RFU)	(RFU)
7	(RFU)	(RFU)



## 6.3 Inline Measurement Commands

The Tagsurance Tester can be instructed to conduct a series of tests (i.e. inline case) for a tag. Tests may include point tests (i.e. response tests in specific frequency-power points), and/or read tests (i.e. reading the tag memories). Inline case data are uploaded through the serial interface, and ran by using specific serial commands. Alternatively, they can also be stored and applied from the device memory (up to 5 cases).

INLINE MEASUREMENT COMMANDS			
Function	Command	Response	Next state
<b>Upload case data</b>			
• header	['L'][Data Length H][Data Length L]	error code <sup>4</sup>	IDLE
• data for point tests	['P'][Tolerance][Npoints][Mode + f <sub>H</sub> ][f <sub>L</sub> ][Power]		
• data for read tasks	['R'][Bank + f <sub>H</sub> ][f <sub>L</sub> ][Power][Word Pointer] [Word Count][Repetitions + Tolerance]		
• data for write tasks	['W'][Bank + f <sub>H</sub> ][f <sub>L</sub> ][Power][Word Pointer] [Repetitions + Tolerance][Increment] [Word Count][Data]		
• data for sweep task	['S'][Start f <sub>H</sub> ][Start f <sub>L</sub> ] [Stop f <sub>H</sub> ][Stop f <sub>L</sub> ] [Step f <sub>H</sub> ][Step f <sub>L</sub> ]		
• data for sensitivity test	['C'][ f <sub>H</sub> ][f <sub>L</sub> ][P <sub>H</sub> ][P <sub>L</sub> ][P <sub>UCL</sub> ][Uncertainty]		
<b>Start (loaded) case</b>	'C'	error code <sup>4</sup>	WAIT FOR TRIG <sup>2</sup>
<b>Trigger</b>	'T'	pass/fail &data <sup>5</sup>	WAIT FOR TRIG
<b>Stop (running) case<sup>1</sup></b>	'X'	(none)	IDLE
<b>Save (active) case</b>	['F'][case number (0-4)]	error code <sup>4</sup>	IDLE
<b>Load saved case</b>	['O'][case number (0-4)]	error code <sup>4</sup>	IDLE
<b>Get active case data</b>	'PD'	case data <sup>3</sup>	IDLE

<sup>1</sup> Any char other than 'T' also stops the running case and causes the device to return to IDLE state

However, for the sake of simplicity, 'X' is recommended

<sup>2</sup> If no valid case is loaded, the device will return error and return to IDLE state

<sup>3</sup> If no valid case is loaded, the device will return {0x00, 0x00} (i.e. number of data bytes is zero)

<sup>4</sup> For description of the error codes, see Section 5.11: "Error handling"

<sup>5</sup> For description of the data, see section 6.3.2: "Running an inline case"



### 6.3.1 Uploading Inline Case Data

Test case data can be uploaded by using a specific command sequence which consists of a header and a data sequence. Header initiates the activity in the IDLE state and defines the amount of data included in the case data. Data sequence, on the other hand, defines the tests to be made when a case is performed. Possible measurements are point-wise (frequency, power) response tests and reading of tag memories.



While designing a case, the user must take care of the case data size. Maximum length of the data sequence is 510 bytes, and the maximum length of the result data vector is 100 bytes.

#### Case Data Header

[‘L’][Data Length H][Data Length L]

Where

[‘L’] is the control char that initiates the activity from the IDLE state  
[Data length] (16bits) indicates the number of bytes included in the data sequence that follows

#### Definition of the Point Tests

[‘P’][Tolerance][Npoints][Mode<sub>1</sub> + f<sub>H1</sub>][f<sub>L1</sub>][Power<sub>1</sub>]...[Mode<sub>N</sub> + f<sub>HN</sub>][f<sub>LN</sub>][Power<sub>N</sub>]

Where:

[‘P’] is the identifier of the point test definitions  
[Tolerance] indicates the amount of point tests that may fail while the tag still passes the test  
[Npoints] is the amount of test points  
[Mode] (2 MSB bits) defines if the tag: must respond (01), must not respond (10), response is indifferent (00)  
[f] (14 bits) defines the carrier frequency presented as [MHz]x10  
[Power] defines the carrier power (presented as: [dBm]x4+128)

#### Definition of the Read Tests

[‘R’][Bank + f<sub>H</sub>][f<sub>L</sub>][Power][Word Pointer][Word Count][Repetitions + Tolerance]

Where:

[‘R’] is the identifier of the read test definitions  
[Bank] (memory bank, 2 MSB bits) is the identifier of the tag’s memory bank (00, 01, 10, or 11)  
[f] (14 bits) defines the carrier frequency (presented as [MHz]x10)  
[Power] defines the carrier power (presented as [dBm]x4+128)  
[Word pointer] identifies the start address for the read command (0-128, EBV format)  
[Word count] defines the number of words to be read from the tag  
[Repetitions] number of repetitions (4 MSB bits) is the number of times the read-test is performed  
[Tolerance] is the number of times the read test may fail while it still is considered to be successful

### Definition of the Write Tests

[‘W’][Bank +  $f_H$ ][ $f_L$ ][Power][Word Pointer][Repetitions + Tolerance][Increment][Word Count][Data]

Where:

[‘W’] is the identifier of the write test definitions  
[Bank] (memory bank, 2 MSB bits) is the identifier of the tag’s memory bank (00, 01, 10, or 11)  
[ $f$ ] (14 bits) defines the carrier frequency (presented as [MHz]x10)  
[Power] defines the carrier power (presented as [dBm]x4+128)  
[Word pointer] identifies the start address for the read command (0-128, EBV format)  
[Repetitions] number of repetitions (4 MSB bits) is the number of times the read-test is performed  
[Tolerance] is the number of times the read test may fail while it still is considered to be successful  
[Increment] defines the increment for the data per cycle (0 for static data, >0 for incremental data)  
[Word Count] defines the number of words to be written to the memory bank (max. word count is 8)  
[Data] is the data that is written to the tag (number of bytes = 2 x word count)

### Definition of the Sweep Test

[‘S’][Start  $f_H$ ][Start  $f_L$ ][Stop  $f_H$ ][Stop  $f_L$ ][Step  $f_H$ ][Step  $f_L$ ]

Where:

[Start  $f$ ] defines the start frequency  
[Stop  $f$ ] defines the stop frequency  
[Step  $f$ ] defines the frequency step size

### Definition of the Sensitivity Test

[‘C’][ $f_H$ ][ $f_L$ ][ $P_H$ ][ $P_L$ ][ $P_{LCL}$ ][ $P_{UCL}$ ][Uncertainty]

Where:

[ $f$ ] defines the frequency  
[ $P_H$ ] defines the highest tested power  
[ $P_L$ ] defines the lowest tested power  
[ $P_{LCL}$ ] defines the lower control limit  
[ $P_{UCL}$ ] defines the upper control limit  
[Uncertainty] defines the uncertainty criterion

## 6.3.2 Running an Inline Case

### Starting the Measurement

The test case can be started by sending a 'Start case' command from the serial port ('C'). Having received the command, the device checks that a valid case has been loaded, and goes to the WAITING FOR TRIGGER state. If a valid case is not available, the device returns to the IDLE state. The action taken can be interpreted from the error flag sent as a response to the command.



In some case, it may be useful to download the active case data back from the device. This can be done by using command: 'PD', which returns the case length and data, if one has been loaded.

### Triggering

In the WAIT FOR TRIGGER state a measurement can be initiated by providing a trigger from the serial port by sending: 'T', or by applying trigger signal to the external trigger pin. In response, the device performs the measurement, sends the results to the serial port, and returns to the WAIT FOR TRIGGER state.

### Interpretation of Results

The results are provided to the serial port in the following format:

Byte 1: Pass/fail byte, 0x00 (failed), 0x01 (passed)

Byte 2...N: Test details

- Point Test results are provided as single bits starting from the first empty byte in the result vector. LOW represents a failed test, HIGH represents a passed test.
- Read Test results are provided with an error byte (see table below) followed by the data read. If the task was failed, the data is replaced with 0x00's.
- Write Test results are provided with an error byte (see table below).
- Sweep Test results are provided as bytes representing the power thresholds at tested frequencies. Powers are indicated as:  $[(\text{dBm}) \times 4] + 128$ .
- Sensitivity Test results are provided with an error byte indicating that LCL/UCL criterion was passed (0x00), failed (0x01) or in a case where the LCL/UCL criterion is not defined inside the measurement range and the result is out of measurement range (0x02). Next byte is the sensitivity threshold measured. Powers are indicated as:  $[(\text{dBm}) \times 4] + 128$ .

INTERPRETATION OF READ/WRITE TASK ERROR BYTE			
0x00	0 <sub>10</sub>	No errors	Tolerance condition was met
0x01	1 <sub>10</sub>	Connection failed	Tag did not respond to query
0x02	2 <sub>10</sub>	Command failed	Tag did not respond to the command
0x03	3 <sub>10</sub>	Inventory failed	Tag responds to query, but inventory failed
0x04	4 <sub>10</sub>	Tag error message 1	Other error (not covered by the other tag error codes)
0x34	52 <sub>10</sub>	Tag error message 2	Memory overrun (too high memory address)
0x44	68 <sub>10</sub>	Tag error message 3	Memory locked (memory cannot be read/written)
0xB4	180 <sub>10</sub>	Tag error message 4	Insufficient power (carrier level too low)
0xF4	244 <sub>10</sub>	Tag error message 5	Non-specified (error-specific codes not supported by the tag)

INTERPRETATION OF SENSITIVITY TEST TASK ERROR BYTE			
0x00	0 <sub>10</sub>	No errors	Tolerance condition was met
0x01	1 <sub>10</sub>	Out of tolerances	Tolerance condition was not met, the threshold power exceeded the upper control limit or was below the lower control limit
0x02	2 <sub>10</sub>	Out of range	UCL and LCL are not defined inside the measurement range and the response power is out of measurement range.



The WAIT FOR TRIGGER mode is indicated by the ready led in the device front panel being lit and the ready/busy pin in the external connector low. During the case execution, the BUSY pin is high and the BUSY led in the device front-panel lit. Furthermore, the outcome of the test (PASS/FAIL) is indicated by the PASS and FAIL LEDs in the device front panel, and the PASS/FAIL pin in the external connector.

### Stopping the Measurement

To stop the case and to return to the IDLE state, the user must send 'X' to the serial port. This will cause the device to stop performing the case and to go to the IDLE state. Also any other character will have the same effect, but for the sake of simplicity, 'X' is recommended.

### 6.3.3 Test Case Storage and Handling

After a case has been successfully uploaded, it can be stored in the device memory. The device will respond to the command by returning an error flag, which indicates if the task could be successfully performed. It then returns to the IDLE state. Load function works the same way. Furthermore, once loaded, the active case data can be downloaded through the serial port by using command: 'PD'.

The following commands can be used to perform these actions:

['F'][Memory Location]

save active case to a memory location (0x0 to 0x4)

['O'][Memory Location]

load case from a memory location (0x0 to 0x4)

['PD']

Transmit active case data to the serial port



To avoid the need to build code for uploading the inline test cases, they can be loaded and restored by using the application software. This allows simplification of the control code.

## 6.4 Other Measurement Commands

In addition to inline cases, the device can perform separate tests that can be executed in the IDLE state by using specific commands. These tests include: response threshold test and system self-test.

OTHER MEASUREMENT FUNCTIONS			
Function	Command	Response	Next state
Threshold sweep	['S'][Start $f_H$ ][Start $f_L$ ][Stop $f_H$ ][Stop $f_L$ ][Step $f_H$ ][Step $f_L$ ]	[Err][N Bytes][Data] <sup>1</sup>	IDLE
Read	['R'][Bank + $f_H$ ][ $f_L$ ][Power][Word Pointer][Word Count][Repetitions + Tolerance]	[Err][ErrByte][Data] <sup>2</sup>	IDLE
BlockWrite	['W'][Bank + $f_H$ ][ $f_L$ ][Power][Word Pointer][Word Count][Data]	[Err][ErrByte] <sup>3</sup>	IDLE

<sup>1</sup>Byte count and data will follow if the command is valid (user must check the error flag, 0x00=no error)

<sup>2</sup>errByte (0x00 = no error) followed by the data read (repeated string of 0x00's if read was failed)

<sup>3</sup>errByte (0x00 = no error)

### 6.4.1 Threshold Sweep

The Threshold Sweep test performs a frequency sweep and measures the power threshold needed to wake up the tag through a certain frequency band. The sweep start frequency, stop frequency and the size of the frequency step are defined by the user. The device responds to the command by sending an error byte. The byte count and measurement data will follow if the command is valid.

**Command:** ['S'][Start  $f_H$ ][Start  $f_L$ ][Stop  $f_H$ ][Stop  $f_L$ ][Step  $f_H$ ][Step  $f_L$ ]

**Response:** [err][N bytes][Threshold Power<sub>1</sub>][Threshold Power<sub>2</sub>]... [Threshold Power<sub>N</sub>]

Where

[Err]	identifies if the command was valid (0x00=no error)
[ErrByte]	indicates the error while performing the task
[N bytes]	byte count of the following data
[Threshold Power <sub>N</sub> ]	the threshold power at N <sup>th</sup> measurement frequency

Frequencies are given in format: [MHz]x10

Powers are given in format: [dBm]x4+128

### 6.4.2 Read

The Read Test performs a single read operation for tag.

**Command:** ['R'][Bank +  $f_H$ ][ $f_L$ ][Power][Word Pointer][Word Count][Repetitions + Tolerance]

**Response:** [Err][ErrByte][Data Byte<sub>1</sub>][Data Byte<sub>2</sub>]...[Data Byte<sub>N</sub>]

Where

[Err]	identifies if the command was valid (0x00=no error)
[ErrByte]	indicates the error while performing the task
[Data Byte]	the data read from the tag is divided to N data bytes

### 6.4.3 Block Write

The Block Write performs a single write operation for tag.

**Command:** ['W'][Bank + f<sub>H</sub>][f<sub>L</sub>][Power][Word Pointer][Word Count][Data]

**Response:** [err][errByte]

Where

[Err]	identifies if the command was valid (0x00=no error)
[ErrByte]	indicates the error while performing the task

## 6.5 Tag Encoding, Locking, and Killing Commands

The Tagsurance provides two methods for encoding, locking, or killing a tag. Provided are basic commands for encoding, locking, and killing. The encode command can be used to program and lock EPC memory, user memory, access password, and kill password. The lock command can be used to set tag lock bits, and the kill command is used to perform tag killing procedure with optional kill password encoding.

It is also possible to implement a custom encoding session by using a separate encoding command set. This allows utilization of more complicated encoding procedures infeasible with basic commands. Custom command set includes: inventory with optional access, block write, read, lock, kill, and close session. The session is started with inventory and closed with end session command.

TAG ENCODING, LOCKING, AND KILLING FUNCTIONS			
Function	Command	Response	Next state
BASIC COMMANDS			
Encode	['H'][Data Length H][Data Length L] [f <sub>H</sub> ][f <sub>L</sub> ][Power][Acc-pwd <sub>1,MSB</sub> ][Acc-pwd <sub>2</sub> ][Acc-pwd <sub>3</sub> ][Acc-pwd <sub>4</sub> ] [EPC Word count][EPC Word pointer][EPC Data] [User Word count][User Word pointer][User Data] [Enab kill-pwd prog][Kill-pwd <sub>1,MSB</sub> ][Kill-pwd <sub>2</sub> ][Kill-pwd <sub>3</sub> ][Kill-pwd <sub>4</sub> ] [Enab acc-pwd prog][Acc-pwd <sub>1,MSB</sub> ][Acc-pwd <sub>2</sub> ][Acc-pwd <sub>3</sub> ][Acc-pwd <sub>4</sub> ] [Enab lock prog][Payload <sub>1,MSB</sub> ][Payload <sub>2</sub> ][Payload <sub>3</sub> ] <sup>5</sup>	[Err] <sup>1</sup> [Task][ErrByte] <sup>2-4</sup>	IDLE
Lock	['D'][f <sub>H</sub> ][f <sub>L</sub> ][Power][Repetitions + Tolerance] [Payload <sub>1,MSB</sub> ][Payload <sub>2</sub> ][Payload <sub>3</sub> ]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	IDLE
Kill	['K'][f <sub>H</sub> ][f <sub>L</sub> ][Power] [Acc-pwd <sub>1,MSB</sub> ][Acc-pwd <sub>2</sub> ][Acc-pwd <sub>3</sub> ][Acc-pwd <sub>4</sub> ] [Kill-pwd <sub>1,MSB</sub> ][Kill-pwd <sub>2</sub> ][Kill-pwd <sub>3</sub> ][Kill-pwd <sub>4</sub> ]	[Err] <sup>1</sup> [Task][ErrByte] <sup>2-4</sup>	IDLE
CUSTOM COMMAND SET			
Inventory	['I'][f <sub>H</sub> ][f <sub>L</sub> ][Power][Acc-pwd <sub>1,MSB</sub> ][Acc-pwd <sub>2</sub> ][Acc-pwd <sub>3</sub> ][Acc-pwd <sub>4</sub> ]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	IDLE/ENCODING
BlockWrite	['E']['W'][Bank][Word pointer][Word count][Data]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	ENCODING
Read	['E']['R'][Bank][Word pointer][Word count]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	ENCODING
Lock	['E']['L'][Payload <sub>1,MSB</sub> ][Payload <sub>2</sub> ][Payload <sub>3</sub> ]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	ENCODING
Kill	['E']['K'][Kill-pwd <sub>1,MSB</sub> ][Kill-pwd <sub>2</sub> ][Kill-pwd <sub>3</sub> ][Kill-pwd <sub>4</sub> ]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	ENCODING
End session	['X']	-	IDLE

<sup>1</sup>Err represents system error code (0x00=no error). For description of the codes, see Section 5.11: "Error handling"

<sup>2</sup>ErrByte (and task) will follow Err if the command is valid (Err=0x00)

<sup>3</sup>Task indicates the last task performed before end session (See task indices table below)

<sup>4</sup>ErrByte represents error code from the last task (0x00 = no error)

<sup>5</sup>Payload byte syntax is presented in the table in chapter 6.5.2.

INTERPRETATION OF TASK IDENTIFIER BYTE		
0x00	0 <sub>10</sub>	Inventory (and access) sequence
0x01	1 <sub>10</sub>	EPC memory programming
0x02	2 <sub>10</sub>	User memory programming
0x03	3 <sub>10</sub>	Kill password programming
0x04	4 <sub>10</sub>	Access password programming
0x05	5 <sub>10</sub>	Lock bit programming
0x06	6 <sub>10</sub>	Kill procedure

### 6.5.1 Encode

The encode command performs programming and locking of EPC memory, user memory, access password, and kill password according to user provided parameters.

#### Command:

[‘H’] Data Length H][Data Length L]...

...[fH][fL][Power][Acc-pwd<sub>1,MSB</sub>][Init acc-pwd<sub>2</sub>][Init acc-pwd<sub>3</sub>][Init acc-pwd<sub>4</sub>]...

...[EPC Word count][EPC Word pointer][EPC Data]...

...[User Word count][User Word pointer][User Data]...

...[Enable kill password programming][Kill-pwd<sub>1,MSB</sub>][Kill-pwd<sub>2</sub>][Kill-pwd<sub>3</sub>][Kill-pwd<sub>4</sub>]...

...[Enable access password programming][Acc-pwd<sub>1,MSB</sub>][Acc-pwd<sub>2</sub>][Acc-pwd<sub>3</sub>][Acc-pwd<sub>4</sub>]...

...[Enable lockbits programming][Payload<sub>1,MSB</sub>][Payload<sub>2</sub>][Payload<sub>3</sub>]

Where

[‘H’]	is the identifier of the encode parameters
[Data length]	(16bits) is the number of bytes included in the data sequence that follows
[f]	(14 LSB bits) defines the carrier frequency (presented as [MHz]x10)
[Power]	defines the carrier power (presented as [dBm]x4+128)
[Init acc-pwd <sub>1-4</sub> ]	are access password bytes used to perform access after inventory (MSB first)
[Word Count]	defines the number of words to be written to the memory bank
[Word Pointer]	identifies the start address for the write command (0-128, EBV format)
[Data]	is the data that is written to the tag (number of bytes = 2 x word count)
[Enable ... programming]	enables/disables the programming tasks (0x00=disabled, 0x01=enabled)
[Acc-pwd <sub>1-4</sub> ]	are access password bytes (MSB first) to be programmed
[Kill-pwd <sub>1-4</sub> ]	are kill password bytes (MSB first) to be programmed
[Payload <sub>1-3</sub> ]	are lock command payload bits (20 bits, MSB first) + 4 dummy bits (LSB)

Notes:

- inventory is made with access if initial access password is non-zero
- the amount of bytes after data length has to match with the data length
- the amount of bytes sent has to match with the word count (i.e. data and word pointer not sent, if word count=0)
- password bytes are provided only if password programming is enabled (Enable ... programming = 0x01)
- lock bits are provided only if lockbits programming is enabled (enable lockbits programming = 0x01)

#### Response:

[Err][Task][errByte]

Where

[Err]	identifies if the command was valid (0x00=no error)
[Task]	indicates the last task performed
[ErrByte]	indicates the error while performing the task (0x00=no error)



## 6.5.2 Lock

Lock performs locking procedure for a tag at predefined frequency and power.

### Command:

[‘D’][fH][fL][Power][Repetitions + Tolerance][ Payload<sub>1,MSB</sub>][Payload<sub>2</sub>][Payload<sub>3</sub>]

Where

[‘D’] is the identifier of the lock parameters  
[f] (14 LSB bits) defines the carrier frequency (presented as [MHz]x10)  
[Power] defines the carrier power (presented as [dBm]x4+128)  
[Repetitions] (4 MSB bits) is the maximum number of repetitions the task is performed  
[Tolerance] (4 LSB bits) is the maximum number of repetitions allowed to fail  
[Payload<sub>1-3</sub>] are lock command payload bits (20 bits, MSB first) + 4 dummy bits (LSB) (see table below)

### Response:

[Err][errByte]

Where

[Err] identifies if the command was valid (0x00=no error)  
[ErrByte] indicates the error while performing the task (0x00=no error)

	KILL PASSWORD		ACCESS PASSWORD		EPC MEMORY		TID MEMORY		USER MEMORY	
	23 (MSB)	22	21	20	19	18	17	16	15	14
MASK BITS	0:skip 1:write	0:skip 1:write	0:skip 1:write	0:skip 1:write	0:skip 1:write	0:skip 1:write	0:skip 1:write	0:skip 1:write	0:skip 1:write	0:skip 1:write
	13	12	11	10	9	8	7	6	5	4
ACTION BITS	pwd write/ read	perma lock	pwd write/ read	perma lock	pwd write	perma lock	pwd write	perma lock	pwd write	perma lock
	3	2	1	0						
DUMMY BITS	-	-	-	-						

### 6.5.3 Kill

Kill performs killing procedure for a tag at predefined frequency and power.

#### Command:

[‘K’][fH][fL][Power]...  
...[Acc-pwd<sub>1,MSB</sub>][Acc-pwd<sub>2</sub>][Acc-pwd<sub>3</sub>][Acc-pwd<sub>4</sub>]...  
...[Kill-pwd<sub>1,MSB</sub>][Kill-pwd<sub>2</sub>][Kill-pwd<sub>3</sub>][Kill-pwd<sub>4</sub>]

Where

[‘K’]	is the identifier of the kill parameters
[f]	(14 LSB bits) defines the carrier frequency (presented as [MHz]x10)
[Power]	defines the carrier power (presented as [dBm]x4+128)
[Acc-pwd <sub>1-4</sub> ]	are access password bytes (MSB first)
[Kill-pwd <sub>1-4</sub> ]	are kill password bytes (MSB first)

#### Response:

[Err][Task][errByte]

Where

[Err]	identifies if the command was valid (0x00=no error)
[Task]	indicates the last task performed
[ErrByte]	indicates the error while performing the task (0x00=no error)

Notes:

- A tag will not accept kill command if kill password is zero. If the kill password provided with the command is zero, the Tagsurance will try to program kill password before attempting to kill the tag.
- If kill password is locked, reprogramming can only be performed if the tag is in secured state. If the access password provided with the command is non-zero, the Tagsurance will perform inventory with access.

#### 6.5.4 Custom Command Set: Inventory with Optional Access

Inventory with optional access command turns carrier on and performs inventory with the tag. Then, it continues with the optional access procedure, if a non-zero access password is provided with the command.

##### Command:

[‘I’] [fH][fL][Power][Acc-pwd<sub>1,MSB</sub>][Acc-pwd<sub>2</sub>][Acc-pwd<sub>3</sub>][Acc-pwd<sub>4</sub>]

Where

[‘I’] is the identifier of the inventory parameters  
[f] (14 bits) defines the carrier frequency (presented as [MHz]x10)  
[Power] defines the carrier power (presented as [dBm]x4+128)  
[Acc-pwd<sub>1-4</sub>] are access password bytes used to perform access after inventory (MSB first)

##### Response:

[Err][errByte]

Where

[Err] identifies if the command was valid (0x00=no error).  
[ErrByte] indicates the error while performing the task (0x00=no error)

#### 6.5.5 Custom Command Set: Block Write

Block Write Performs a single write operation for tag. It can be performed after a successful inventory.

##### Command:

[‘E’][‘W’][Bank][Word Pointer][Word Count][Data]

Where

[‘E’] is the identifier of the encode command set  
[‘W’] is the identifier of the write parameters  
[Bank] (memory bank, 2 MSB bits) is the identifier of the tags memory bank (00, 01, 10, or 11)  
[Word Pointer] identifies the start address for the write command (0-128, EBV format)  
[Word Count] defines the number of words to be written to the memory bank  
[Data] is the data that is written to the tag (number of bytes = 2 x word count)

##### Response:

[Err][errByte]

Where

[Err] identifies if the command was valid (0x00=no error)  
[ErrByte] indicates the error while performing the task (0x00=no error)

### 6.5.6 Custom Command Set: Read

Read performs a single read operation for a tag. It can be performed after a successful inventory.

#### Command:

[‘E’][‘R’][Bank][Word Pointer][Word Count]

Where

[‘E’] is the identifier of the encode command set  
[‘R’] is the identifier of the read parameters  
[Bank] (memory bank, 2 MSB bits) is the identifier of the tags memory bank (00, 01, 10, or 11)  
[Word Pointer] identifies the start address for the read command (0-128, EBV format)  
[Word Count] defines the number of words to be written to the memory bank

#### Response:

[Err][errByte]

Where

[Err] identifies if the command was valid (0x00=no error)  
[errByte] indicates the error while performing the task (0x00=no error)

### 6.5.7 Custom Command Set: Lock

Lock performs a locking procedure for a tag. It can be performed after a successful inventory.

#### Command:

[‘E’][‘L’][Payload<sub>1,MSB</sub>][Payload<sub>2</sub>][Payload<sub>3</sub>]

Where

[‘E’] is the identifier of the encode command set  
[‘L’] is the identifier of the lock parameters  
[Payload<sub>1-3</sub>] are lock payload bits (20 bits, MSB first) + 4 dummy bits (LSB) (see table below)

#### Response:

[Err][errByte]

Where

[Err] identifies if the command was valid (0x00=no error)  
[errByte] indicates the error while performing the task (0x00=no error)

### 6.5.8 Custom Command Set: Kill

Kill performs a killing procedure for a tag. It can be performed after a successful inventory.

#### Command:

[‘E’][‘K’] [Kill-pwd<sub>1,MSB</sub>][Kill-pwd<sub>2</sub>][Kill-pwd<sub>3</sub>][ Kill-pwd<sub>4</sub>]

Where

[‘E’] is the identifier of the encode command set

[‘K’] is the identifier of the kill parameters

[Kill-pwd<sub>1-4</sub>] are kill password bytes (MSB first)

#### Response:

[Err][errByte]

Where

[Err] identifies if the command was valid (0x00=no error)

[ErrByte] indicates the error while performing the task (0x00=no error)

### 6.5.9 Custom Command Set: End Session

End session turns carrier down to power-off the tag

#### Command:

[‘X’]

#### Response:

None

## 6.6 Combining Encoding with Performance Testing

In order to allow encoding while running a performance test case, the user must activate “encode after performance testing” option. This is done by using a setup command while the device is in IDLE state. With this option activated, the Tagsurance will automatically go to IDLE state after completing performance test case. In this state, encoding can be performed by using the encode commands. After completing the encoding procedure, performance testing is reinitiated from serial port.

In normal testing mode, the Tagsurance front-panel led lights and back panel signals are updated after the performance testing is done. With “encode after performance testing” option activated, the indicators are updated with the reinitialization of the test case. Pass/fail signals are affected by the encode functions performed and the test result is failed if any of the encoding activities (encode, kill, encode command set functions) fail.

Once activated (‘PME’), the option will stay on until it is deactivated from the serial port (‘PMD’), or the device power is recycled.

COMBINING ENCODING WITH PERFORMANCE TESTING			
Function	Command	Response	Next state
Activate encode after performance test option	‘PME’	(none)	IDLE
Deactivate encode after performance test option	‘PMD’	(none)	IDLE
Reinitialize inline case (after encoding)	‘C’	Err <sup>1</sup>	WAIT FOR TRIG

<sup>1</sup>Err represents system error code (0x00=no error). For description of the codes, see Section 5.11: “Error handling”

## 6.7 Error Handling

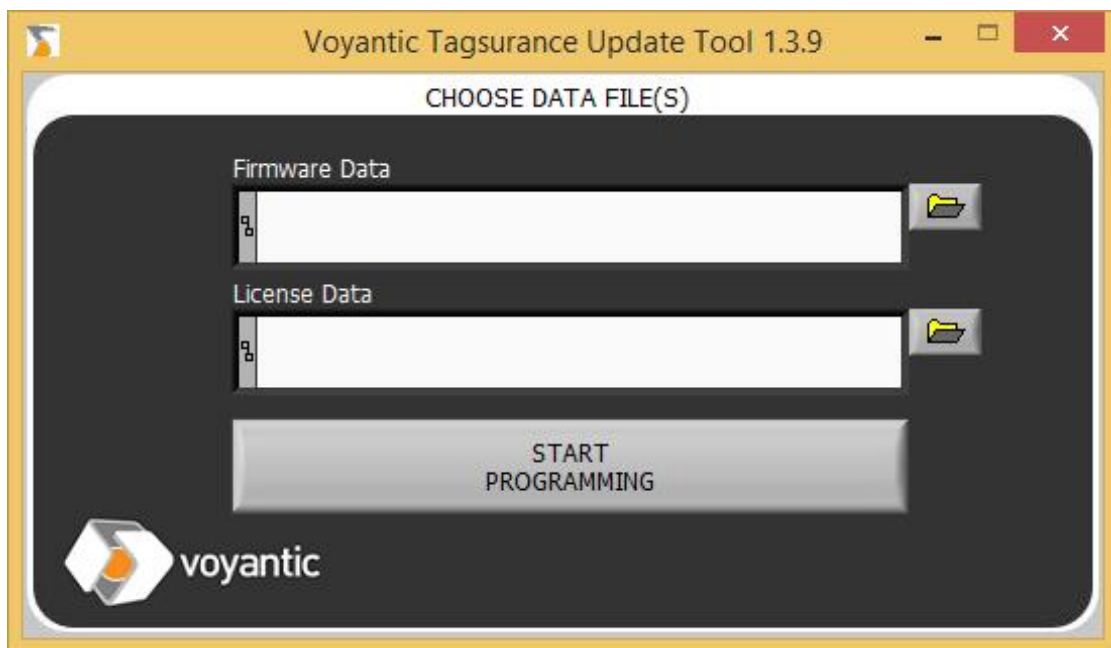
If errors occur during the data transmission or if invalid commands are given, it will be identified by the error code returned. No error is indicated 0x00. Any other value indicates that an error occurred.

INTERPRETATION OF SYSTEM ERROR CODE(S) (0 -> no error, 1 --> error)		
Bit 0	invalid input data sequence	valid range: 1 to 510 bytes
Bit 1	timeout during data reception	maximum delay between data bytes: 1000ms
Bit 2	invalid memory bank	valid range: 0 to 4
Bit 3	invalid word count	valid range: >0 (read test), 1-8 (write test)
Bit 4	invalid command characters	valid values: 'P', 'R', 'W', and 'S'
Bit 5	output data size required too large	valid range: 0 to 100 bytes
Bit 6	invalid frequency	valid range: 860 to 960MHz (extended: 800 to 1100MHz)
Bit 7	Invalid power	valid range: -10 to +25dBm
0xFF	license error	requested task not covered by current license options

## 7 Updating the Device Firmware and License

The USB memory stick delivered with the device contains also an Update Tool, which is used to update either the device firmware or the license. The Update Tool is located in the “Tagsurance Update Tool” folder. To update the device firmware or license, follow the instructions below:

1. Power on the Tagsurance unit and connect to the PC with serial cable
2. Start the update software on the USB stick by executing: Tagsurance Update Tool\Update\_Tool.exe  
The following interface will appear:



**Fig 36 Tagsurance Update Tool Interface**

3. If you wish to update the firmware, insert the file path in the “Firmware Data” field or browse and define the file path by clicking the folder icon on the right of the “Firmware Data” field

If you wish to update the license, insert the file path in the “License Data” field or browse and define the file path by clicking the folder icon on the right of the “License Data” field

4. Press the “Start Programming” button in the interface
5. After receiving an instruction window pop-up, press down the programming button in the back panel and recycle device power while keeping the button pressed.
6. Press ‘ok’ button in the instruction window
7. The program will run the update procedure and inform when the update is ready

# Appendix A:

## TCP Remote Access Interface for Tagsurance GUI

Compatible with Tagsurance GUI versions 1.6.0 ->

### A.1 Overview

TCP/IP interface allows accessing and controlling the Tagsurance test system remotely over a network connection. It can, for example, be used to provide control of measurement events to the main computer unit while the computer with the Tagsurance GUI is kept close to the production line (Fig A1). This allows efficient use of the graphical indicators in the Tagsurance GUI while the data is collected and stored elsewhere.

In this mode, the Tagsurance GUI will act as a TCP server, and the client can control the measurement activities by using a specific command set. Overtaking the control requires a simple handshaking procedure.

The handshake procedure is described in chapter A.2, and the commands in chapter A.3.

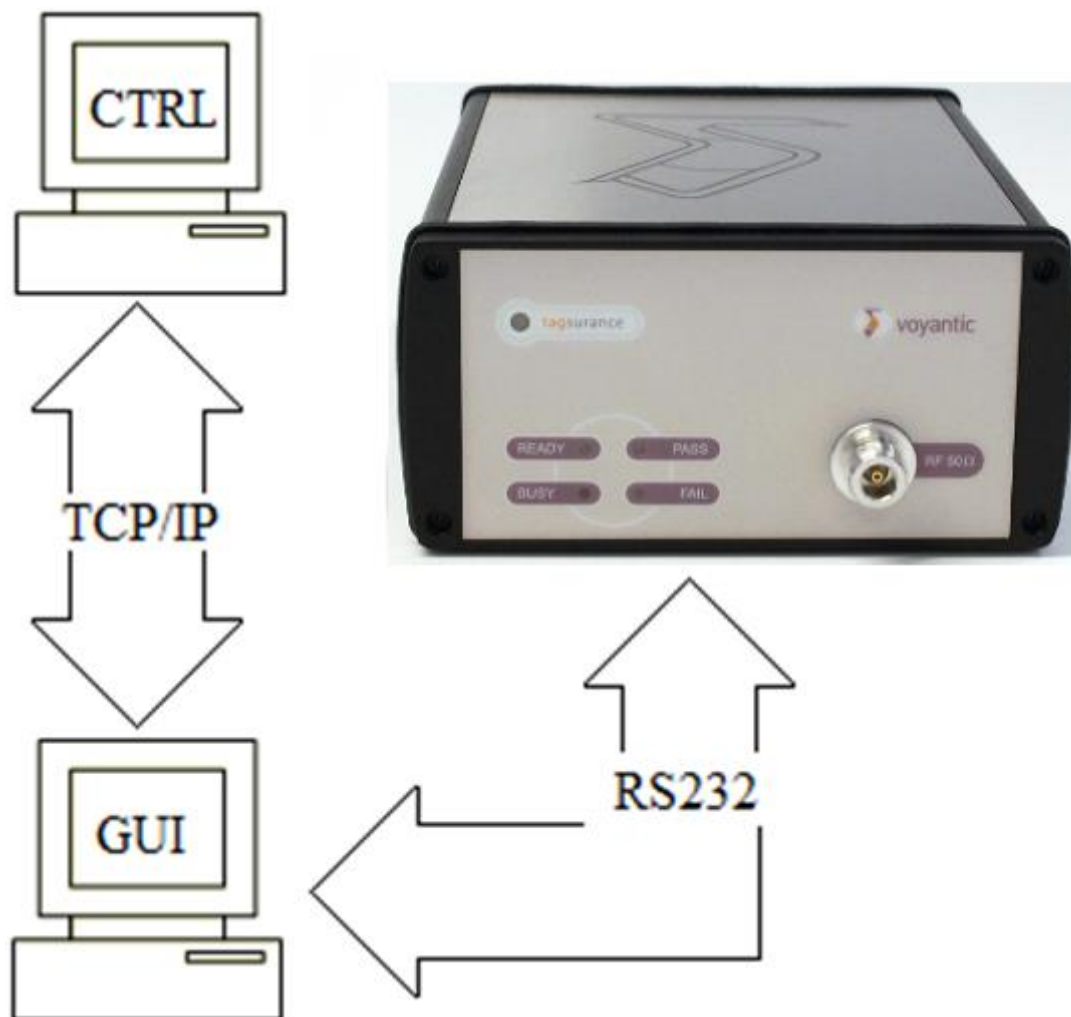


Fig A1. Example of a TCP/IP controlled setup



## A.2 TCP/IP Connection Establishment

To enable accessing the Tagsurance TCP/IP interface, the user must follow a simple procedure:

1. Run tagsurance\_gui.exe
2. Define port settings from System settings
  - Define TCP port number (default: 6342)
  - Enable remote access
  - Save and exit
3. Tagsurance GUI will now start polling for connection attempts at the port defined.  
This is indicated by a frame blinking around 'Device ready' indication led
4. Establish connection
  - Client opens TCP connection (computer IP address, and port number needed)
  - Tagsurance GUI recognizes connection attempt and waits for 'TCP Test' (timeout: 10s)
  - Client transmits 'TCP Test' (=F0 00 00) within 10s from the connection attempt

For a valid command, the Tagsurance GUI responds with 'TCP ready' (=F1 00 00)

→ Tagsurance GUI remote access interface is activated and ready to use

For an invalid command, the Tagsurance GUI responds with 'ERR' (= FF 01 00 xx)

→ Connection attempt is terminated, and remote access interface is not activated

Else (i.e. if timeout occurs)

→ Connection attempt is terminated, and remote access interface is not activated

5. After the handshake procedure has been successfully completed, the remote access interface is ready to be used. This is indicated in the Tagsurance GUI by a blue ring around the 'Device ready' indication led.



To avoid the need for manual initialization, it is also possible to generate a windows batch file to run the Tagsurance GUI and open it directly in remote access mode. Detailed instructions on how to use program call arguments, is provided in Appendix C.

## A.3 Commands Description

Control over the network is made by sending specific commands which initiates certain activities.

The commands consist of three parts: command byte, two length bytes, and the possible command parameters (see table below). The length bytes indicate the length of the last part of the command (i.e. the amount of parameter bytes). Data is in hexadecimal format and numbers are sent low byte first.

Command	Length Low Byte (Nlow)	Length High Byte Nhigh	Parameters (N bytes of data)
0x00	0x00	0x00	...

List of available commands is given below. Detailed description is provided in the next chapters.

CODE	COMMAND	DESCRIPTION	INIT STATE	NEXT
0x01	Connect	Connect to Tagsurance Tester Unit	idle/inline	no change
0x02	Connection State	State of Connection to Tagsurance Tester Unit	-	-
0x03	GCL	Get Case List	idle/inline	no change
0x04	CL	Case List (semicolon “;” separated)	-	-
0x05	LSC	Load and Start Specific Case	idle/inline	inline
0x06	SCL	Specific Case Loaded and Started	-	-
0x07	TRIG	Serial trigger to Tagsurance Tester Unit	inline	inline
0x08	READY	Serial triggering successful (and device is ready)	-	-
0x09	STOP	Stop running case	inline	idle
0x0A	STOPPED	Case stopped	-	-
0x10	GTR	Get Test Result	idle/inline *	no change
0x11	TR	Test Results	-	-
0x12	GBS	Get Buffer Status	idle/inline *	no change
0x13	BS	Buffer State	-	-
0x30	READ	Read data from tag memory	idle/inline	no change
0x31	WRITE	Write data to tag memory	idle/inline	no change
0x32	SWEEP	Measure tag’s response threshold curve	idle/inline	no change
0x33	ENCODE	Program and lock tag memories	idle/inline	no change
0x34	KILL	Kill tag	idle/inline	no change
0xF0	TCP Test	TCP Connection Test	handshake**	idle**
0xF1	TCP Ready	State of TCP connection	-	-
0xFF	ERR	Error from Tagsurance	-	-

\*Buffer state is initialized and the contents flushed when a new case is started.

\*\* This command is used during handshake procedure. Command has no effect in other states (idle, inline).

### A.3.1 Connect

INIT STATE: idle, inline

NEXT STATE: no change

FUNCTION: test the state of RS232 connection between the Tagsurance GUI and the Tagsurance tester unit

TAGSURANCE RESPONSE: Connection State, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x01	Connect
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

Connect	Length Low Byte	Length High Byte
0x01	0x00	0x00

Cmd: 01 00 00

### A.3.2 Connection State

The Tagsurance response to a successful connection test initiated by “Connect” from client.

The Tagsurance will test the connection with the Tester unit and respond with serial number if connection is ok. Otherwise, it will respond with “ERR”.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x02	Connect
1	Length Low	0x03	Length Low
2	Length High	0x00	Length High
3	Connection State	0x00    0x01	0x00 for Not Connected 0x01 for Connected
4	Serial Number Low	0xFF	Tagsurance Tester serial number low byte
5	Serial Number High	0xFF	Tagsurance Tester serial number high byte

Example Transmit:

Con State	Length Low Byte	Length High Byte	Con State	Serial No. Low Byte	Serial No. High Byte
0x02	0x03	0x00	0x01	0x04	0x00

Cmd: 02 03 00 01 04 00

### A.3.3 GCL (Get Case List)

INIT STATE: idle, inline

NEXT STATE: no change

FUNCTION: ask for a list of available test cases in the data folder

TAGSURANCE RESPONSE: CL, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x03	Get Case List (GCL)
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

GCL	Length Low Byte	Length High Byte
0x03	0x00	0x00

Cmd: 03 00 00

### A.3.4 CL (Case List)

The Tagsurance response to “GCL” from client.

The Tagsurance GUI will search for valid test files from the data folder (“...\Tagsurance\Data\Test case files\”), and return a list of them separated by ‘;’. Included in the list are only the cases that can be run in the current system. To be included, the file names should include keyword: “Test”. Validation is licencing sensitive.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x04	Case List (CL)
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High
3	Count of Case	0x00 ... 0xFF	Number of available cases
4	String First Case Name	...	First case name in hex
...	Semicolon for Separation	0x3B (;)	Semicolon to separate different names for case
...	Case Name	...	... in hex
...	Semicolon for Separation	0x3B (;)	Semicolon to separate different names for case
...	String Last Case Name	...	Name for last case in hex

Example Transmit:

CL	Length Low Byte	Length High Byte	Count of Case	Case 1	Separator	Case2	Separator	Case 3
0x04	0x12	0x00	0x03	“Test1”	0x3B	“Test2”	0x3B	“Test3”

Cmd: 04 12 00 03 54 65 73 74 31 3B 54 65 73 74 32 3B 54 65 73 74 33

### A.3.5 LSC (Load Specific Case)

INIT STATE: idle, inline

NEXT STATE: inline

FUNCTION: initialize and start a new inline case specified in the indicated case file

TAGSURANCE RESPONSE: SCL, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x05	Load and Start Specific Case (LSC)
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High
3	Case Name	...	Name of the Case

Example Transmit:

LSC	Length Low Byte	Length High Byte	Case Name
0x05	0x05	0x00	"Test1"

Cmd: 05 05 00 54 65 73 74 31

### A.3.6 SCL (Specific Case Loaded and Started)

The Tagsurance response to a successful test case initialization initiated by "LSC" from client.

The Tagsurance will first search for the file specified from the data folder, validate the contents, and then start it. The response is provided after the case has been loaded and started successfully.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x06	Specific Case Loaded and Started (SCL)
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High
3	Case Name	...	Name of the Case

Example Transmit:

SCL	Length Low Byte	Length High Byte	Case Name
0x06	0x05	0x00	"Test1"

Cmd: 06 05 00 54 65 73 74 31

### A.3.7 TRIG (Serial trigger to Tagsurance Tester Unit)

INIT STATE: inline

NEXT STATE: inline

FUNCTION: trigger a new test measurement specified in the initiated test case

TAGSURANCE RESPONSE: READY, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x07	Test TCP connection
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

TCP Test	Length Low Byte	Length High Byte
0x07	0x00	0x00

Cmd : 07 00 00

### A.3.8 READY (Serial Triggering Successful)

The Tagsurance response to successful 'TRIG' from client.

The Tagsurance will trigger a new measurement from serial port and wait for the triggered measurement to be performed. Response is provided after the test has been completed.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x08	Connect
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

TCP ready	Length Low Byte	Length High Byte
0x08	0x00	0x00

Cmd : 08 00 00

### A.3.9 STOP (Stop Running Test Case)

INIT STATE: inline

NEXT STATE: idle

FUNCTION: stop the initiated test case

TAGSURANCE RESPONSE: STOPPED, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x09	Test TCP connection
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

STOP	Length Low Byte	Length High Byte
0x09	0x00	0x00

Cmd: 09 00 00

### A.3.10 STOPPED (Test Case Stopped)

The Tagsurance response to 'STOP' from client.

The Tagsurance will stop the inline case that has been initiated. Response is provided after the case has been stopped and the system returned to idle state.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x0A	Connect
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

STOPPED	Length Low Byte	Length High Byte
0x0A	0x00	0x00

Cmd: 0A 00 00

### A.3.11 GTR (Get Test Result)

INIT STATE: inline, idle

NEXT STATE: no change

FUNCTION: acquire measurement data from the results buffer

TAGSURANCE RESPONSE: TR, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x10	Get Test Result
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

GTR	Length Low Byte	Length High Byte
0x10	0x00	0x00

Cmd: 10 00 00

### A.3.12 TR (Test Result)

TR header is common, but result data format depends on the calling command.

#### The Tagsurance response to: "GTR" from client

The Tagsurance will check the status of the inline case results buffer and return the first data that has not been read. For example, if "GTR" is provided after three triggered measurements, included in the response are the data from the measurement that was triggered first. Second "GTR" will return second dataset, etc.

#### Tagsurance response to: "READ/WRITE/SWEEP/KILL" from client

In case of WRITE, result data includes an error code.

In case of READ, result data includes error code and the acquired data, separated by '/'.

In case of SWEEP, result data includes threshold values at test frequencies, separated by '\tab'.

In case of ENCODE/KILL, result data includes latest task indicator and error code, separated by '/'.

INTERPRETATION OF TASK IDENTIFIER BYTE		
0x00	0 <sub>10</sub>	Inventory (and access) sequence
0x01	1 <sub>10</sub>	EPC memory programming
0x02	2 <sub>10</sub>	User memory programming
0x03	3 <sub>10</sub>	Kill password programming
0x04	4 <sub>10</sub>	Access password programming
0x05	5 <sub>10</sub>	Lock bit programming
0x06	6 <sub>10</sub>	Kill procedure



Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x11	Test Result
1	Length Low	0xXX	Length Low
2	Length High	0xXX	Length High
3	Failed/Passed	0x00    0x01	0x00 → Failed 0x01 → Passed
...	Result data	...	Other Test Results...
...	End Sequence	0x0D0A	End sequence

Example Transmit:

TR	Length Low Byte	Length High Byte	Failed	Result	End Sequence
0x11	0xXX	0xXX	0x00	"all information"	0x0D0A

Cmd: 11 10 00 00 61 6C 6C 20 69 6E 66 6F 72 6D 61 74 69 6F 6E 73 0D 0A

The syntax of result data is similar to result lines in the Tagsurance GUI log file. The first provided test result, after loading a new case with the "LSC" command, includes the log file header lines. Following test results consist of a single result line.

*The example result data of the first "TR" after the case loading*

Tagsurance	10.1.2013	13:55			
Sensitivity test specifications		Frequency [MHz]	Lowest power power [dBm]		Highest tested power [dBm]
	Lower control limit [dBm]		Upper control limit [dBm]		Uncertainty criterion [dBm]
Sensitivity 1	866,00	-10,00	25,00	-10,00	25,00 0,25
Case specifications (Point tolerance: 0)					
Frequency [MHz]		925,90	866,00		
Power [dBm]		5,00			
Mode		must respond	Sensitivity 1		
Results					
Time stamp	Pass/Fail	Tested points	Sensitivity 1		
13:55:16	FAIL	0	2/25,00		

*The example result data of other (second, third and so on) "TR" after the case loading*

13:55:18	FAIL	0	2/25,00
----------	------	---	---------

*The example result data of simple encode command (successful EPC write + lock)*

5/0
-----

### A.3.13 GBS (Get Buffer Status)

INIT STATE: inline, idle

NEXT STATE: no change

FUNCTION: test inline test results data buffer state

TAGSURANCE RESPONSE: BS, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x12	Get Buffer Status
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

GBS	Length Low Byte	Length High Byte
0x12	0x00	0x00

Cmd: 12 00 00

### A.3.14 BS (Buffer Status)

The Tagsurance response to result buffer state query initiated by “GBS” from client.

The Tagsurance will count the amount of unread data in the results buffer, and check if the data bank has overflowed. The maximum amount of unread data in the buffer is 65535 results.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x13	Get Test Result
1	Length Low	0x03	Length Low
2	Length High	0x00	Length High
3	Nlow	0xXX	Number of results in buffer (low byte)
4	Nhigh	0xXX	Number of results in buffer (high byte)
5	Buffer OVF	0x00   0x01	buffer OVF (0x01), no error (0x00)

Example Transmit:

BS	Length Low Byte	Length High Byte	Nlow	Nhigh	Buffer overflow
0x13	0x03	0x00	0x01	0x00	0x00

Cmd: 13 03 00 01 00 00

### A.3.15 READ (Read Tag Memory)

INIT STATE: inline, idle

NEXT STATE: no change

FUNCTION: perform a single tag memory read

TAGSURANCE RESPONSE: TR, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x30	Simple Read Tag Memory
1	Length Low	0x06	Length Low
2	Length High	0x00	Length High
3	Flow	0xXX	Frequency (F/100kHz's) (low byte)
4	Fhigh	0xXX	Frequency (F/100kHz's) (high byte)
5	Power	0xXX	Power Level (dB*4+128)
6	Memory Bank	0xXX	Tag Memory to be read
7	Word Address	0xXX	First word address to be read (0 to 127)
8	Word Count	0xXX	Number of words to be read (1 to 127)

Example Transmit:

READ	Length Low Byte	Length High Byte	Flow (866MHz)	Fhigh (866MHz)	Power (-2dBm)	Bank (TID)	Addr	WCount (2 pcs)
0x30	0x06	0x00	0xD4	0x21	0x78	0x10	0x00	0x02

Cmd: 30 06 00 D4 21 78 10 00 02

### A.3.16 WRITE (Simple Write)

INIT STATE: inline, idle

NEXT STATE: no change

FUNCTION: perform a single tag memory write

TAGSURANCE RESPONSE: TR, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x31	Simple Write Tag Memory
1	Length Low	0xXX	Length Low
2	Length High	0x00	Length High
3	Flow	0xXX	Frequency (F/100kHz's) (low byte)
4	Fhigh	0xXX	Frequency (F/100kHz's) (high byte)
5	Power	0xXX	Power Level (dB*4+128)
6	Memory Bank	0xXX	Tag Memory to be written
7	Word Address	0xXX	First word address to be written (0 to 127)
8	Word Count	0xXX	Number of words to be written (1 to 127)
9...	Data	0x...	Data to be written

Example Transmit:

WRITE	Length Low Byte	Length High Byte	Flow (866MHz)	Fhigh (866MHz)	Power (-2dBm)	Bank (EPC)	Addr	WCount (1 pcs)	Data
0x31	0x08	0x00	0xD4	0x21	0x78	0x01	0x02	0x01	0x0000

Cmd: 31 08 00 D4 21 78 01 02 01 00 00

### A.3.17 SWEEP (Simple Sweep)

INIT STATE: inline, idle

NEXT STATE: no change

FUNCTION: perform a single response threshold sweep for a tag

TAGSURANCE RESPONSE: TR, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x32	Simple Threshold Sweep
1	Length Low	0x06	Length Low
2	Length High	0x00	Length High
3	Fstart,low	0xXX	Start freq (F/100kHz's) (low byte)
4	Fstart,high	0xXX	Start freq (F/100kHz's) (high byte)
5	Fstop,low	0xXX	Stop freq (F/100kHz's) (low byte)
6	Fstop,high	0xXX	Stop freq (F/100kHz's) (high byte)
7	Fstep,low	0xXX	Freq step (F/100kHz's) (low byte)
8	Fstep,high	0xXX	Freq step (F/100kHz's) (high byte)

Example Transmit:

SWEEP	Length Low Byte	Length High Byte	Start,low (800MHz)	Start,high (800MHz)	Stop,low (900MHz)	Stop,high (900MHz)	Step,low (10MHz)	Step,high (10MHz)
0x32	0x06	0x00	0x40	0x1F	0x28	0x23	0x64	0x00

Cmd: 32 06 00 40 1F 28 23 64 00

### A.3.18 ENCODE (All-In-One)

INIT STATE: inline, idle

NEXT STATE: no change

FUNCTION: perform encoding and locking operations for tag memories

TAGSURANCE RESPONSE: TR, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x33	Encode tag
1	Length Low	0xXX	Length Low
2	Length High	0xXX	Length High
3	Flow	0xXX	Frequency (F/100kHz's) (low byte)
4	Fhigh	0xXX	Frequency (F/100kHz's) (high byte)
5	Power	0xXX	Power Level (dB*4+128)
6...9	Access password <sup>2</sup>	0XXXXXXXX	All 0's: no access, else: access after inventory
10	EPC memory action byte	0xXX	Operations to perform for EPC memory <sup>3</sup>
.	Word Address	0xXX	First word address to be written (0 to 127) <sup>1</sup>
.	Word Count	0xXX	Number of words to be written (1 to 127) <sup>1</sup>
.	EPC Data	0xXX	Data to be written <sup>1</sup>
.	USER memory action byte	0xXX	Operations to perform for USER memory <sup>3</sup>
.	Word Address	0xXX	First word address to be written (0 to 127) <sup>1</sup>
.	Word Count	0xXX	Number of words to be written (1 to 127) <sup>1</sup>
.	USER Data	0xXX	Data to be written <sup>1</sup>
.	KILL-PWD action byte	0xXX	Operations to perform for kill password <sup>3</sup>
.	KILL-PWD data	0xXX	Kill password (4 bytes, MSB first) <sup>1</sup>
.	ACCESS-PWD action byte	0xXX	Operations to perform for access password <sup>3</sup>
.	ACCESS-PWD data	0xXX	Access password (4 bytes, MSB first) <sup>1</sup>

<sup>1</sup> Defined only if data is programmed with data (action byte value is: 0x01...0x05).

<sup>2</sup> If access password provided is 0x00000000, access is not performed.

<sup>3</sup> Memory programming operations for different action byte values.

Action byte	Encoding action	Write operation	Lock operation
0x00	no action	-	-
0x01	write	program data to memory	-
0x02	write + unlock	program data to memory	unlock memory
0x03	write + lock	program data to memory	lock memory <sup>1</sup>
0x04	write + permalock	program data to memory	lock memory permanently <sup>1</sup>
0x05	write + permaunlock	program data to memory	unlock memory permanently
0x06	unlock	-	unlock memory
0x07	lock	-	lock memory <sup>1</sup>
0x08	permalock	-	lock memory permanently <sup>1</sup>
0x09	permaunlock	-	unlock memory permanently

<sup>1</sup> EPC, USER: write protection, RESERVED (password memory): read and write protection.

#### Example Transmit 1:

Frequency 866MHz, Power 15dBm, no access after inventory

Write 4 bytes words to EPC memory, starting form word address 0x02, and set write lock on

Write access password (0x01020304)

ENCODE	Length Low Byte	Length High Byte	Flow (866MHz)	Fhigh (866MHz)	Power (15dBm)	ACC- PWD (MSB)	ACC- PWD ...	ACC- PWD ...	ACC- PWD (LSB)
0x33	0x15	0x00	0xD4	0x21	0xBC	0x00	0x00	0x00	0x00

EPC memory action byte (write + lock)	Word address (0x02)	Word count (4 bytes)	Data (4 bytes)
0x03	0x02	0x02	0x11223344

USER memory action byte (no action)	KILL-PWD action byte (no action)
0x00	0x00

ACC-PWD action byte (write)	Byte 1 (MSB)	Byte 2	Byte 3	Byte 4 (LSB)
0x01	0x01	0x02	0x03	0x04

Cmd: 33 15 00 D4 21 BC 00 00 00 00 03 02 02 11 22 33 44 00 00 01 01 02 03 04

#### Example Transmit 2:

Frequency 866MHz, Power 15dBm, no access

Lock EPC memory, and write and lock access password (0x01020304)

ENCODE	Length Low Byte	Length High Byte	Flow (866MHz)	Fhigh (866MHz)	Power (15dBm)	ACC- PWD (MSB)	ACC- PWD ...	ACC- PWD ...	ACC- PWD (LSB)
0x33	0x0F	0x00	0xD4	0x21	0xBC	0x00	0x00	0x00	0x00

EPC memory action byte (no action)	USER memory action byte (no action)	KILL-PWD action byte (no action)
0x00	0x00	0x00

ACC-PWD action byte (write + lock)	Byte 1 (MSB)	Byte 2	Byte 3	Byte 4 (LSB)
0x03	0x01	0x02	0x03	0x04

Cmd: 33 0F 00 D4 21 BC 00 00 00 00 00 00 03 01 02 03 04

### A.3.19 KILL

INIT STATE: inline, idle

NEXT STATE: no change

FUNCTION: perform a single kill operation for a tag

TAGSURANCE RESPONSE: TR, or ERR

Notes!

A tag will not accept kill command if kill password is zero. If the kill password provided with the command is zero, the Tagsurance will try to program kill password before attempting to kill the tag.

If kill password is locked, reprogramming can only be performed if the tag is in secured state. If the access password provided with the command is non-zero, the Tagsurance will perform inventory with access.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0x34	Kill tag
1	Length Low	0x0B	Length Low
2	Length High	0x00	Length High
3	Flow	0xXX	Frequency (F/100kHz's) (low byte)
4	Fhigh	0xXX	Frequency (F/100kHz's) (high byte)
5	Power	0xXX	Power Level (dB*4+128)
6...9	Access password <sup>1</sup>	0XXXXXXXX	0x00: no access, else: access after inventory
10...13	Kill password <sup>2</sup>	0XXXXXXXX	0x00: written before kill

<sup>1</sup> If access password provided is 0x00000000, access is not performed.

<sup>2</sup> If kill password provided is 0x00000000, password memory is programmed before kill operation. This will require access, if reserved memory is locked, and access password is non-zero.

Example Transmit: Frequency 866MHz, Power 15dBm, no access, no pre-programmed kill password.

KILL	Length Low Byte	Length High Byte	Flow (866MHz)	Fhigh (866MHz)	Power (15dBm)	ACC- PWD (MSB)	ACC- PWD ...	ACC- PWD ...	ACC- PWD (LSB)
0x34	0x0B	0x00	0xD4	0x21	0xBC	0x00	0x00	0x00	0x00

KILL-PWD (MSB)	KILL-PWD ...	KILL-PWD ...	KILL-PWD (LSB)
0x00	0x00	0x00	0x00

Cmd: 34 0B 00 D4 21 BC 00 00 00 00 00 00 00 00



### A.3.20 TCP Test

INIT STATE: handshake (within 10s from opening TCP connection)

NEXT STATE: no change

FUNCTION: perform handshake with the Tagsurance GUI

TAGSURANCE RESPONSE: TCP Ready, or ERR

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0xF0	Test TCP connection
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

TCP Test	Length Low Byte	Length High Byte
0xF0	0x00	0x00

Cmd: F0 00 00

### A.3.21 TCP Ready

Tagsurance answer to 'TCP Test' from client.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0xF1	Connect
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High

Example Transmit:

TCP Ready	Length Low Byte	Length High Byte
0xF1	0x00	0x00

Cmd: F1 00 00

### A.3.22 ERR (Error from Tagsurance)

The Tagsurance response in case of an error.

Byte No.	Function	Value (HEX)	Description
0	Operation code, command	0xFF	Error from Tagsurance
1	Length Low	0x00	Length Low
2	Length High	0x00	Length High
3	Parameter Byte	0x00 ... 0xFF	Error code

Example:

ERR	Length Low Byte	Length High Byte	Error Code
0xFF	0x01	0x00	0x00

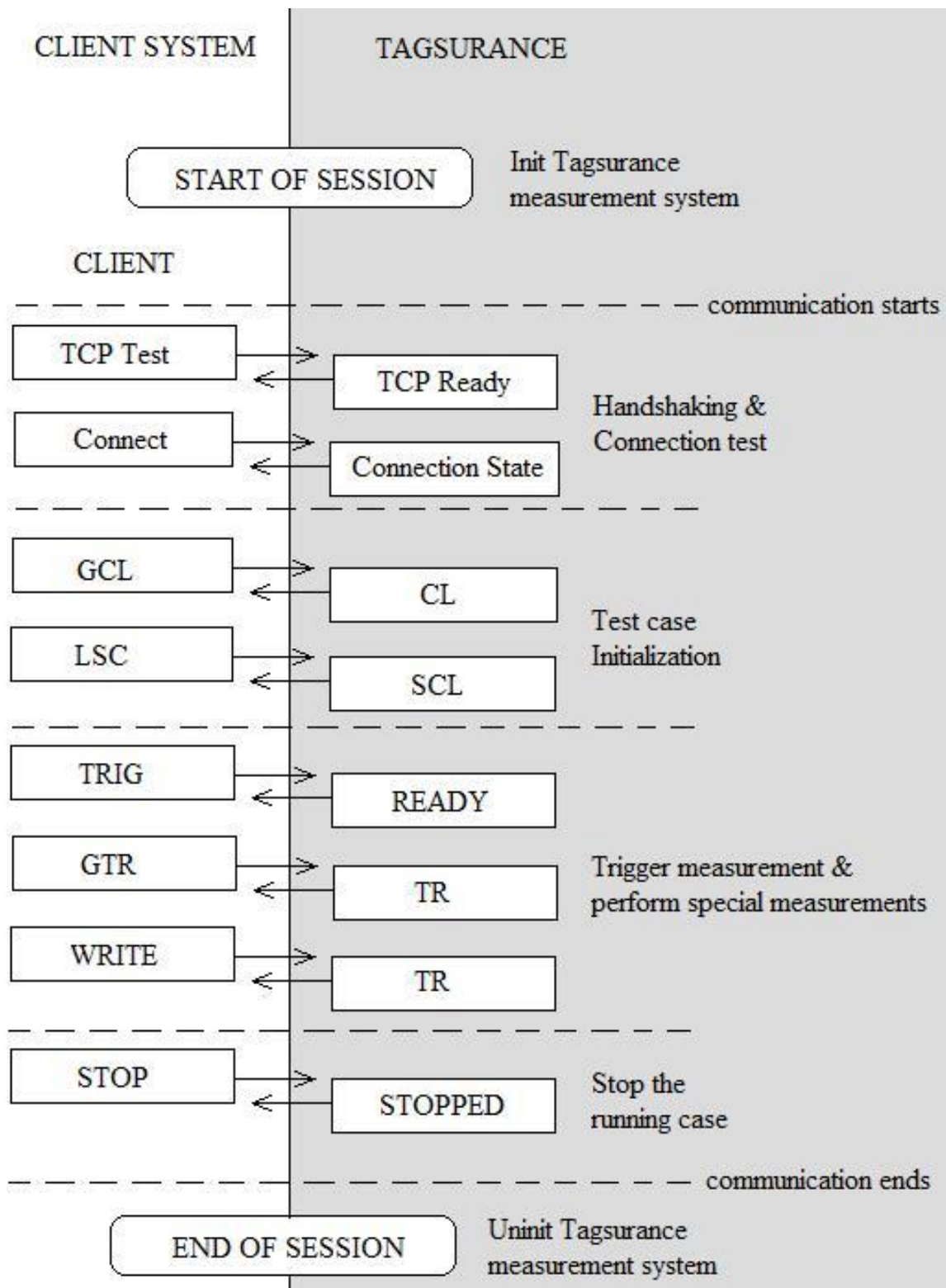
Cmd: FF 01 00 01

Error codes:

Code	Command	Description
0x00	Invalid command	The command provided was not valid
0x01	Handshaking failed	'TCP test' not received during handshake
0x02	Device busy	Device is busy and command cannot be performed
0x03	RS232 connection lost	No connection to Tagsurance Tester unit
0x04	Bad request	The requested action cannot be performed in current operation mode
0x05	Reception timeout	Timeout occurred during command reception
0x10	Case Init Error 0	Case file not found
0x11	Case Init Error 1	Case file is corrupted
0x12	Case Init Error 2	Case parameters are invalid
0x13	Case Init Error 3	Licence options exceeded
0x21	Invalid parameters	Command parameters are invalid or insufficient
0x22	Licence error	Current licence does not allow the action requested
0x30	Buffer Empty	No data available in the result data buffer
0x31	Buffer OVF	Result data buffer has overflowed

## A.4 Application Example

Below is described a simple test session example.



## **A.5    *Heartbeat***

### **A.5.1        Automatic Connection Monitoring**

The Tagsurance GUI remote access interface monitors the connection state continuously, and automatically closes the network connection when a connection break is detected. Automatic connection monitoring mechanism sets some requirements for the client system, but makes connection management easier and more flexible in case of communication errors and connection issues. For handling a connection timeout there are two methods, either of which should be implemented in the client software.

#### **METHOD 1** (Tagsurance GUI initiated heartbeat)

The Tagsurance GUI keeps track of data reception rate and the connection timeout period value is 1 min. If no new data is received during the timeout period, the reception buffer will be cleared and a connection test command sent to the network client system ("TCP Test"). The client should respond to this command within 5 seconds, or the connection will be closed. To implement this kind of heartbeat mechanism, the client system should have a mechanism which can detect a connection test command from the Tagsurance and respond within the given 5 seconds period.

#### **METHOD 2** (Client system initiated heartbeat)

Data reception rate tracking cannot be disabled, but the client may prevent the timeout by sending data at least every minute. To implement this kind of heartbeat mechanism, the client system could e.g. keep track of data rate and send an occasional connection test ("TCP Test") whenever the transmission interval is close to connection timeout value (1 min). When this kind of solution is used, it is recommended to leave some margin to timeout value because the TCP connection can have jitter, and the Tagsurance GUI will detect timeout based on the received data. Setting the heartbeat transmission interval to e.g. 50 seconds should allow enough error margin for possible network delays.

### **A.5.2        Recommended Communication Reset Routine**

In addition to automatic connection initialization, the connection timeout mechanism also provides means for resetting communication after communication error, or misspelt command frame (e.g. where data length is not equal to data byte count). Whenever the client detects a transmission error, or if the Tagsurance GUI is not responding correctly, the communication can simply be put on halt until the Tagsurance GUI issues connection test command ("TCP Test"). The Tagsurance GUI clears the reception buffer before sending this frame, and by responding to this command, the client can ensure that data buffers are reset and communication can be continued from a fresh base. This kind of issues should be rare in a well-established system, but can occur e.g. due to network issues or strong environmental interferences. Implementing the recommended reset routine makes the system more robust and more tolerant of communication issues.

## Appendix B:

### UDP Test Data Broadcasting Interface

The Tagsurance GUI UDP Broadcast Interface is an one-directional interface used to send test data to third party software. UDP services can be occupied from system settings dialog, or by including proper keywords to the Tagsurance GUI program call arguments (see appendix C). When UDP broadcasting is enabled, the Tagsurance GUI transmits an UDP frame including measurement data after each test performed. Data format and contents depend on the operating mode.

**The Impinj ItemEncode mode** is intended as a service which enables easy and efficient integration of the Tagsurance test system and the Impinj ItemEncode encoding platform. In this mode, the Tagsurance GUI sends pass/fail byte and result data from the first read test in the Impinj ItemEncode compatible frame format (see table below). The Impinj ItemEncode uses TID to bind performance test outcome with the tag to-be-encoded and to determine encoding actions. In order to allow this operation, the 1<sup>st</sup> read test in the Tagsurance performance test case should be TID read over the whole TID number.

Byte No.	Function	Value (HEX)	Description
0..3	Header	0x76746965	Impinj ItemEncode data frame identifier
4	Failed/Passed	0x00    0x01	0x00 → Failed 0x01 → Passed
5	Data Length Low	0x00	Data Length Low (represents word count)
6	Data Length High	0x00	Data Length High (represents word count)
7...	Result data	...	Read1 data (TID)

**Non-parsed data broadcasting mode** is intended as a generic service for third party logging applications, allowing e.g. easy test log generation in a remote target. In this mode test log file contents (excluding yield and count information) are sent test by test (i.e. line-by-line) in ASCII format. Log file header is included with the first data frame. Frame format is described in the table below. Data length includes pass/fail byte, data, and end sequence characters.

Byte No.	Function	Value (HEX)	Description
0	Header	0x00	Connect
1	Data Length Low	0xFF	Data Length Low <sup>1</sup>
2	Data Length High	0xFF	Data Length High <sup>1</sup>
3	Failed/Passed	0x00    0x01	0x00 → Failed 0x01 → Passed
4...	Result data	...	Test data <sup>2,3</sup>
...	End Sequence	0x0D0A	End sequence

<sup>1</sup>Data length counts in failed/passed byte (1 byte), result data bytes, and end sequence bytes (2 bytes).

<sup>2</sup>Test data includes one log file string per frame.

<sup>3</sup>Log file header is included to the first frame.

**Parsed data broadcasting mode** is intended as a generic service for third party logging and control applications, allowing easy post-processing of result data. In this mode, the result data is parsed to sub-frames each having a recognizable header. Frame format is presented and sub-frame identifier listed in the tables below.

Byte No.	Function	Value (HEX)	Description
0	Header	0x01	Connect
1	Data Length Low	0xXX	Data Length Low <sup>1</sup>
2	Data Length High	0xXX	Data Length High <sup>1</sup>
3	Failed/Passed	0x00    0x01	0x00 → Failed 0x01 → Passed
4...	Result data	...	Test result SUBFRAMES (see next table)
...	End Sequence	0x0D0A	End sequence

<sup>1</sup>Data length counts in failed/passed byte (1 byte), result data bytes, and end sequence bytes (2 bytes).

SUB-FRAME TYPE	FIELD NAME	Value (HEX)	Description
POINT TEST RESULTS	Identifier	0x80	‘P’
	Task index	0x01	Always 1
	Data length	0xXX	(Number of points)/8
	Data	.....	Point test data (LSB is 1 <sup>st</sup> point test result)
WRITE TEST(S) RESULTS	Identifier	0x87	‘W’
	Task index	0xXX	Write test index (>0)
	Data length	0x01	Always 1 (error code)
	Data	0xXX	Write test error code (0x00 = no error)
READ TEST(S) RESULTS	Identifier	0x82	‘R’
	Task index	0xXX	Read test index (>0)
	Data length	0xXX	Error code + number of bytes read
	Error code	0xXX	Read test error code (0x00 = no error)
	Data	0xXX	Bytes read (all 0x00 if operation failed)
SWEEP TEST RESULTS	Identifier	0x83	‘S’
	Task index	0x01	Always 1 (max 1 sweep test allowed per test case)
	Data length	0xXX	$[(F_{high} - F_{low}) / F_{step}] + 1$
	Data	0xXX	Threshold powers ( (byte-128)/4 [dBm] )
SENSITIVITY TEST RESULTS	Identifier	0x67	‘C’
	Task index	0xXX	Sensitivity test index (>0)
	Data length	0x01	Always 1 (threshold power at test frequency)
	Data	0xXX	Threshold power ( (byte-128)/4 [dBm] )

## Appendix C:

# Enabling Tagsurance GUI Options by Using Program Call Arguments

Some Tagsurance GUI features need to be reconfigured at startup. Such are TCP remote access interface, UDP result broadcasting services, and serial data protection modes. To avoid manual initialization, it is possible to instruct the GUI to configure these settings automatically, by including certain keywords (see table below) to program call. In Windows operating system, a convenient way of doing this is to create a batch file (.bat) which can be run to open GUI with the required options properly configured.

The following steps are required to create a valid batch file:

1. Create a new text file
2. Write tagsurance\_gui.exe program call with arguments  
e.g. "<program path>\Tagsurance\_GUI.exe – ARGUMENT1 ARGUMENT2 ..."
  - a. Some operating systems do not allow space-characters in the file name or path. If the target file cannot be found, try modifying the path and executable name.
  - b. " -- " marks the start of argument list
  - c. if there are multiple arguments, they should be space-separated
3. Save the file with .bat extension.

ENABLED SYSTEM FEATURE	RELATED PROGRAM CALL ARGUMENTS
<b>TCP REMOTE ACCESS INTERFACE<sup>1</sup></b>	TCP_EN TCP_PORT_portNumber - portNumber is host PC transmit port
<b>RESULT DATA BROADCASTING<sup>2</sup></b> - Impinj ItemEncode mode - Send result data, non-parsed - Send result data, parsed	IMPINJ_ITEMENCODE_EN_localPort_RemotePort_RemoteIP BROADCAST_NONPARSED_DATA_EN_localPort_RemotePort_RemoteIP BROADCAST_PARSED_DATA_EN_localPort_RemotePort_RemoteIP - localPort is host PC transmit port - remotePort is target system reception port (default:36364) - remoteIP is target system IP address
<b>SERIAL DATA PROTECTION<sup>3</sup></b> - Send data with checksum - Send data with framesync	CHECKSUM_EN FRAMESYNC_EN
<b>FREEZE SCREEN CONTROL<sup>4</sup></b>	FROZEN_CONTROLS

<sup>1</sup> both options have to be defined to initialize TCP mode.

<sup>2</sup> only one broadcasting service type can be initiated at a time.

<sup>3</sup> it is possible to enable checksum, framesync, or both.

<sup>4</sup> this option disables window closing possibility

# Appendix D:

## Summary of Serial Interface Commands

### DEVICE SETTINGS

SYSTEM CONFIGURATION			
Function	Command	Response	Next state
Enable external trigger (default)	'PTE'	(none)	IDLE
Disable external trigger	'PTI'	(none)	IDLE
Set trigger (ext) to rising edge (default)	'PTH'	(none)	IDLE
Set trigger (ext) to falling edge	'PTL'	(none)	IDLE
Enable internal gen2 chip	'POC'	(none)	IDLE
Disable internal gen2 chip (default)	'POA'	(none)	IDLE
Set carrier time (default: 480=[1][224]=2,5ms)	'PC'[time_high][time_low]	(none)	IDLE
Set pass signal to static mode (default)	'PIPS'	(none)	IDLE
Set pass signal to pulsed mode	'PIPP'	(none)	IDLE
Enable "failed = HIGH" option	'PIPM'	(none)	IDLE
Disable "failed = HIGH" option (default)	'PIPN'	(none)	IDLE
Save settings	'PFS'	(none)	IDLE
Load saved settings	'PFL'	[fbyte <sub>L</sub> ][fbyte <sub>H</sub> ] [time <sub>H</sub> ][time <sub>L</sub> ]	IDLE
Reset factory default settings	'PFR'	(none)	IDLE

### INTERPRETATION AND DEFINITION OF CARRIER TIME

#### calculation formula

carrier\_time[ms] = time/160 – 0,5ms

time = (carrier\_time[ms] + 0,5ms) \* 160

#### valid range

112 - 560 = 0,2ms - 3ms

### INTERPRETATION OF THE FBYTE DATA

bit	FBYTE <sub>L</sub>	FBYTE <sub>H</sub>
0	trigger mode (0->ext, 1->int, default: ext)	(RFU)
1	trigger edge (0->falling, 1->rising, default: rising)	(RFU)
2	pass signal mode (0->static, 1->pulsed, default: static)	(RFU)
3	failed = HIGH option (0->disabled, 1->enabled, default: disabled)	(RFU)
4	(RFU)	(RFU)
5	(RFU)	(RFU)
6	(RFU)	(RFU)
7	(RFU)	(RFU)



## PERFORMANCE TESTING

INLINE MEASUREMENT COMMANDS			
Function	Command	Response	Next state
<b>Upload case data</b>			
• header	['L'][Data Length H][Data Length L]	error code <sup>4</sup>	IDLE
• data for point tests	['P'][Tolerance][Npoints][Mode + f <sub>H</sub> ][f <sub>L</sub> ][Power]		
• data for read tasks	['R'][Bank + f <sub>H</sub> ][f <sub>L</sub> ][Power][Word Pointer] [Word Count][Repetitions + Tolerance]		
• data for write tasks	['W'][Bank + f <sub>H</sub> ][f <sub>L</sub> ][Power][Word Pointer] [Repetitions + Tolerance][Increment] [Word Count][Data]		
• data for sweep task	['S'][Start f <sub>H</sub> ][Start f <sub>L</sub> ][Stop f <sub>H</sub> ][Stop f <sub>L</sub> ] [Step f <sub>H</sub> ][Step f <sub>L</sub> ]		
• data for sensitivity test	['C'][f <sub>H</sub> ][f <sub>L</sub> ][P <sub>H</sub> ][P <sub>L</sub> ][P <sub>LC</sub> ][P <sub>UC</sub> ][Uncertainty]		
<b>Start (loaded) case</b>	'C'	error code <sup>4</sup>	WAIT FOR TRIG <sup>2</sup>
<b>Trigger</b>	'T'	pass/fail &data <sup>5</sup>	WAIT FOR TRIG
<b>Stop (running) case<sup>1</sup></b>	'X'	(none)	IDLE
<b>Save (active) case</b>	['F'][case number (0-4)]	error code <sup>4</sup>	IDLE
<b>Load saved case</b>	['O'][case number (0-4)]	error code <sup>4</sup>	IDLE
<b>Get active case data</b>	'PD'	case data <sup>3</sup>	IDLE

<sup>1</sup> any char other than 'T' also stops the running case and causes the device to return to IDLE state. 'X' is recommended.

<sup>2</sup> If no valid case is loaded, the device will return error and return to IDLE state.

<sup>3</sup> If no valid case is loaded, the device will return {0x00, 0x00} (i.e. number of data bytes is zero).

<sup>4</sup> Byte count and data will follow if the command is valid (user must check the error flag, 0x00=no error).

OTHER MEASUREMENT FUNCTIONS			
Function	Command	Response	Next state
<b>Threshold sweep</b>	['S'][Start f <sub>H</sub> ][Start f <sub>L</sub> ][Stop f <sub>H</sub> ][Stop f <sub>L</sub> ][Step f <sub>H</sub> ][Step f <sub>L</sub> ]	[Err][N Bytes][Data] <sup>1</sup>	IDLE
<b>Read</b>	['R'][Bank + f <sub>H</sub> ][f <sub>L</sub> ][Power][Word Pointer] [Word Count][Repetitions + Tolerance]	[Err][ErrByte][Data] <sup>2</sup>	IDLE
<b>BlockWrite</b>	['W'][Bank + f <sub>H</sub> ][f <sub>L</sub> ][Power][Word Pointer][Word Count][Data]	[Err][ErrByte] <sup>3</sup>	IDLE

<sup>1</sup>Byte count and data will follow if the command is valid (user must check the error flag, 0x00=no error)

<sup>2</sup>errByte (0x00 = no error) followed by the data read (repeated string of 0x00's if read was failed)

<sup>3</sup>errByte (0x00 = no error)

## ENCODING, LOCKING, AND KILLING

TAG ENCODING, LOCKING, AND KILLING FUNCTIONS			
Function	Command	Response	Next state
BASIC COMMANDS			
Encode	['H'][Data Length H][Data Length L] [f <sub>H</sub> ][f <sub>L</sub> ][Power][Acc-pwd <sub>1,MSB</sub> ][Acc-pwd <sub>2</sub> ][Acc-pwd <sub>3</sub> ][Acc-pwd <sub>4</sub> ] [EPC Word count][EPC Word pointer][EPC Data] [User Word count][User Word pointer][User Data] [Enab kill-pwd prog][Kill-pwd <sub>1,MSB</sub> ][Kill-pwd <sub>2</sub> ][Kill-pwd <sub>3</sub> ][Kill-pwd <sub>4</sub> ] [Enab acc-pwd prog][Acc-pwd <sub>1,MSB</sub> ][Acc-pwd <sub>2</sub> ][Acc-pwd <sub>3</sub> ][Acc-pwd <sub>4</sub> ] [Enab lock prog][Payload <sub>1,MSB</sub> ][Payload <sub>2</sub> ][Payload <sub>3</sub> ]	[Err] <sup>1</sup> [Task][ErrByte] <sup>2-4</sup>	IDLE
Lock	['D'][f <sub>H</sub> ][f <sub>L</sub> ][Power][Repetitions + Tolerance] [Payload <sub>1,MSB</sub> ][Payload <sub>2</sub> ][Payload <sub>3</sub> ]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	IDLE
Kill	['K'][f <sub>H</sub> ][f <sub>L</sub> ][Power] [Acc-pwd <sub>1,MSB</sub> ][Acc-pwd <sub>2</sub> ][Acc-pwd <sub>3</sub> ][Acc-pwd <sub>4</sub> ] [Kill-pwd <sub>1,MSB</sub> ][Kill-pwd <sub>2</sub> ][Kill-pwd <sub>3</sub> ][Kill-pwd <sub>4</sub> ]	[Err] <sup>1</sup> [Task][ErrByte] <sup>2-4</sup>	IDLE
CUSTOM COMMAND SET			
Inventory	['I'][f <sub>H</sub> ][f <sub>L</sub> ][Power][Acc-pwd <sub>1,MSB</sub> ][Acc-pwd <sub>2</sub> ][Acc-pwd <sub>3</sub> ][Acc-pwd <sub>4</sub> ]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	IDLE/ENCODING
BlockWrite	['E']['W'][Bank][Word pointer][Word count][Data]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	ENCODING
Read	['E']['R'][Bank][Word pointer][Word count]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	ENCODING
Lock	['E']['L'][Payload <sub>1,MSB</sub> ][Payload <sub>2</sub> ][Payload <sub>3</sub> ]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	ENCODING
Kill	['E']['K'][Kill-pwd <sub>1,MSB</sub> ][Kill-pwd <sub>2</sub> ][Kill-pwd <sub>3</sub> ][Kill-pwd <sub>4</sub> ]	[Err] <sup>1</sup> [ErrByte] <sup>2,4</sup>	ENCODING
End session	['X']	-	IDLE

<sup>1</sup>Err represents system error code (0x00=no error). For description of the codes, see Section 5.11: "Error handling"

<sup>2</sup>ErrByte (and task) will follow Err if the command is valid (Err=0x00)

<sup>3</sup>Task indicates the last task performed before end session (See task indices table below)

<sup>4</sup>ErrByte represents error code from the last task (0x00 = no error)

INTERPRETATION OF TASK IDENTIFIER BYTE		
0x00	0 <sub>10</sub>	Inventory
0x01	1 <sub>10</sub>	EPC memory programming
0x02	2 <sub>10</sub>	User memory programming
0x03	3 <sub>10</sub>	Kill password programming
0x04	4 <sub>10</sub>	Access password programming
0x05	5 <sub>10</sub>	Lock bit programming
0x06	6 <sub>10</sub>	Kill procedure

COMBINING ENCODING WITH PERFORMANCE TESTING			
Function	Command	Response	Next state
Activate encode after performance test option	'PME'	(none)	IDLE
Deactivate encode after performance test option	'PMD'	(none)	IDLE
Reinitialize inline case	'C'	Err <sup>1</sup>	WAIT FOR TRIG

<sup>1</sup>Err represents system error code (0x00=no error). For description of the codes, see Section 5.11: "Error handling"

## Appendix E:

### Summary of Error Indicators

INTERPRETATION OF SYSTEM ERROR CODE(S) (0 -> no error, 1 --> error)		
Bit 0	invalid input data sequence	valid range: 1 to 510 bytes
Bit 1	timeout during data reception	maximum delay between data bytes: 1000ms
Bit 2	invalid memory bank	valid range: 0 to 4
Bit 3	invalid word count	valid range: >0 (read test), 1-8 (write test)
Bit 4	invalid command characters	valid values: 'P', 'R', 'W', and 'S'
Bit 5	output data size required too large	valid range: 0 to 100 bytes
Bit 6	invalid frequency	valid range: 860 to 960MHz (extended: 800 to 1100MHz)
Bit 7	Invalid power	valid range: -10 to +25dBm
0xFF	license error	requested task not covered by current license options

INTERPRETATION OF READ/WRITE TASK ERROR BYTE			
0x00	0 <sub>10</sub>	No errors	Tolerance condition was met
0x01	1 <sub>10</sub>	Connection failed	Tag did not respond to query
0x02	2 <sub>10</sub>	Command failed	Tag did not respond to the command
0x03	3 <sub>10</sub>	Inventory failed	Tag responds to query, but inventory failed
0x04	4 <sub>10</sub>	Tag error message 1	Other error (not covered by the other tag error codes)
0x34	52 <sub>10</sub>	Tag error message 2	Memory overrun (too high memory address)
0x44	68 <sub>10</sub>	Tag error message 3	Memory locked (memory cannot be read/written)
0xB4	180 <sub>10</sub>	Tag error message 4	Insufficient power (carrier level too low)
0xF4	244 <sub>10</sub>	Tag error message 5	Non-specified (error-specific codes not supported by the tag)

INTERPRETATION OF SENSITIVITY TEST TASK ERROR BYTE			
0x00	0 <sub>10</sub>	No errors	Tolerance condition was met
0x01	1 <sub>10</sub>	Out of tolerances	Tolerance condition was not met, the threshold power exceeded the upper control limit or was below the lower control limit
0x02	2 <sub>10</sub>	Out of range	The response power is out of measurement range