go2signals

Manual

go2ANALYSE

Product Version v24.1.0

November 3, 2023

PROCITEC®
HOUSE OF SIGNALS

Imprint

PROCITEC GmbH

Rastatter Straße 41

D-75179 Pforzheim

Germany

Phone: +49 7231 15561 0

Fax: +49 7231 15561 11

Email: service@procitec.com

Web: www.procitec.com

Authorised Representative:

Dipl.-Ing. (FH) Dipl.-Inf. (FH) Jens Heyen

Registration Court:

HRB 504702 Amtsgericht Mannheim

Tax ID:

DE 203 881 534

Document ID:

PROCITEC-IMA-go2ANALYSE_E-d4974bba7e

Contents

1. General

1.1. Welcome to go2ANALYSE

The Bitstream Processor (go2ANALYSE) is powerful software for offline analysis of bitstreams, e.g. the analysis of decoder characteristics. It can either be used as an offline tool or in combination with COMINT software suite such as go2DECODE. Easy-to-use functions allow fast and partially automatic processing of bitstream inputs. It can display the bitstream in several modes, the bitstream can be examined applying functions such as pattern search, tagging of relevant patterns or descrambling. Statistical analysis functions are available as well as autocorrelation functions, fast pattern searching, application of different code tables and the search for periodic and non-periodic sequences. User defined analysis functions can be added by use of the decoder description language DDL.

1.2. About this manual

This user manual provides basic information about the operation and the application of go2ANALYSE. The first chapter gives an overview of the functional contexts. Installation instructions are included in the second chapter. The third chapter gives an overview of operating the go2ANALYSE software. The details about the functions and displays are explained in the subsequent chapters. For more information please contact service@procitec.com.

1.3. The go2signals product-line

The use of radio communication is constantly rising. The traditional approach of monitoring these more and more connected signal scenarios with a manual approach of channel stepping and manual search is not promising for future challenges.

The product line go2signals covers customer requirements from traditional manual signal handling up to fully automatic intelligence system. This provides processing speed and user comfort of automatic intelligence systems even from single user working positions. It is the perfect solution for mobile, stand-alone and remote controlled applications as well as a start into the world of automatic monitoring.

SUITES	
MONITORING SUITE	Bundles all products around monitoring and signal production
ANALYSIS SUITE	Bundles all products around signal analysis, signal simulation and decoder development
COMPONENTS SUITE	Bundles all products around integrable system components

Table 1: go2signals Suites

PRODUCTS	
GO2MONITOR	Radio monitoring, signal classification, signal decoding and signal recording software solution for complete signal scenario surveillance (HF, VHF, UHF, SAT bands)
GO2MONITOR LOWSWAP	Radio monitoring, signal classification, signal decoding and signal recording software solution for complete signal scenario surveillance (HF, VHF, UHF, SAT bands) optimized for low-SWaP equipment
GO2MONITOR OPERATOR	Application for setting up additional operator stations for manual signal processing and result viewing for an existing central go2MONITOR system
GO2MONITOR RESULT	Application to set up an additional workplace for go2monitor result processing
SCL	Software integrable automatic signal modulation and modem classification C++ library
GO2DECODE	Analysis, demodulation and decoding of known and unknown radio signals. Creating and editing of customer defined modems and decoders
GO2ANALYSE	Analysis, evaluation and manipulation of bit streams for the determination of coding characteristics

Table 2: go2signals Products

2. Installation

2.1. System Requirements

The following operating systems are supported (64bit only):

Windows®

- Windows® 10 (de/en)
- Windows® 11 (de/en)

PC or notebook with a minimum of

- one hard disk
- one DVD-ROM drive (for installation only)
- one free USB port (dongle version only).

2.2. Installation Instructions

An installation wizard guides you through the setup step by step through the entire installation.

go2ANALYSE is not compatible with an ampersand character (“&”) in the username. Please choose a different username.

Make sure no dongle is connected to the USB port of your computer.

If the software was delivered on DVD, insert the go2ANALYSE DVD into the DVD-ROM drive. If the software was downloaded, unpack the delivered ZIP-archive.

2.3. Copy Protection via CodeMeter®

An application protected by CodeMeter® can only run if the CodeMeter® is connected and its driver is installed. The CodeMeter® may be shipped with the software or can already be at hand. An encrypted license file (.maw) is needed. It contains information about the CodeMeter® and the unlocked features depending on the licensed configuration of the software.

If you desire to use a CodeMeter® already at hand, please contact our support at service@procitec.com.

2.4. Installing the Software

Make sure **no** dongle is yet connected to the USB port of your computer.

Insert the go2ANALYSE DVD into the DVD-ROM drive. The setup will start automatically; otherwise start the file “setup.exe” from the DVD. Follow the instructions on the screen.

2.5. Connecting the Dongle

2.5.1. Local CodeMeter®-Connecting

Connect the CodeMeter® to an available USB Port of the computer. The CodeMeter® must remain connected to the local USB port while using the software.

Note: If the CodeMeter® was previously connected, diconnect it. Restart and reconnect the CodeMeter®.

The installation is now complete.

2.5.2. CodeMeter®-Connecting via Network

Note: The connection of a CodeMeter® on a network is described below for the Windows® operating system.

Depending on the configuration of the software, copy protection can also be provided by another computer or server on the network. Therefore, a CodeMeter® containing multiple licenses has to be connected to this "copy protection server". These licenses can then be provided via network to the go2ANALYSE installations on client computers.

To install a copy protection server, follow the steps below.

1. Uninstall all CodeMeter® components
2. Install the CodeMeter® runtime from DVD (applies only to the server)
3. Connect the CodeMeter®
4. Start the server as described below

2.5.2.1. Open Server

The CodeMeter® control center shown in Figure 1 is opened by selecting <**Start Menu**><**All Programs**><**CodeMeter**><**CodeMeter Control Center**>.

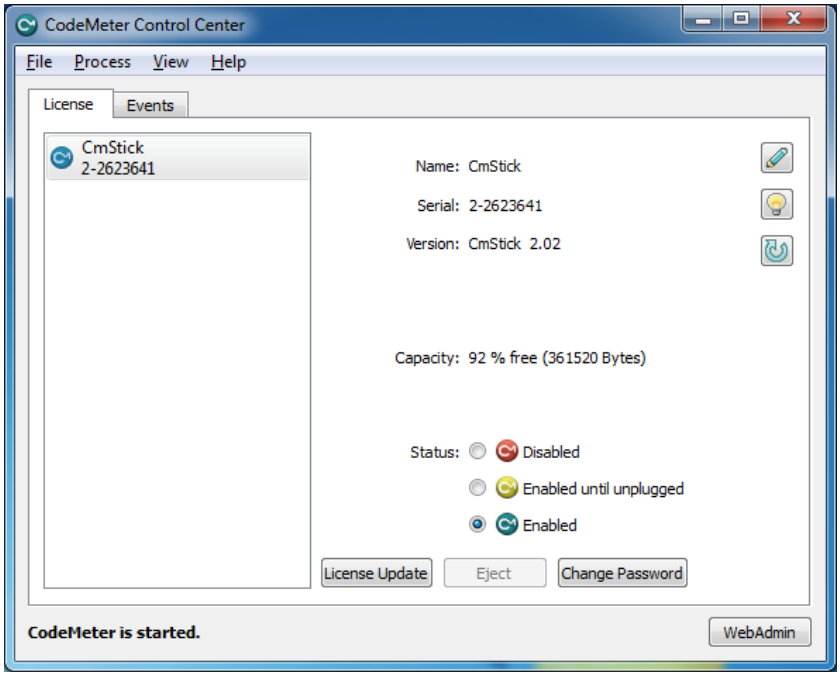


Figure 1: CodeMeter® Control Center

To call the CodeMeter® WebAdmin module, click <**WebAdmin**> in the CodeMeter® Control Center.

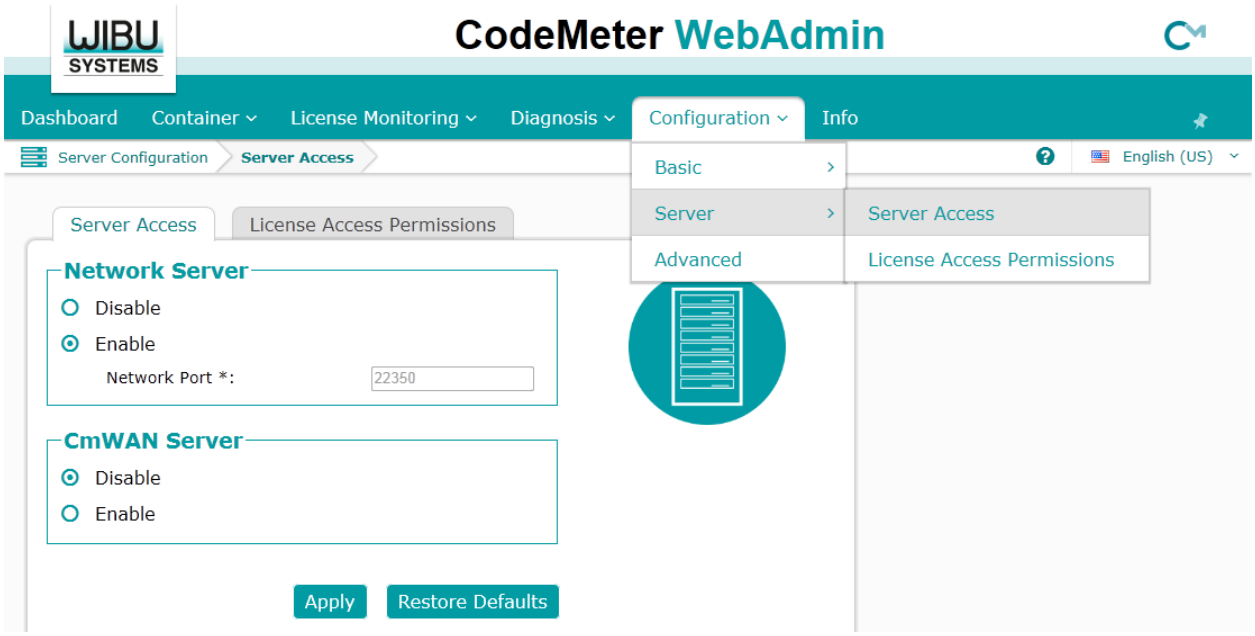


Figure 2: Setup CodeMeter® Server

In the Network Server section, select <**Enable**> and click <**Apply**>.

Note: The server service can also be disabled (stopped) here.

In the CodeMeter® Control Center (see Figure 1), select <**Process**><**Restart CodeMeter Service**>.

The WebAdmin also provides information about connected CodeMeter® such as the quantity of used and available licenses.

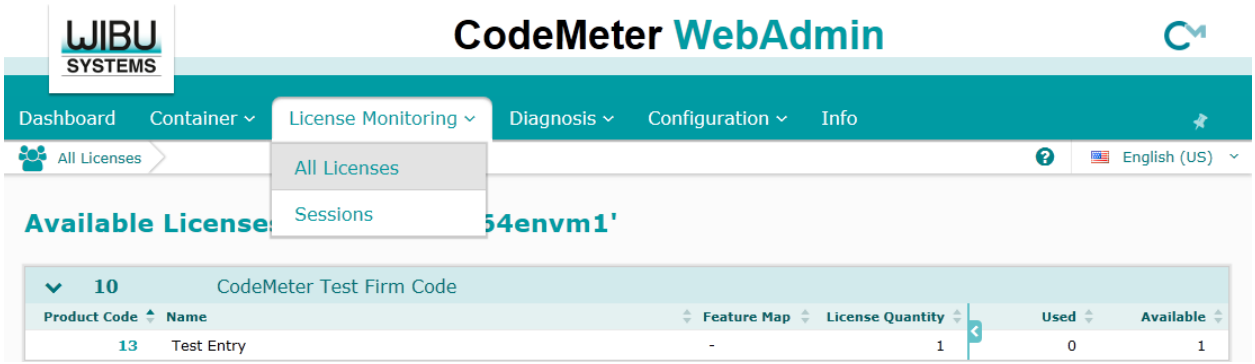


Figure 3: License Information

The CodeMeter® should now also be accessible by the go2ANALYSE installation on the client computer. If not, you can add the server name or its IP address to the Server Search List on the client via the WebAdmin interface.

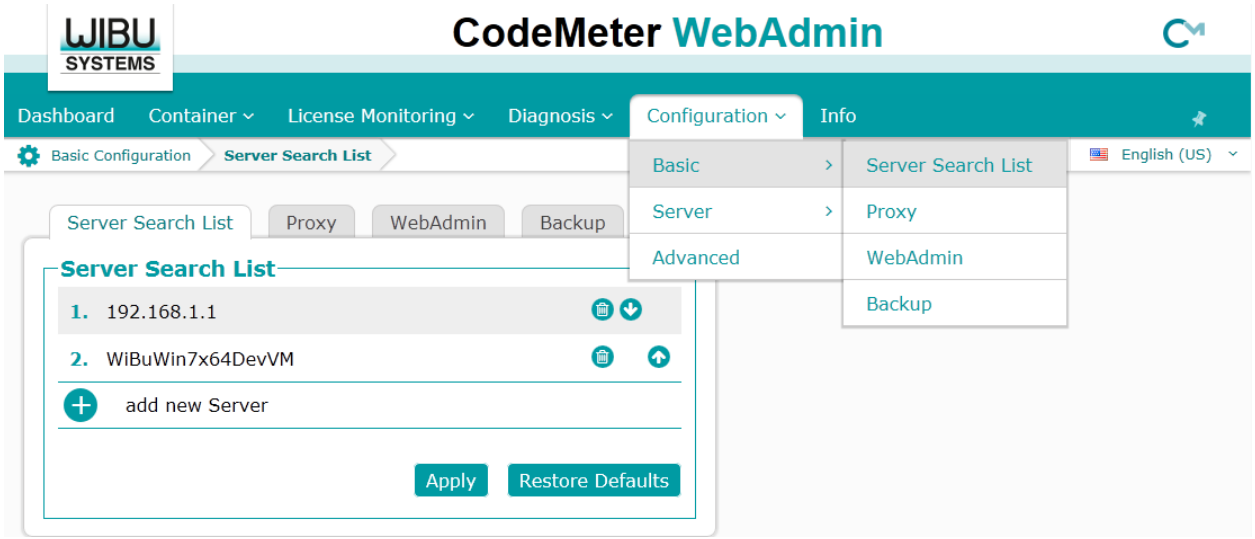


Figure 4: Server Search List

2.5.2.1.1. Stop Network Server

The network server is terminated via the CodeMeter® WebAdmin module (see Figure 2). Under the Server Access tab, select <**Disable**> and confirm by clicking the <**Apply**> button.

2.5.2.2. Troubleshooting

Should the remote client not be able to connect to the server, the firewall should be configured to allow communication on port 22350.

Another possibility to establish a connection is to stop and then restart the server.

2.5.2.3. Connected Clients

As soon as the CodeMeter® runtime software is installed on a connected network client, it will be possible to access its WebAdmin module via:

http://<ClientNameOrIPAddress>:22350/index.html

2.6. Starting the Software

When you start the software for the first time, you will be asked to point to the location of the provided license file. The file will be automatically copied to the appropriate subfolder in your user folder.

1. Start the application
2. Click <**Yes**>
3. In the file dialog that opens, select the path to the MAW file that is included to activate the software and select the correct file
4. The file will be copied to your user folder and renamed to "default.maw"

Note: When you launch the application for the first time, you may receive a warning from the Windows® Firewall. This is because individual applications communicate with each other on localhost via TCP/IP. This communication can be approved without any security risk.

2.7. License Renewal

If you have a new license file - for a new software version or with extended options - you have to delete the old license file from the appropriate user folder:

- Windows®

  %USERPROFILE%\go2SIGNALS\go2ANALYSE x.y\default.maw

Note: "x.y" denotes the old version, e.g. v19.1

After you restart the software, you will be asked to point to the location of the provided license file. The file will be automatically copied to the appropriate subfolder in your user folder.

2.8. Update From Older Versions

go2ANALYSE stores all user modified data in the respective user folder on

- Windows®

  %USERPROFILE%\go2SIGNALS\go2ANALYSE x.y

Note: x.y denotes the major (x) and minor (y) version, e.g. v19.1 for go2ANALYSE v19.1.

These files will remain when you uninstall the software.

2.8.1. Update

If there is only a change in the revision number (last number of the version, e.g. from v19.1.1 to v19.1.2) go2ANALYSE will immediately use your existing user data.

However, if you prefer a clean installation you can either delete or rename the folder go2ANALYSE x.y on

- Windows®

```
%USERPROFILE%\go2SIGNALS
```

before you launch go2ANALYSE.

2.8.2. Upgrade

In case of the installation of a new version an existing older version will be automatically uninstalled.

2.9. Uninstallation

On Windows®, select in the <**Control Panel**> <**Programs and Features**> go2ANALYSE and then click <**Uninstall**>.

3. Basics

3.1. Software Start

To start go2ANALYSE click the desktop icon or via the operating system's taskbar.

The graphical user interface of the go2ANALYSE is shown without any display window when go2ANALYSE is started for the first time after installation.

3.2. Overview

Figure 5 below gives an overview of the go2ANALYSE interface. It shows all displays which are available in go2ANALYSE.

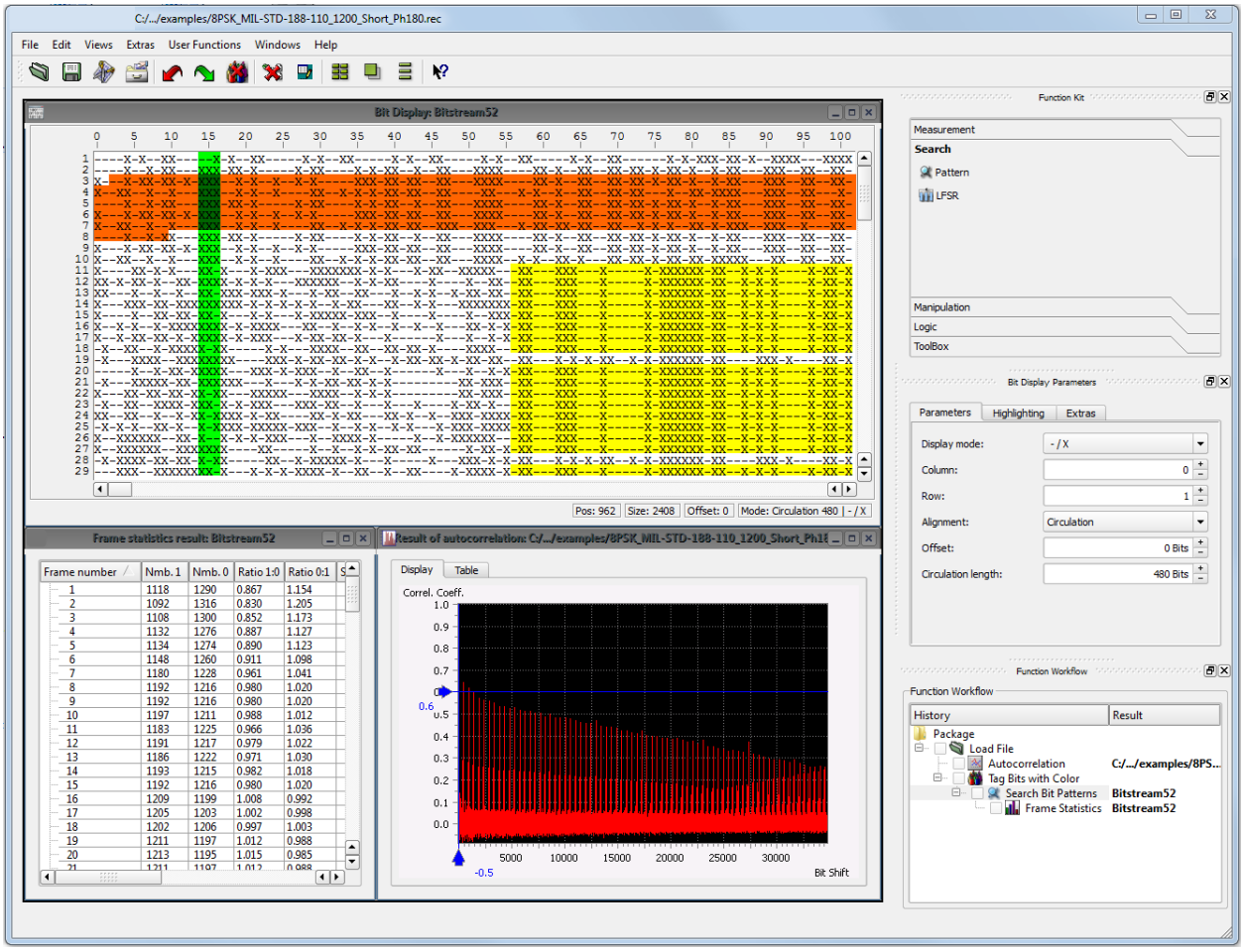


Figure 5: go2ANALYSE Interface

go2ANALYSE is divided into a main window with the menu bar, the toolbar and the docking area where the <**Function Kit**> resides, and a workspace where the various displays are inserted.

In Figure 5 above nearly all displays available are open.

- The Bit Display is the main display of the go2ANALYSE. It shows the current bitstream, is used for navigating in the bitstream and allows for changing of the bitstream visualization (see Bit Display on page 33).
- The Measurement Display shows the result of measurement functions which have been applied to the bitstream, it allows for verifying the results and provides cursors (see Measurement Features on page 41).
- The Result Display shows the result of statistical functions and of the LFSR (linear feedback shift registers) sequence search, it lists the sorted results (see Result Display on page 55).
- Additional Text Displays (not shown here) may appear showing results of the User Functions (see User Functions on page 75).

3.3. go2ANALYSE Components

3.3.1. Main Window

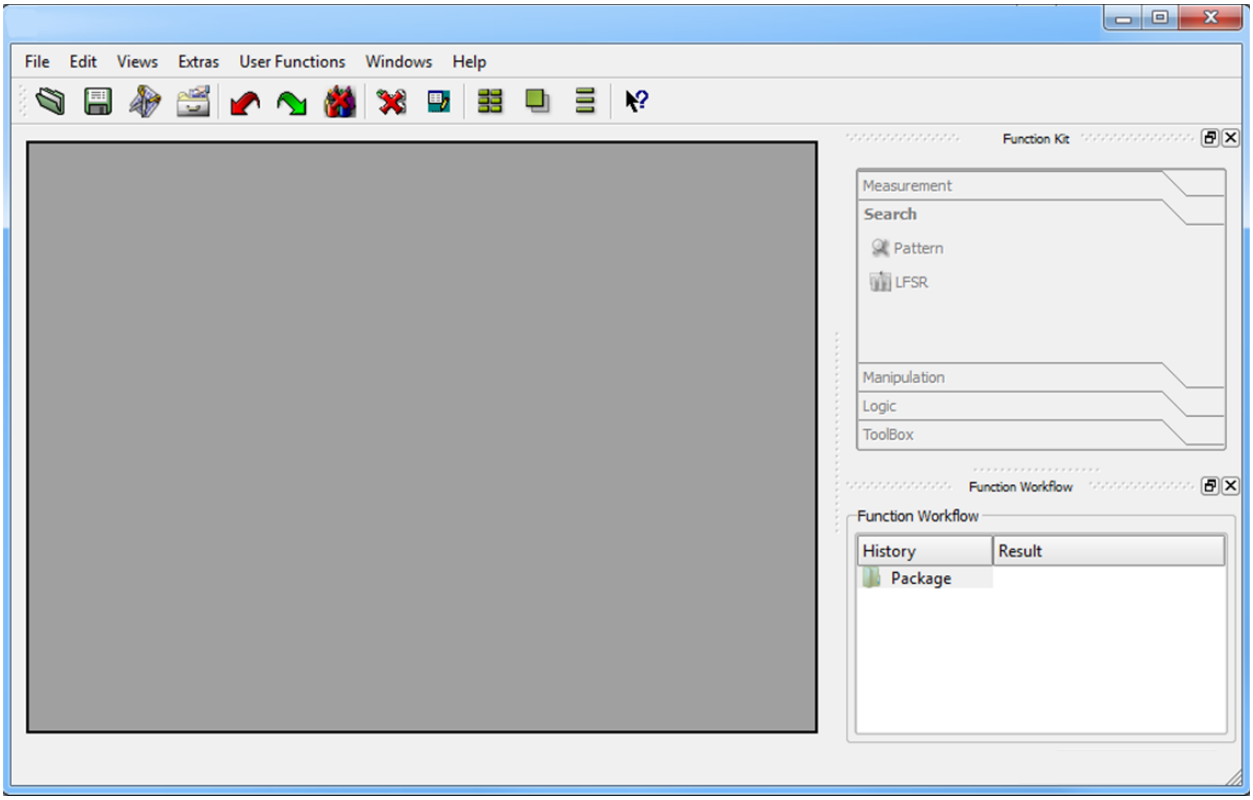


Figure 6: go2ANALYSE Main Window

The main window consists of the menu bar and the toolbar, the workspace and the <**Function Kit**>. It provides docking areas to dock windows such as the property sheets of the various displays, the <**Function Kit**>, the dialog box of the go2ANALYSE function and the toolbar.

The buttons of the <**Function Kit**> are disabled if no bitstream is loaded. They are enabled when opening a bitstream. They are disabled again when the current bitstream is closed.

3.3.2. Menu Bar

The Menu Bar provides quick access to important functions.

3.3.2.1. File Menu

Selecting <**File**> the menu in Figure 7 will be displayed. It contains all actions with regard to files.

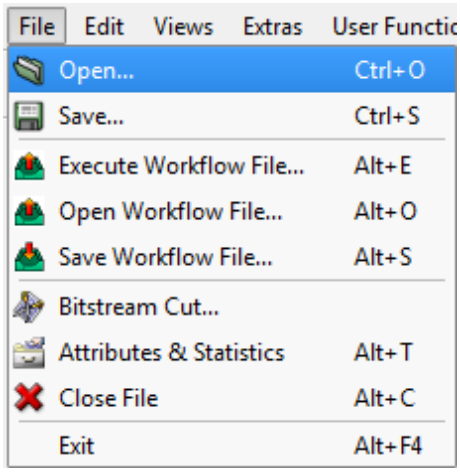


Figure 7: Menu File

Menu Item	Icon	Shortcut	Description
<**Open...**>		<**Ctrl**>+<**O**>	Open bitstream in text format
<**Save...**>		<**Ctrl**>+<**S**>	Save current state of bitstream
<**Execute Workflow File...**>		<**Alt**>+<**E**>	Execute workflow
<**Open Workflow File...**>		<**Alt**>+<**O**>	Open workflow
<**Save Workflow File...**>		<**Alt**>+<**S**>	Save current workflow
<**Bitstream Cut...**>			Cut oversized "*.rec" file into partial bitstream
<**Attributes & Statistics**>		<**Alt**>+<**T**>	Open file characteristics and statistics dialog
<**Close File**>		<**Alt**>+<**C**>	Close current bitstream and all open displays

Table 3: File Menu - Functions

3.3.2.2. Edit Menu

Selecting <**Edit**> the menu in Figure 8 will be displayed. This menu contains undo/redo actions and a list of recent actions.

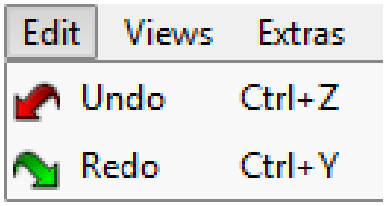


Figure 8: Menu Edit

Menu Item	Icon	Shortcut	Description
<**Undo**>		<**Ctrl**>+<**Z**>	Undo last action affecting the bitstream
<**Redo**>		<**Ctrl**>+<**Y**>	Redo last action undone

Table 4: Edit Menu - Functions

3.3.2.3. Views Menu

Selecting <**Views**> the menu in Figure 9 will be displayed. This menu contains actions to open new displays.

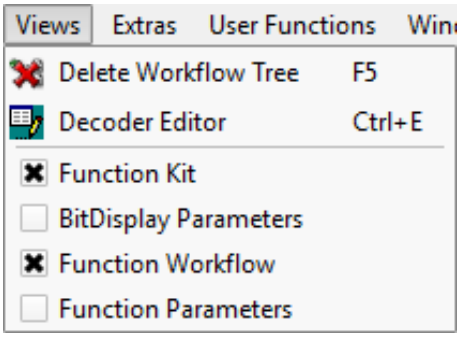


Figure 9: Menu Views

Menu Item	Icon	Shortcut	Description
<**Delete Workflow Tree**>		<**F5**>	Delete current workflow Tree
<**Decoder Editor**>		<**Ctrl**>+<**E**>	Open the decoder editor

Table 5: Views Menu - Functions

3.3.2.4. Extras Menu

Selecting <**Extras**> the menu in Figure 10 will be displayed.

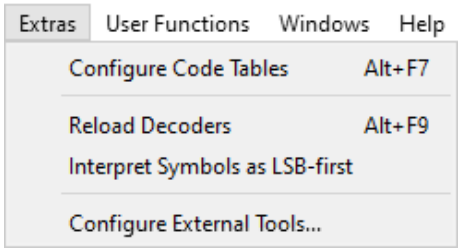


Figure 10: Menu Extras

Menu Item	Shortcut	Description
<**Configure Code Tables**>	<**Alt**>+<**F7**>	Open the <**Configure Code Table**> dialog box to manage code tables
<**Reload Decoders**>	<**Alt**>+<**F9**>	Reload decoders in <**User Functions**>
<**Interpret Symbols as LSB-first**>		Set the symbol bit order to least significant bit (LSB) first. The default is most significant bit (MSB) first. Enabling this mode changes the conversion between symbols and bits for record-based bitstreams. For example, consider a record-based bitstream with 3 bits per symbol containing the symbol numbers 4, 3, 2 and 1. By default, these are converted using MSB-first, resulting in 100 011 010 001. With this mode enabled LSB-first is used, resulting in 001 110 010 100 - the bit order within each symbol is reversed. This is reflected in the Bit Display. Note, this does not affect how <**User Functions**> process record-based bitstreams. Special care is required when setting parameters and interpreting the results.
<**Configure External Tools...**>		

Table 6: Extras Menu - Functions

3.3.2.5. User Functions Menu

Selecting <**User Functions**> the menu in Figure 11 will be displayed.

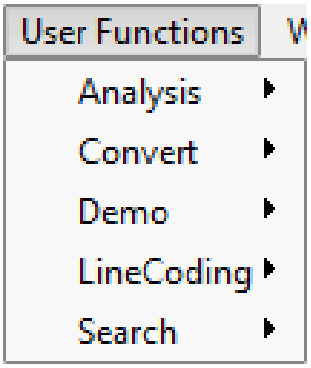


Figure 11: Menu User Functions

Menu Item	Description
<**Analysis**>	General functions to analyze the bit stream
<**Convert**>	General functions to modify the bit stream
<**LineCoding**>	Function to recover line cod-ings
<**Search**>	Function to search for specific bit stream characteristics

Table 7: User Functions Menu - Functions

3.3.2.6. Windows Menu

Selecting <**Windows**> the menu in Figure 12 will be displayed. This menu contains actions to manage open windows.

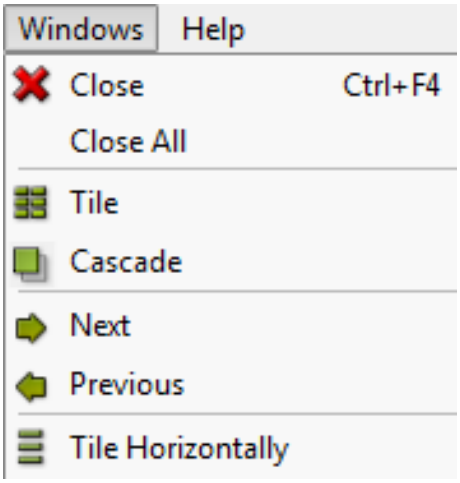


Figure 12: Menu Windows

Menu Item	Icon	Shortcut	Description
<**Close**>		<**Ctrl**>+<**F4**>	Close active window
<**Close All**>			Close all open displays
<**Tile**>			Arrange all open windows side by side
<**Cascade**>			Cascade all open windows
<**Next**>			Next window
<**Previous**>			Previous window
<**Tile Horizontally**>			Arrange all open windows one below the other. All windows have the same height.

Table 8: Windows Menu - Functions

3.3.2.7. Help Menu

Selecting <**Help**> the menu in Figure 13 will be displayed.

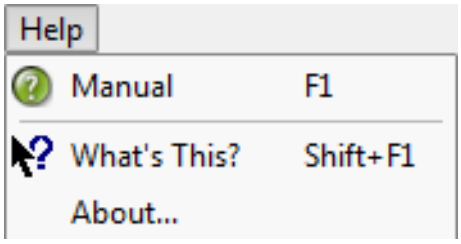


Figure 13: Menu Help

Menu Item	Icon	Shortcut	Description
<**Manual**>		<**F1**>	Opens the present Operating Manual by means of Adobe® Reader. You will be able to search any help information using the Adobe® Reader's built-in search function.
<**What's This**>		<**Shift**>+<**F1**>	The cursor changes into the "What's This" cursor. The next click on a GUI element will supply online help, if available
<**About...**>			Show the current version number

Table 9: Help Menu - Functions

3.3.3. Toolbar

The toolbar displays the major functions of the Menu Bar in icons. These icons are activated by clicking the respective icon with the left mouse button.

Figure 14: Toolbar

Icon	Function
	Load bitstream
	Save bitstream
	Call bitstream cut function
	Open attributes and statistics dialog
	Undo last action affecting the bitstream

Icon	Function
	Redo last action made to the bitstream
	Clear tags of chosen color
	Close current file and all windows
	Open the decoder editor
	Arrange all open windows vertically
	Cascade all open windows
	Arrange all open windows horizontally
	What's This? help

Table 10: Toolbar Icons

3.3.4. Function Kit

The <**Function Kit**> provides quick access to the go2ANALYSE functions.

Function Category	Function	Icon	Description
<**Measurement**>			
	<**Correlation**>		Analyze bitstream by means of autocorrelation
	<**Bit Length Analysis**>		Analyze length of the runs of ones and zeros in the file
	<**Frame Statistics**>		Analyze ratio of zeros in ones of frames with user-defined length
	<**Parity & Weight**>		Analyze parity and weight of frames with user-defined length
<**Search**>			This category contains search functions and functions for verification
	<**Pattern**>		Search for specific bit sequence in the stream
	<**Search LFSR**>		Search for bit sequence generated by an LFSR with specified maximum linear complexity
<**Manipulation**>			
	<**Delete**>		Delete sequence of bits from stream

Function Category	Function	Icon	Description
	<Tag Bits>		Tag bit sequence with a specific color
	<Reverse Bits>		Reverse bit sequence at specified position
	<Clear Tags>		Clear all tags or only tags with a specific color
<Logic>			
	<AND>		Logical AND of a specified bit sequence with the bitstream
	<OR>		Logical OR of a specified bit sequence with the bitstream
	<NOT>		Invert bitstream at a specified position
	<XOR>		Logical XOR of a specified bit sequence with the bitstream
	<XOR Bitstreams>		XOR two Bitstreams
<Toolbox>			This category contains all functions not belonging to other categories
	<Diff Bitstream>		XOR two Bitstreams
	<Map Bits to Text>		Map bits manually to text

Table 11: Function Kit Categories

3.4. Help Features

For quick help the go2ANALYSE shows a tooltip to any button or text/spin box if the mouse cursor resides longer than approximately 3 seconds on this icon or box.

Figure 15 below shows the example of the tooltip What's This?.

Figure 15: Toolbar with Tooltip for Bit Display Icon

All major features of the go2ANALYSE interface also have a "What's This?" help. It is displayed on clicking the toolbar icon or after pressing <**Shift**>+<**F1**>.

3.5. First Steps

3.5.1. Open Bitstream

Open the file to be analyzed by pressing the file open icon  on the toolbar or choose <**File**><**Open...**>. You may also open one of the previously processed files by selecting one of the file names listed below the <**File**><**Exit**> item as shown in Figure 16.

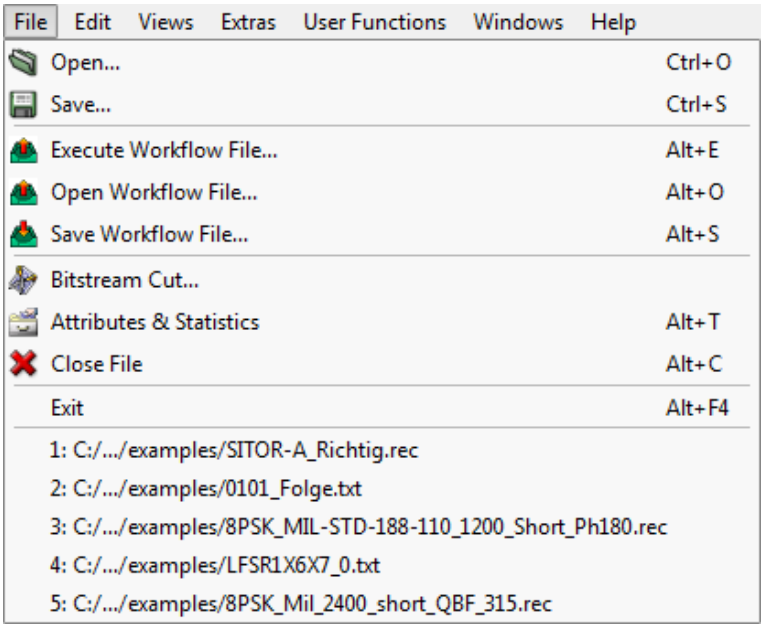


Figure 16: File Menu

The standard file dialog of the operating system is opened and you are prompted to choose a directory and a file.

There are three different file types for bitstreams to choose from: "*.txt"-based bitstream and "*.rec" based bitstream, all other file types will be read as pure binary bit streams.

3.5.1.1. Text-based Bitstreams: "*.txt"

A "*.txt" bitstream only contains the information about the bits in the bitstream and no extra information can be viewed when such a stream is loaded. go2ANALYSE converts the characters of a "*.txt" stream using Table 12- any other characters will be ignored!

Character	Converted Bit
0	0
-(Dash)	0
_(Underline)	0
L	0
. (Dot)	0
1	1

Character	Converted Bit
X	1
x	1
H	1

Table 12: Text Based Bitstream Conversion

3.5.1.2. Record-Based Bitstreams: "*.rec"

go2ANALYSE, go2DECODE and APC/SDA are able to record demodulator outputs containing the full set of information, which is used be succeeding decoders. Besides the bitstream itself this is symbol qualities, symbol times, symbol bounds, burst bounds etc. Some go2ANALYSE functions rely on the availability of these additional information.

3.5.1.3. Binary Bitstreams: "*.*"

Any other file, i.e. a file with neither "*.txt" nor "*.rec" extension, will be read as a pure binary bitstream in bytewise order.

3.5.2. Cut Bitstream (Partial Bitstream)

With go2ANALYSE large "*.rec" files can be resized. To do so, open the Partial Bitstream dialog box by selecting <**File**><**Bitstream Cut...**>. In the Open File dialog box displayed, choose the desired "*.rec" file and press the button <**Open**>. go2ANALYSE shows the following dialog box.

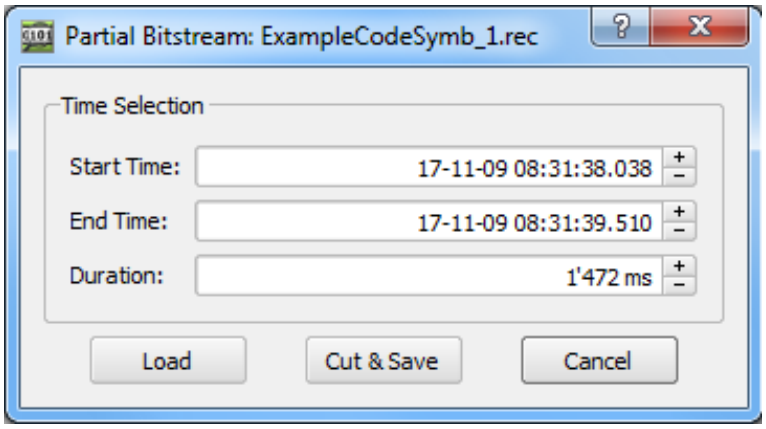


Figure 17: Partial Bitstream Dialog Box

Edit the spin boxes for <**Start Time**>, <**End Time**> or <**Duration**>, if desired, and press the button <**Load**>. go2ANALYSE shows the bitstream section in <**Bit Display**> in accordance with the time stamp selected between start and end time. Save the partial bitstream by use of the button <**Cut & Save**> and the <**Save File**> dialog box displayed.

Note: Repeat loading the bitstream until the desired partial bitstream has been selected.

3.5.3. Navigate within Bitstream

To navigate within the bitstream, move the scrollbars of <**Bit Display**> or edit the parameters in <**Parameters**><**Column**> and <**Parameters**><**Row**>.

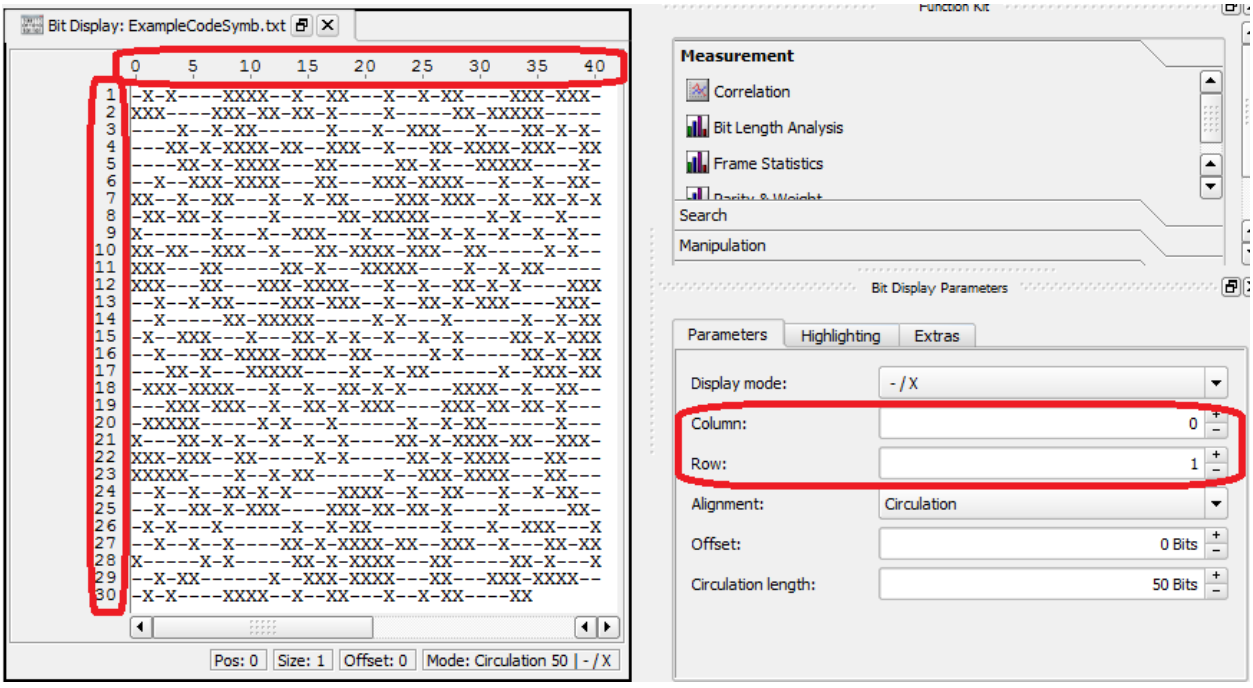


Figure 18: Bit Display Parameters - Column and Row

If no scrollbars are visible, the display shows the entire bitstream.

Changing the parameter <**Circulation length**> (wrap length) will change the alignment of the bits, thus patterns occurring periodically will be visible.

Example

The first bitstream is displayed with a circulation length of 51 and no structure or periods are visible:

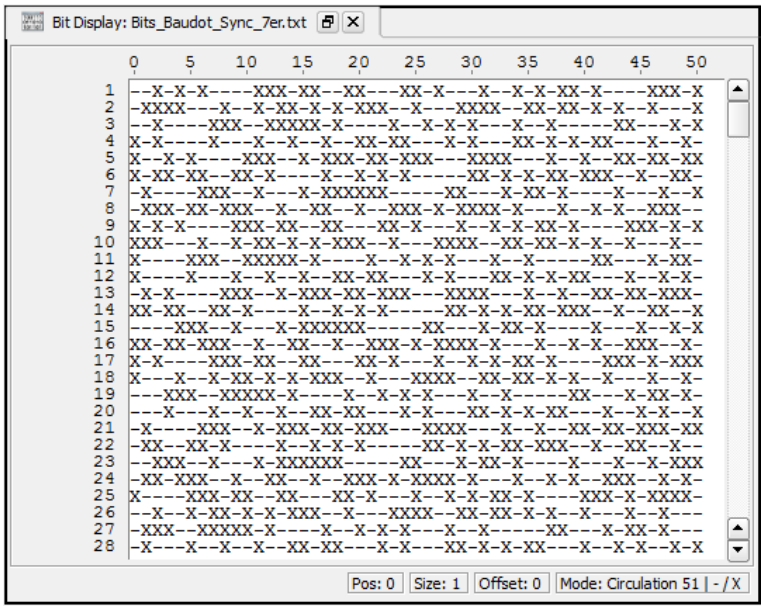


Figure 19: Bitstream with Circulation Length of 51

After changing the circulation length to 49, the same stream looks as follows:

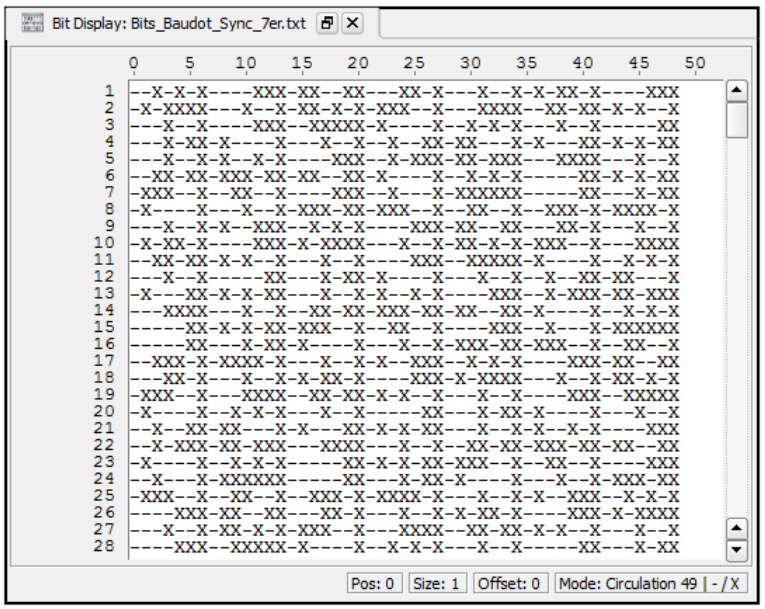


Figure 20: Same Stream as above with Circulation Length now 49

3.5.4. Basic Editing Features

go2ANALYSE provides editing features for bitstreams known from typical text editors. These features include <**Copy**>, <**Replace**> and <**Insert**> of bits as shown in Figure 21.

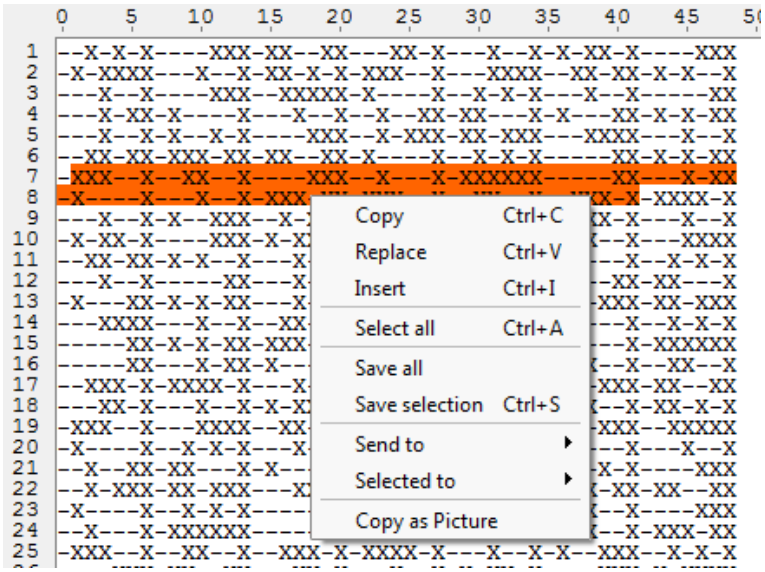


Figure 21: Bit Display - Basic Editing Features

To copy some bits to the clipboard highlight them with the mouse, click the right mouse button and select <**Copy**> from the popup menu.

To insert the bits into the stream, select (highlight) the position at which the bits are to be inserted, click the right mouse button and choose <**Insert**>. The bits are written into the stream right before the highlighted/selected bits, the selected bits are not replaced by the insertion.

Note: <**Bit Display**> has no cursor as a usual text editor. To select the position at which to insert the bits you have to make a selection of at least one bit with the mouse.

If you choose <**Replace**> instead of <**Insert**>, the selected bits of the bitstream are replaced with the bits from the clipboard.

go2ANALYSE also provides editing functions which are uncommon to text editors but basic features for a bit editor such as inversion or tagging of interesting bit sequences.

To invert the bitstream (inversion means changing the value of each bit from 0 to 1 and from 1 to 0) click the <**Function Kit**><**Logic**><**Not**> button  . Then Figure 22 is shown.

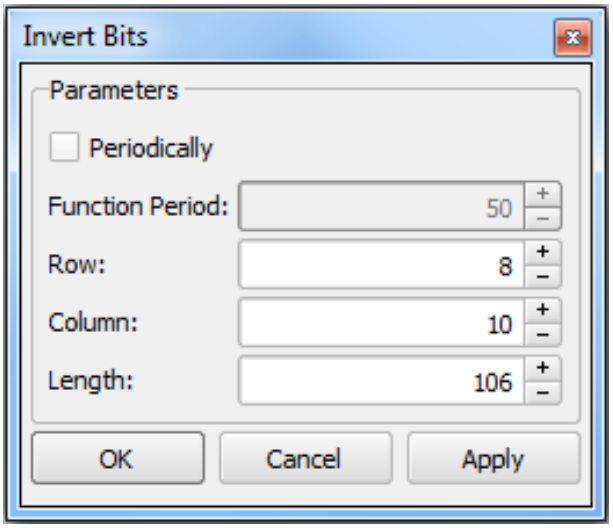


Figure 22: Exemplary Dialog Box

If the <**Periodically**> checkbox is activated and confirmed with <**OK**> the entire bitstream is inverted.

Note: When pressing <**Apply**> instead of <**OK**>, the current dialog box will remain visible. This will facilitate the repeating of functions.

<**Tag Bits**> serves to highlight interesting bit sequences in a chosen color. The tag remains on the bit until it is cleared by the Clear Tags function, replaced by another tag or until the bitstream is closed. The Reverse Bits function swaps the bits defined by the parameter of the function.

Note: For details see chapters Tag Bits, Reverse Bits and Clear Tags.

3.5.5. Define Bits for Application of go2ANALYSE Functions

Most of the built-in functions of go2ANALYSE cannot be applied until you click the adequate button in the <**Function Kit**> Menu. The click always causes the dialog box to be displayed (see previous chapter).

This dialog displays the current function parameters before the function is applied to the stream. The function parameters can be changed to define exactly what the function should do and, even more important, to determine exactly which bits the function is to be applied to.

How do these parameters determine which bits the function is applied to?

Example

Assume the following bitstream with a circulation length of 8:

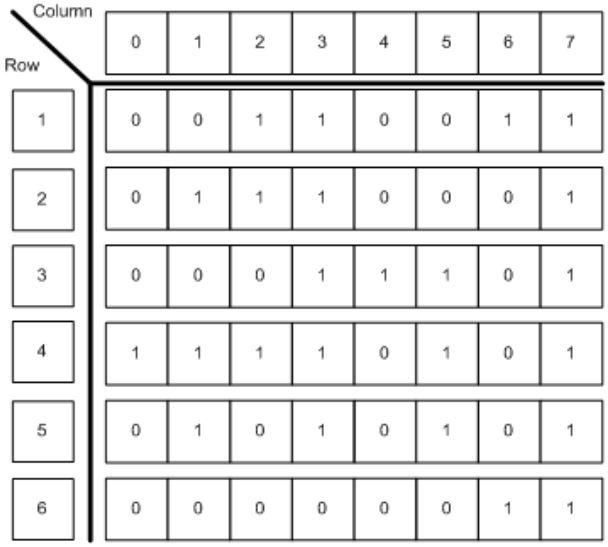


Figure 23: Exemplary Bitstream

The rows are numbered from 1 to 6, the columns from 0 to 7.

To apply the function to the bits in row 1 in columns 2 to 4 (bit 4 should be included) enter the following parameters:

Row 1, Column 2, Length 3

To apply the function to the bits in rows 1 to 6 and columns 2 to 4 activate <**Periodically**> checkbox and set <**Function Period**> to 8.

To apply the function to the bits on squared background in Figure 24 enter the following parameters:

Row 1, Column 0, Length 1 and Function Period 9. <**Periodically**> checkbox is activated.

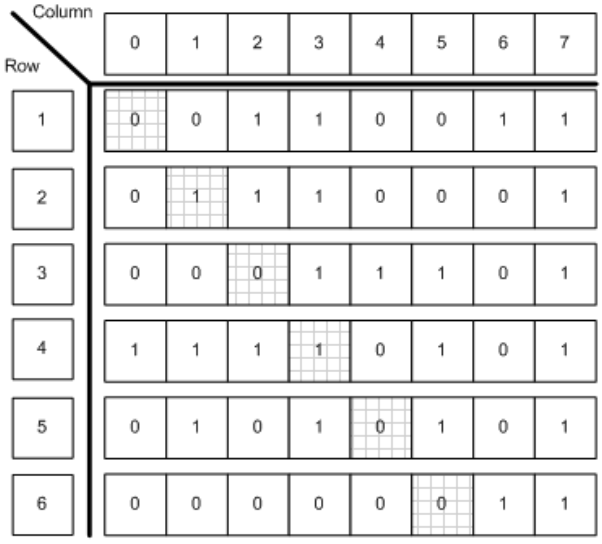


Figure 24: Example of Periodic Function Mode

In the following, a function has been defined to work in periodic mode only if it is executed periodically on a stream.

Note: Some functions always work on the entire bitstream. In this case the dialog box does not provide the parameters <**Function Period**>, <**Row**>,<**Column**>, <**Length**> and the checkbox <**Periodically**>, or these parameters will be disabled.

Disabled parameters are easy to recognize because they are greyed as shown in Figure 25.

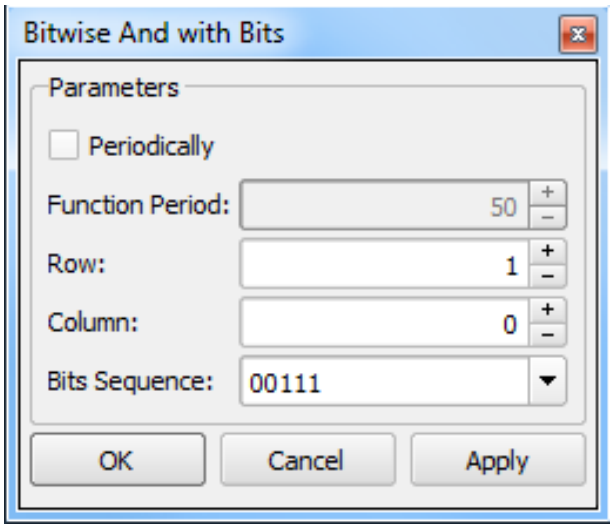


Figure 25: Dialog Box for AND Operation with Disabled Function Period Parameter

By checking an option or changing of another parameter value these function parameters will be enabled.

Example

On activating the <**Periodically**> checkbox in Figure 25 the parameter <**Function Period**> will be enabled. By unchecking this option, the parameter will be disabled again.

Attention: It is possible to select a function period which is lower than length. If you choose a function period lower than length, be aware that the function is applied to the same bits twice or several times!

3.5.6. Undo Changes to the Bitstream

To undo your last action which affected the bitstream either choose the item <**Edit**><**Undo**> or click . go2ANALYSE recovers the bitstream to the state prior to the last action. It is only possible to undo the last ten actions affecting the bitstream.

3.5.7. Redo Changes Undone

To redo the last action undone either choose the item <**Edit**><**Redo**> or click . go2ANALYSE recovers the bitstream to the state prior to undoing the last action. It is not possible to redo more than the undone steps.

3.5.8. Adjust Bitstream View

go2ANALYSE provides various visualization modes for the bitstream. The bitstream view can be changed by changing <**Bit Display Parameters**><**Parameters**><**Display mode**> as shown in Figure 26 and Figure 27.

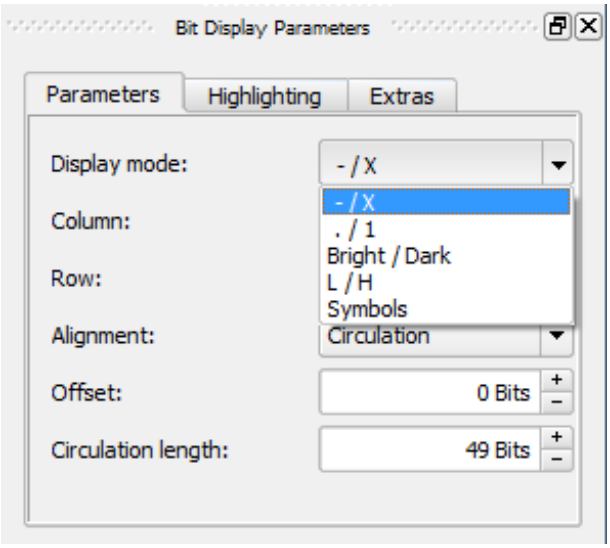


Figure 26: Bit Display - different Display modes

Figure 27: Bitstream View Display Modes

Figure 27 shows the several Display modes of <**Bit Display**>. From Left to Right: -/X; ./1; Bright / Dark; L/H and Symbol Numbers.

If Display mode is set to symbol numbers <**Bit Display**> shows the values of the symbols of the bitstream. Any "*.txt"-based bitstream will only show the values 0 and 1 if <**Display mode**> is set to symbol numbers. Only "*.rec"-based bitstreams will show greater symbol values because the "*.rec" files also store information on whether the stream was demodulated with a multi-valued demodulator.

Note: If the display mode chosen is *Bright / Dark* the area of <**Bit Display**> where no bits are displayed is green. If it were displayed in white, it would be impossible to distinguish the area without bits from an area with only zeros.

---

go2ANALYSE also provides the option to change the size in which the bits are displayed. Choose the correct font (bit) size by clicking <**Bit Display Parameters**><**Extras**><**Font size**> and change the font (bit size).

When processing a record file-based bitstream, go2ANALYSE provides more options to adjust the view of the bitstream.

3.5.8.1. View Bursts

Proceed recorded bitstreams ("*.rec-files") allow burst mode signals for viewing each complete burst in one line. To do so, choose the entry *Burst* in the parameter <**Bit Display Parameters**><**Parameters**><**Alignment**>.

Normally <**Alignment**> is set to *Circulation*. In this mode the bits (or characters respectively) shown are aligned vertically in rows. The length of each row is set to the current value of the parameter <**Circulation length**>, this way "frames" become visible if the value of circulation length corresponds to the repetition rate.

If <**Alignment**> is set to *Burst*, each row shows one burst to a maximum length of 10,000 bits.

The length of the rows may differ from each other, as shown in Figure 28.

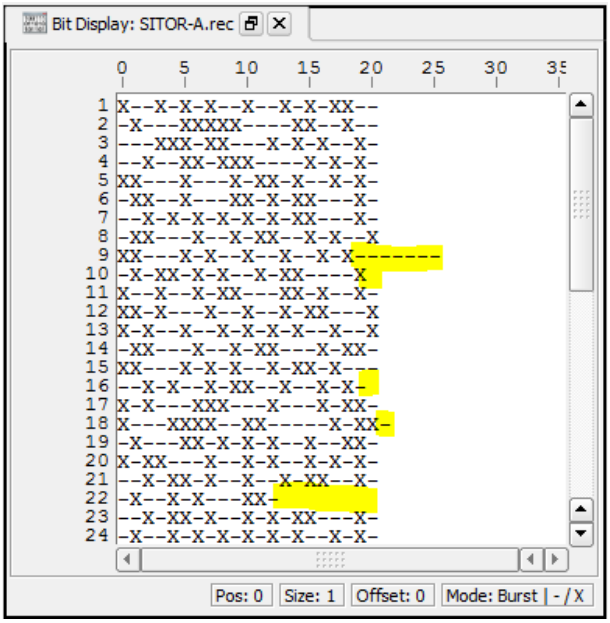


Figure 28: Bitstream with Different Burst Lengths shown with Alignment Burst

3.5.8.2. View Quality Information

Record-based bitstreams also allow for viewing quality information of each stream symbol. The demodulator which produced the record file has produced quality information for each stream symbol. This means that if the communication modem uses single valued symbols (in this case a symbol is equal to one bit) for broadcasting, the record file stores quality information for each bit, otherwise if multi-valued symbols are used, the same bit of quality information is paired to several bits.

To do so, choose <**Extras**><**Bit Display Parameters**><**Show quality**>. After this option is selected the background of each bit is drawn in a scale of grey. The lighter the background of the bit, the better is the quality of this bit.

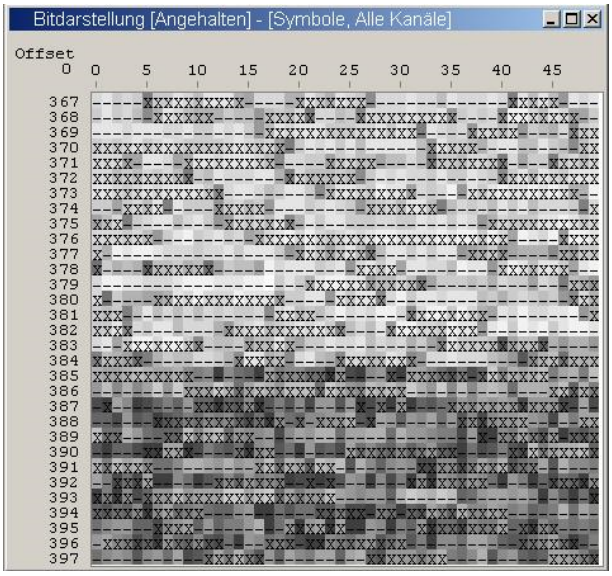


Figure 29: Bitstream with Quality Information

Sixteen different shades of grey are defined for displaying the quality. They range from white to dark grey.

When trying to display a "*.txt"-based bitstream with the <**Show quality**> option all bits are displayed with a white background, because "*.txt"-based bitstreams do not provide any quality information.

3.6. View Bitstream as Text

To view bits as text go2ANALYSE provides the option to map the bits into text using a code table. The output of this mapping function is displayed in the Text Display.

The Text Display opens after activating <**Function Kit**><**Map bits to text**><**Text Display Parameters**>. It shows the complete stream in one line. By default <**Text Display Parameters**> uses the familiar -/X code table. On changing the parameter <**Code Table**> the bits are converted to text using the chosen code table.

Example

<**Bit Display**> shows the stream *ExampleCodeSymb.txt*. After the start of the Text Display, this stream is shown as –/X. Now choose the code table "Baudot". The display will show the following result.

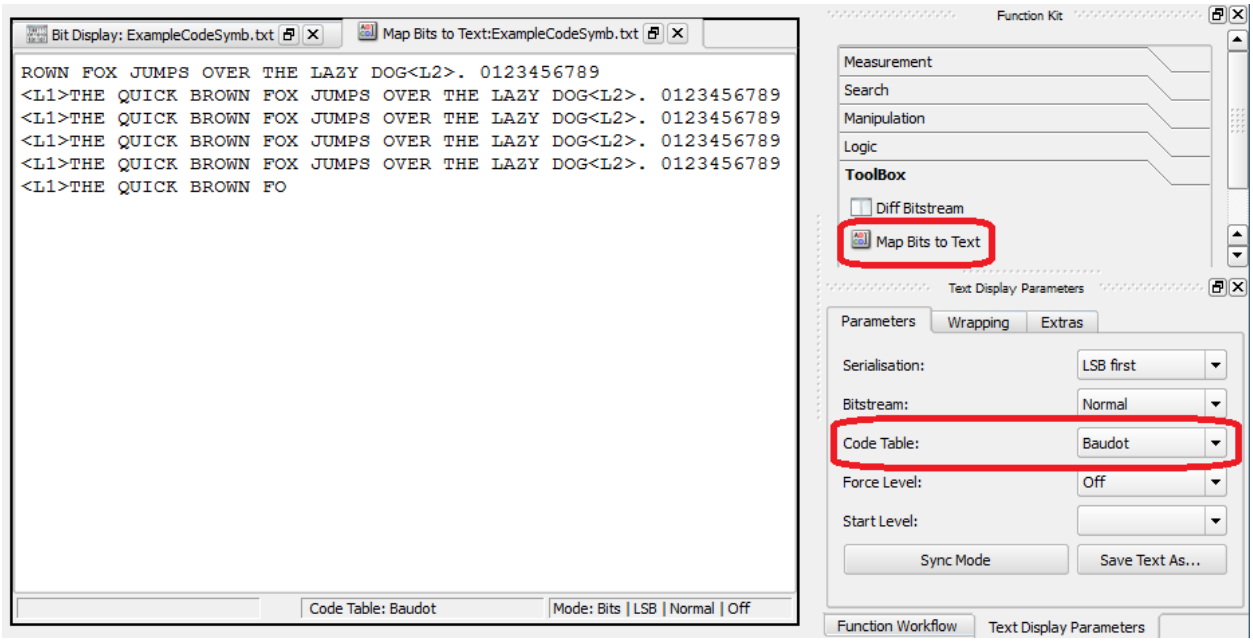


Figure 30: Bitstream ExampleCodeSymb.txt shown in Text Display with Code Table Baudot

For detailed information on using <**Text Display Parameters**> please refer to chapter Text Display on page 37. In order to use other code tables or define your own code tables see chapter Code Tables on page 100 for details.

3.6.1. Customize Workspace

When opening the first bitstream after the installation, the <**Function Kit**> window and the Parameter Display will be docked to the right side of the main window (we say they are docked in the right docking area) and the toolbar will be located below the Menu Bar.

In order to change the appearance of the workspace, go2ANALYSE provides the following customizing options:

- To undock a docked window (i.e. the toolbar, the <**Function Kit**> or the <**Bit Display Parameters**> at first) double click the dock window you wish to undock.
- To re-dock any undocked window simply double click the undocked window, it will be docked in the docking area it was docked in last.
- To dock any undocked window in another docking area (left or right side), move the window to the right or left as far as you can and release the left mouse button.
- To close a docked window, choose the Dockwindows command in <**Views**> and click the entry for the dock window you wish to close.
- If you closed a dock window by opening <**Views**>, choose the dock windows command and select the entry for the dock window you wish to be visible again.
- The toolbar can be undocked as well. It is also possible to close the toolbar or to dock it on the right or left side of the main window.
- The dialog box which is always displayed when pressing a button of the <**Function Kit**> can also be docked to the right or left side. If this dialog is docked it will not be closed on pressing <**OK**> but remains visible.

Note: The docking areas of the main window are located on the right or left side resp. of the main window. Only the toolbar can be docked below the Menu Bar. Displays such as the <**Bit Display**>or the <**Function Kit**><**Measurement**> cannot be docked.

go2ANALYSE will store the current appearance of the workspace at the end of each session.

When starting go2ANALYSE the following settings will be restored:

- The positions of all dock windows except the toolbar
- The list of recent files
- <**Bit Display**> with the file you worked on last (if any)

3.6.2. Save Bitstream

go2ANALYSE provides three options to save changes in the bitstream.

3.6.2.1. Save via File Menu

The first way to save the current bitstream is to select <**File**><**Save...**>, enter a path and a name for the file in the file dialog shown and press the button <**Save**> in the file dialog. The current bitstream is saved to the chosen path with the file name entered. If the file name was changed, the new file name is displayed in the caption of <**Bit Display**> and in the caption of the go2ANALYSE main window.

The file format for bitstreams can be text file (“*.txt”) and decoder input format (“*.rec”) or binary format (any other extension).

The go2ANALYSE provide the possibility for the user to save text file (“*.txt”) in decoder input format (“*.rec”).

When saving text file (“*.txt”) to decoder input format (“*.rec”), a Signal Parameter dialog box opens (see Figure 31).

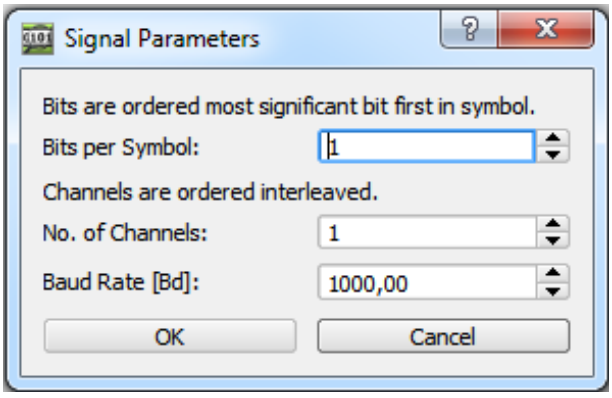


Figure 31: Dialog Box for Signal Parameters

This function allowed adding of Signal Parameters (Bits per Symbol, No. of Channels and Baud Rate) to text file (“*.txt”).

3.6.2.2. Save via Popup Menu of Bit Display

The second way to save the current bitstream is to select <**Save All**> in the popup menu of the <**Bit Display**> (see Figure 32).

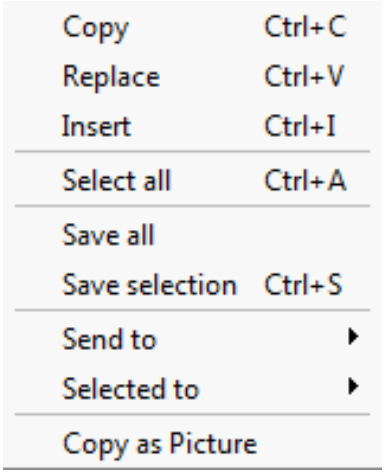


Figure 32: Bit Display - Popup Menu

The popup menu becomes visible on clicking the right mouse button inside the <**Bit Display**>. Having pressed <**Save all**>, enter a path and a name for the file in the file dialog shown and press the button <**Save...**> in the file dialog. The current bitstream is now saved to the chosen path with the file name entered. If the file name was changed, the new file name is displayed in the caption of the <**Bit Display**> and in the caption of the go2ANALYSE main window.

3.6.2.3. Save Section of Bitstream

The third way to save the changes to the current bitstream is to select <**Save all**> or <**Save selection**> from the popup menu of the <**Bit Display**>.

Selecting <**Save all**> will save complete the bitstream to file.

To save only part of the bitstream, proceed as follows:

Select (highlight) the area of bits to save with the mouse (press left button and move the mouse up or down). Having selected all bits of interest release the mouse button and click the right mouse button. Select <**Save selection**> from the popup menu.

Name and type of the destination file can be chosen in the dialog that will appear.

File extension	Data type
"*.txt"	Plain ASCII text
"*.rec"	Record-Based Bitstreams
"*.*"	Binary Bitstreams

Table 13: Bitstream file data types

Note: The captions of the <**Bit Display**> and the main window will not change even if a new file name has been entered because only part of the bitstream was saved. Having saved part of the bitstream, the <**Bit Display**> still shows the original stream.

3.6.3. Attributes & Statistics

For information on the last changes to the current bitstream or the number of ones in this stream, use the <**File**><**Attributes & Statistics**> command (see Figure 16).

On selection of this menu item, go2ANALYSE will show Figure 33.

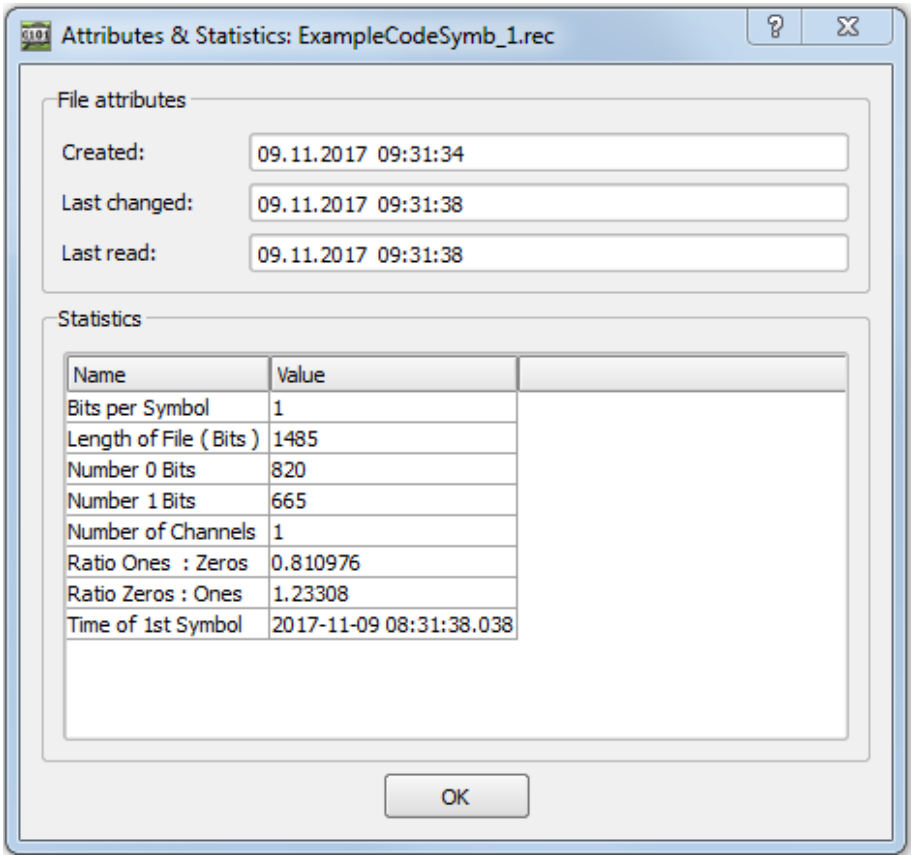


Figure 33: Attributes & Statistics

The dialog shows:

- Create date and time of the file (bitstream)
- Last change date and time
- Last read date and time of the file (bitstream)
- Number of bits per Symbol
- Length of current bitstream in bits
- Total number of zeros in the bitstream
- Total number of ones in the bitstream
- Number of channels in the bitstream
- Ratio of ones to zeros
- The reversal, i.e. the ratio of zeros to ones
- The time of the first symbol in the bitstream

Note: If the file is modified while this dialog is open, the statistics displayed will not be updated. Update the statistics by closing this dialog after a change to the stream and opening it again using <**Alt**>+<**T**>.

3.6.4. Close File and Terminate go2ANALYSE

To close the current bitstream choose <**File**><**Close File**> or press <**Alt**>+<**C**>. All displays associated to a specific bitstream are closed along with the file.

To terminate go2ANALYSE choose <**File**><**Exit**> or press <**Alt**>+<**F4**> or click the  icon of the main window.

In case the bitstream was modified before terminating go2ANALYSE or closing the file, you will be asked to save the modifications. If this message is acknowledged, the bitstream will be saved otherwise all changes will be lost.

3.7. Bit Display

<**Bit Display**> can show bits or code symbols. A code symbol is a letter or a figure from a code table, representing the serialized sequence of bits.

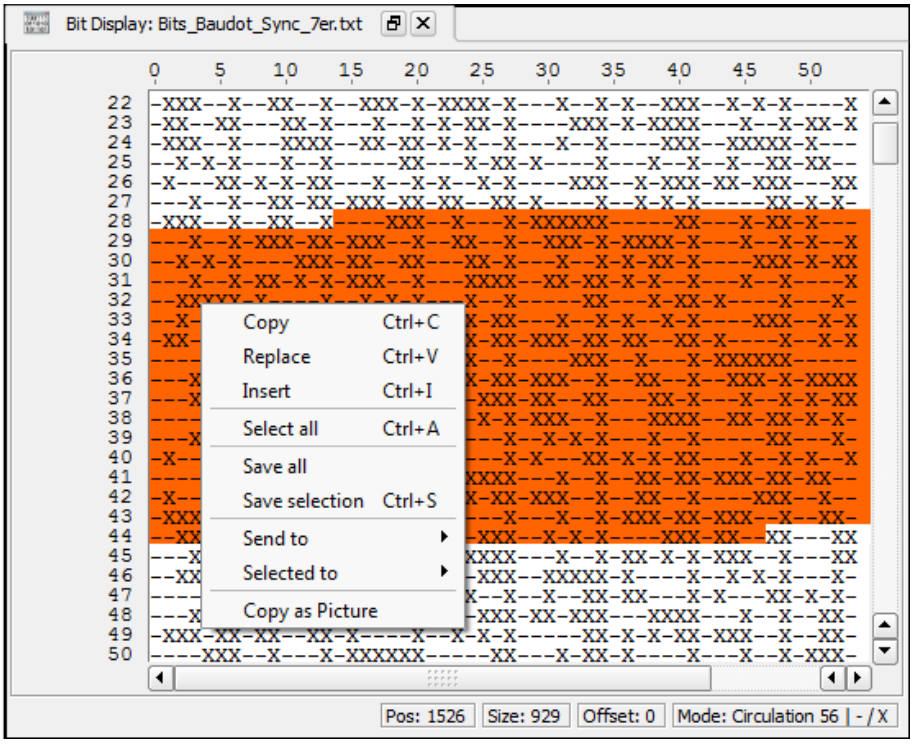


Figure 34: Exemplary Bit Display

The X-axis shows the current column of the bitstream, the Y-axis the current row.

<**Bit Display**> allows for an overview of sequences of symbols. It is used for the analysis of repeating bit patterns. Parts of the bit patterns can be highlighted. The highlighted part of the bit pattern or all bits shown in <**Bit Display**> can be saved in one file.

The scrollbars are only visible if the bitstream is too large to be fully displayed within the current size of <**Bit Display**>.

The status bars shows the current mode and the current offset in bits.

If the parameter serialization is changed, Mode shows the current selection as *MSB* or *LSB*.

3.7.1. Parameters

<**Bit Display**> parameters are displayed by activating the <**Parameters**> tab.

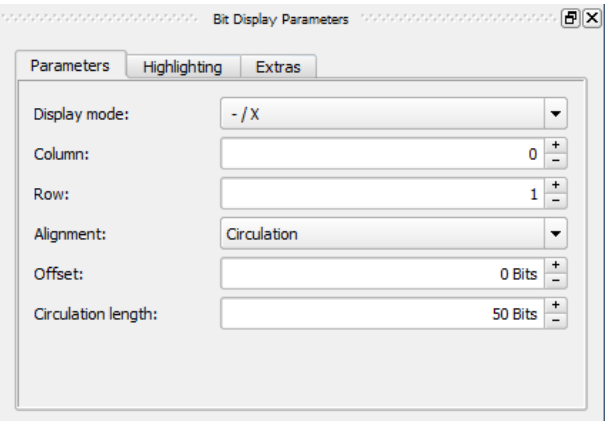


Figure 35: Bit Display Parameters - Parameters Tab

The following parameters are available:

Parameter	Valid Range	Function
<**Display Mode**>	- / X . / 1 Bright/Dark L/H	This parameter sets the mode in which the bits are displayed. When the bits are displayed the character left of the "/" stands for bit 0 and the one on the right of the /" stands for bit 1. Mode – Description: – / X: Displays zeros as"–" and ones as "X" . / 1: Displays zeros as "."and ones as "1" Bright/Dark: Displays zeros in bright and ones in dark squares L/H: Displays zeros as "L" and ones as "H"
<**Column**>	0 -	Sets the start column of the displayed bitstream. A change of column is only possible if the current circulation length is greater than the visible section of <**Bit Display**>. If column is increased by one the bits are "shifted" one column to the left, if it is decreased, the bits are "shifted" to the right. The minimum value of column is zero, the maximum value depends on the current circulation length.

Parameter	Valid Range	Function
<**Row**>	1 -	Sets the start row of the displayed bitstream. A change of row is only possible if the bitstream at the current circulation length has more rows than the visible section of <**Bit Display**>. If row is increased the bits are "scrolled" upwards by one row, if it is decreased, the bits are "scrolled" downwards to the right. The minimum value of row is one, the maximum value depends on the length of the bitstream and on the current circulation length.
<**Alignment**>	Circulation Burst	Changes the mode for defining the length of a row. Mode: Circulation — The parameter circulation defines the length of a row. Mode: Burst — The length of a burst (in symbols) defines the length of a row. Bursts are limited to a max. length of 10,000 bits.
<**Offset**>	0 - 1000	Defines the number of the first bit displayed in the stream from which the bitstream is displayed. If <**Offset**> is > 0 then the first offset bits will not be displayed or serialized.
<**Circulation length**>	2 - 10000	Defines length of a row in bits/code symbols. Changing the circulation length changes the alignment of the bit/code symbols which allows for a visualization of repeating bit patterns. If the circulation length is greater than the visible section of the <**Bit Display**>, the invisible section can be displayed by changing column or by means of the horizontal scrollbar.

Table 14: Bit Display Parameters

3.7.2. Highlighting

The parameters of the current bit selection (the highlighted area) are displayed and changed by activating the <**Highlighting**> tab.

Use the mouse to change the current selection:

- Move the mouse pointer to the start position of the block to be highlighted
- Press the left mouse button
- If the <**Shift**> key is held down during the selection, a block selection is created. If the block selection is already activated via the parameters, the block selection is deactivated again by pressing and releasing the <**Shift**> key.
- Drag the highlight to the end position of the block without releasing the left mouse button
- After releasing the mouse button, the desired block has been defined

Use the displayed parameters to change the current selection:

- When increasing the start below column, the selected area will shrink towards the right of <**Bit Display**>, or when decreasing it will increase towards the left
- When increasing the end below column, the selected area will increase towards the right of <**Bit Display**>, or when decreasing it, it will shrink towards the left
- When increasing the start below row, the selected area will shrink towards the end of <**Bit Display**> row by row, or when decreasing it, it will increase towards the beginning
- When changing the end below row, the selected area will increase towards the end of <**Bit Display**> row by row, or when decreasing it, it will shrink towards the beginning

The end row is always greater than the start row, the same rule applies to <**Column**>.

Use the right mouse button to open <**Bit Display**>. This will enable you to

- Copy the selected bits to the clipboard
- Replace previously copied bits to the bitstream
- Insert previously copied bits to the bitstream
- Save bits in a file
- Send bits to an external program

The individual items of the highlight option can be edited:

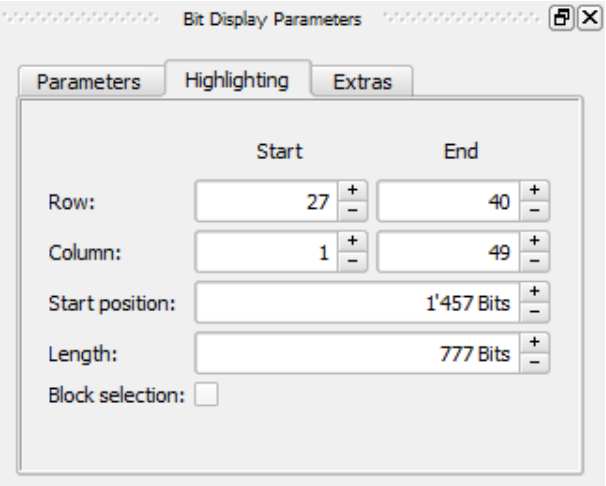


Figure 36: Bit Display Parameters - Highlighting Tab

Parameter	Function
<**Row**>	Defines the start and the end of the row of the highlighted block
<**Column**>	Defines the start and the end of the column of the highlighted block
<**Start position**>	Defines the start of the selection in bits if block selection disabled
<**Length**>	Defines the length of the selection in bits if block selection disabled
<**Block selection**>	When deactivated, all bits are marked continuously from the start position. When block selection is activated, a rectangular section (matching columns only) will be highlighted, i.e. the bits to the left of the start column and the bits to the right of the end column are not marked. Alternative switching with the <**Shift**> key while marking with the mouse. The switching of the block selection can influence the already marked area.

Parameter	Function

Table 15: Bitstream Highlighting Parameters

3.7.3. Extras

The font size or bit size of <**Bit Display**> can be edited activating the <**Extras**> tab.

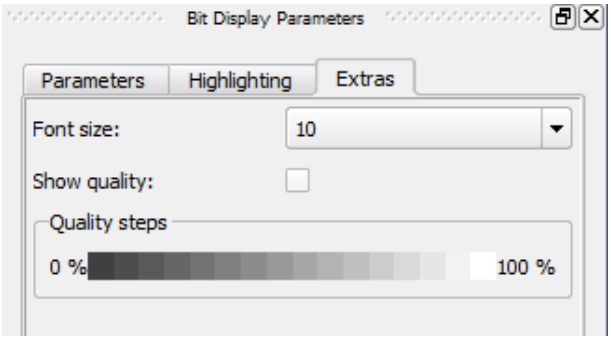


Figure 37: Bit Display Parameters - Extras Tab

The valid range of the <**Font Size**> is 6 to 20 points.

3.8. Text Display

The Text Display can display bits or code symbols. A code symbol is a letter or a figure from a code table, it represents the serialized sequence of bits. It is possible to synchronize the Text Display with the stream currently shown in <**Bit Display**>.

In synchronized mode, the part of the bitstream currently shown in <**Bit Display**> is sent to the Text Display where it is mapped to text. This function allows for tracking the effects of changes made to the bitstream to the text.

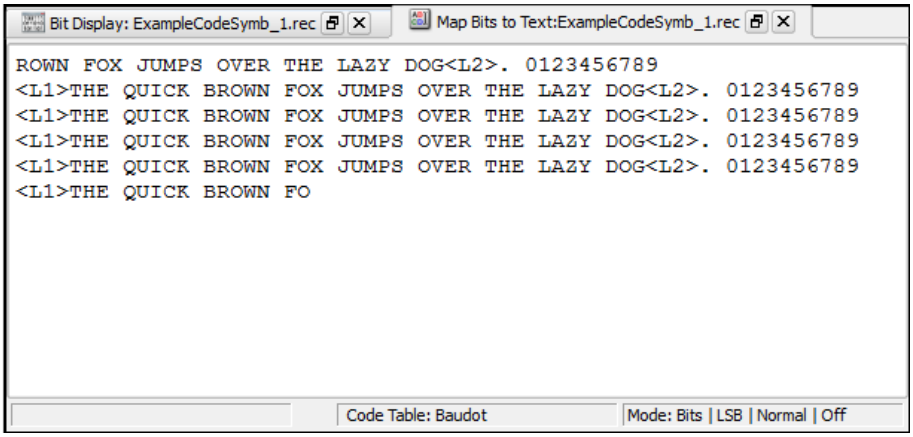


Figure 38: Exemplary Text Display

3.8.1. Parameters

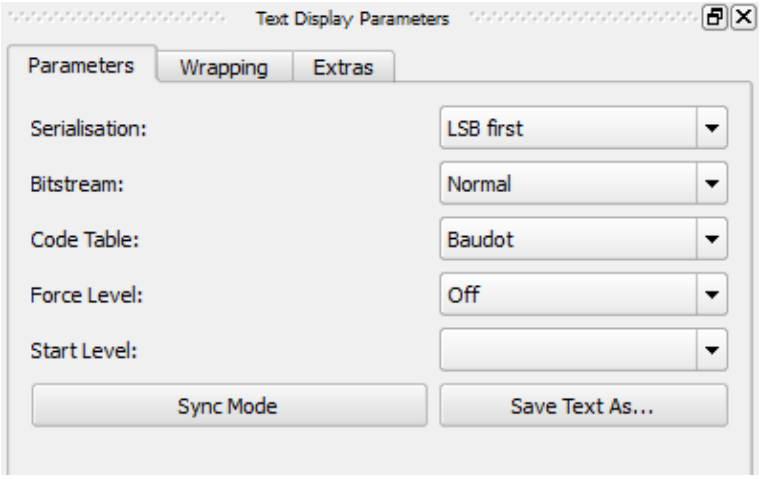


Figure 39: Text Display Parameters - Parameters Tab

The following parameters are available:

Parameter	Valid Range	Function
<**Serialisation**>	LSB first MSB first	The combination mode of the bits to code symbols to is set by the following parameters: Mode: LSB first — The least significant bit (the leftmost bit) has the highest bit of the code symbol. Mode: MSB first — The most significant bit (the rightmost bit) has the highest bit of the code symbol.
<**Bitstream**>	Normal Inverse	Mode: Normal — All bits are serialized straight into a code symbol. Mode: Inverse — All bits are inversed prior to be serialized into a code symbol.
<**Code Table**>	-/X ...	Determines the code table to use for coding the bits to code symbols. These code tables may be selected individually (see chapter Built-In Code Tables on page 114 for details).
<**Force Level**>	Off Level 1 ... Level n	You can change the mode of coding the bits to code symbols. Mode: Off — Figure shifts and letter shifts are allowed for coding Mode: Level 1 — All bits are coded to symbols from Level 1 of the current code table. No other symbols are displayed. Mode: Level n — All bits are coded to symbols from Level n of the current code table. No other symbols are displayed.

Parameter	Valid Range	Function
<**Start Level**>	Level 1 ... Level n	If this option is checked the display assumes that a level shift to the chosen level has been set prior to the start of coding the bits to code symbols.
<**Sync Mode**>	On/Off	Upon activation, the Text Display is synchronised with <**Bit Display**>. A change of certain parameters of <**Bit Display**> will also affect the Text Display. The parameters are Circulation length, Offset, Font size, Row and Column. If any bits are affected by a bit-editing function the text in the Text Display will be affected as well.
<**Save Text As...**>		On activation, the display text can be saved either as ASCII based txt file or UTF8 based file.

Table 16: Text Display Parameters

All parameters except <**Save Text As...**> are disabled if the Text Display is used to view the output of a decoder.

3.8.2. Wrapping

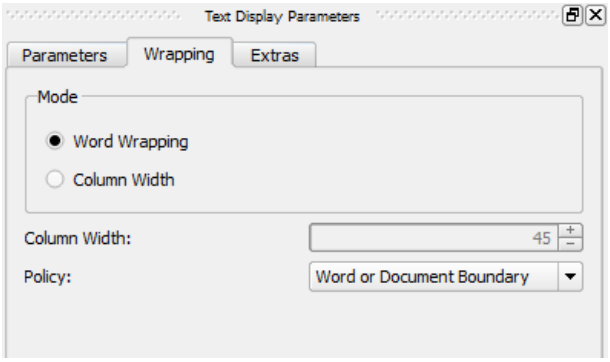


Figure 40: Text Display Parameters - Wrapping Tab

Text wrapping can be changed by activating the <**Wrapping**> tab.

Parameter	Function	
<**Mode**>	The combination mode of the bits to code symbols to is set by the following parameters:	
	Mode	Description
	Word wrapping	Wraps the text at the current width of the display (this is the default). Wrapping is at blanks by default, this can be changed by means of Policy.
	Column Width	Wraps the text at a fixed number of character columns from the display's left. Wrapping is at blanks by default, this can be changed by the means of Policy.

Parameter	Function	
<**Column Width**>	Sets the position (in characters) where text will be wrapped. Unless the policy setting is Anywhere the text will be wrapped at the nearest blank to the current column width.	
<**Policy**>	Mode	Description
	Word Boundary	Break lines at word boundaries, e.g. spaces or new lines.
	Anywhere	Breaks the text anywhere also within words.
	Word or Document Boundary	Break lines at blanks, e.g. spaces or new lines if possible. Break it anywhere otherwise.

Table 17: Text Display Parameters

3.8.3. Extras

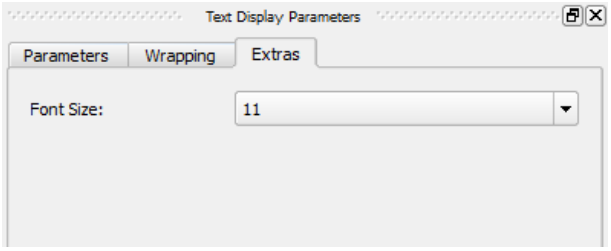


Figure 41: Text Display Parameters - Extras Tab

The <**Font Size**> of a bit in <**Bit Display**> can be changed on <**Extras**>. The valid range is 6 to 20 points.

4. Measurement Features

4.1. Measurement Menu

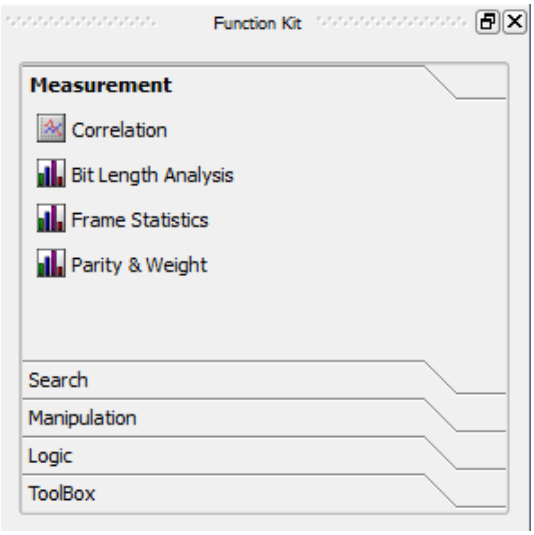


Figure 42: Function Kit - Measurement Menu

4.2. Autocorrelation and Cross-correlation

<**Function Kit**><**Measurement**><**Correlation**> serves to apply an autocorrelation, cross-correlation with selected bits or a cross-correlation with bits sequence.

When selected, the following dialog box is displayed:

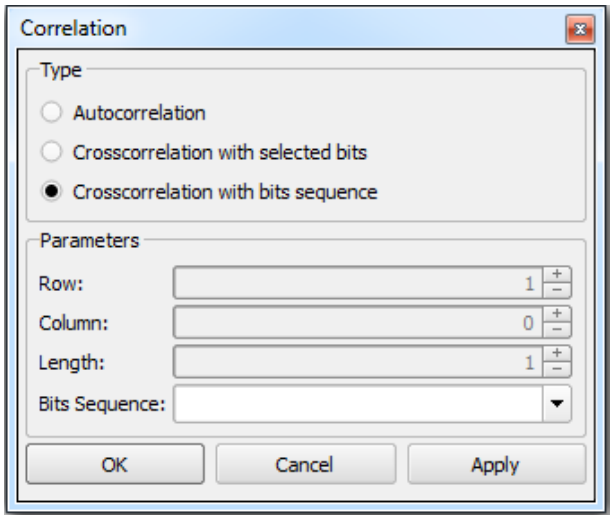


Figure 43: Dialog Box for Cross-correlation with bits sequence

Three types of correlation are available:

- Select <**Autocorrelation**> to execute a circular autocorrelation of the bits after clicking <**OK**> or <**Apply**>.
- Select <**Cross-correlation with selected bits**> to enable the spin boxes for <**Row**>, <**Column**> and <**Length**> of <**Bits Sequence**>. If the box <**Bits Sequence**> is blank, the software correlates the bits that are part of the frame described by the parameters <**Row**>, <**Column**> and <**Length**> with the bitstream. <**Cross-correlation with selected bits**> is non-circular!
- Select <**Cross-correlation with bits sequence**> to enable the spin boxes for <**Bits Sequence**>. The entered bit sequence is correlated with the current bitstream. <**Cross-correlation with bits sequence**> is non-circular!

Circular and Non-Circular Autocorrelation

Circular autocorrelation requires creating a duplicate of the original bitstream.

The duplicate is correlated with the original bitstream and then shifted by one bit towards the end of the stream. The last bit of the duplicate before the shift is wrapped and inserted as the first bit of the duplicate after the shift.

The result of the autocorrelation is calculated as

(Sum of consistent bits – Sum of inconsistent bits) divided by the sum of all bits.

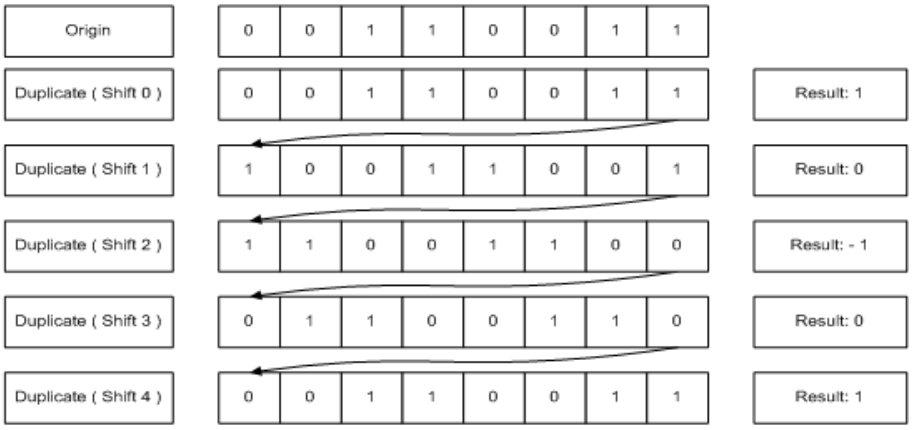


Figure 44: Circular Autocorrelation

The results of circular autocorrelation are repeated if the duplicate is shifted by more than half the length of the bitstream, without any additional information.

A non-circular autocorrelation will not duplicate the entire bitstream. It correlates the bitstream only with part of itself or with a bit sequence entered by the user. Said part, or the entered bit sequence respectively (assume that either of them is N bits long), is correlated with part of the original bitstream with same length.

After the correlation for shift 0, the part of the *original* bitstream is shifted by 1 bit. Now the former first bit is deleted, all other bits are shifted towards the beginning of the part and the last bit is removed from the original stream (see Figure 45 (the bitstream is the same as in the example above)).

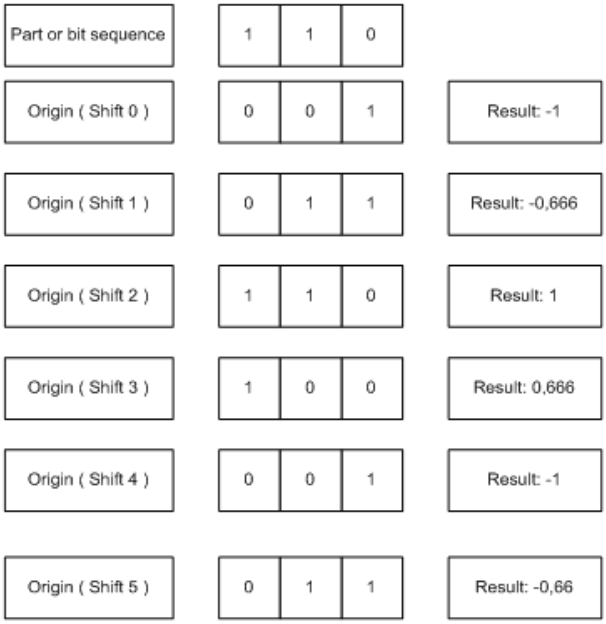


Figure 45: Non-Circular Autocorrelation

The results of the non-circular autocorrelation will not be repeated if the duplicate is shifted by more than half the length of the bitstream.

Note: go2ANALYSE restricts the use of autocorrelation to bitstreams with a size greater than 32 bits.

Example

Apply the autocorrelation to the file *LFSR1X6X7_0.txt:*

Open the file, click the radio button <**Autocorrelation**> and click the <**OK**> button in the dialog box.

go2ANALYSE first displays a progress dialog box which serves to control the progress of the autocorrelation and to cancel its further execution.

To abort the calculation of the autocorrelation, select <**Cancel**>.

Once the progress dialog box is closed, go2ANALYSE opens a Measurement Display (unless already displayed) and shows the result:

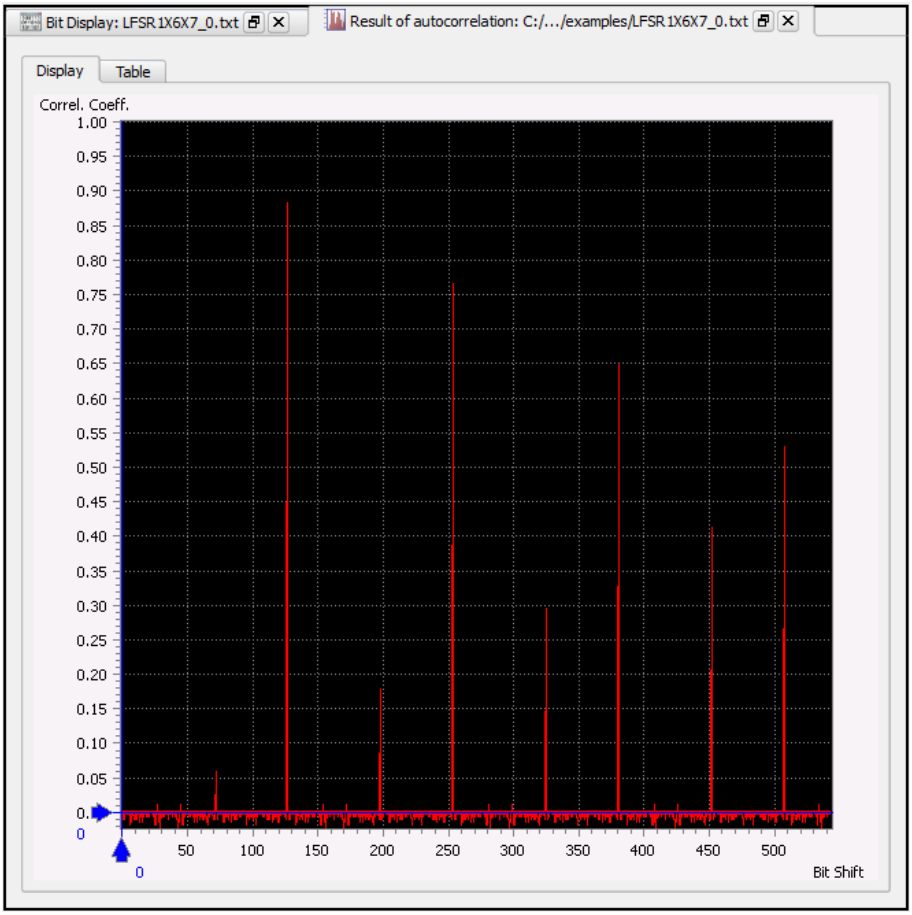


Figure 46: Exemplary Autocorrelation Results

Significant positive peaks indicate a high self-similarity of the bitstream with its duplicate after the attributed number of shifts.

When using circular autocorrelation, the peak at shift 0 must be ignored. The duplicate at shift 0 is a copy of the origin, so the peak at shift 0 is always 1.

Significant negative peaks indicate an inverse similarity of the bitstream with its duplicate.

A high similarity of a bitstream usually occurs if the stream contains repeating bit sequences. Therefore, when entering the distance between two peaks as circulation length value, the similarity of the sequences will be easily visible in <**Bit Display**>.

In the screenshot above, the first significant peak is located at shift 127, the second at shift 254, i.e. the distance between those peaks is 127. This is the number of bits after which the whole pattern is repeated.

4.2.1. Result Table for Autocorrelation and Cross-correlation

This is the table for results of autocorrelation or cross-correlation:

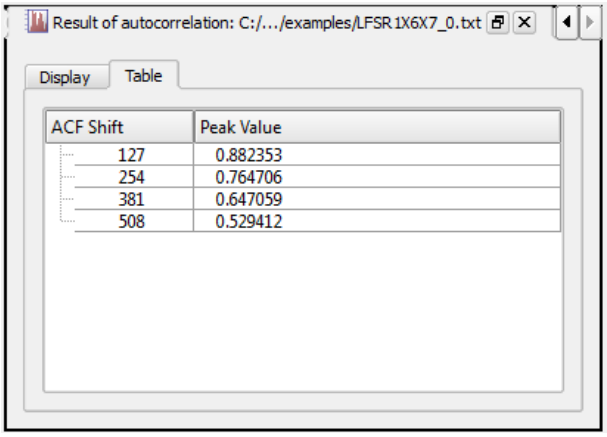


Figure 47: Result Table for Autocorrelation

The first column lists the ACF Shift, the second the result of the autocorrelation for this shift.

The table shows only results greater than half the maximum result.

If the number of results to display exceeds 10,000, only the first 10,000 results will be displayed.

When the result of an autocorrelation or correlation is displayed in this result table, it is possible to double click the first column, and the value of the double-clicked shift will be set as new circulation length of <**Bit Display**>.

4.3. Bit Length Analysis

This function analyses the bitstream with respect to the number runs of ones and zeros with a definite length. On clicking <**Function Kit**><**Measurement**><**Bit Length Analysis**>, the following dialog box is displayed:

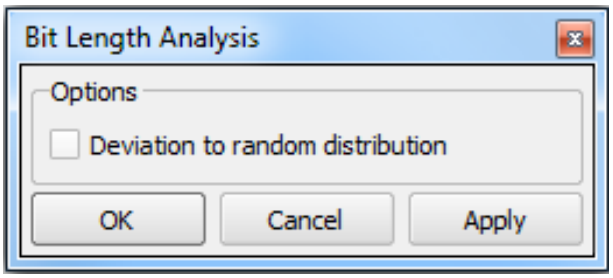


Figure 48: Dialog Box for Bit Length Analysis

If the option <**Deviation to random distribution**> is checked, the function will show the deviation between the results of the bit length analysis of the current bitstream and the result of a bit length analysis of a bitstream with a random sequence of bits. If the option is unchecked, the function will show the absolute result of the bit length analysis.

The definition of the terms run, block and breach is outlined below:

- A run is the designation for a sequence of the same bits, either zeros or ones
- A block is the designation for a run of ones
- A breach is the designation for a run of zeros

Example

We shall use the following bitstream example: 0001 1110 1001. This bitstream features (in order of occurrence):

- One breach with a length of 3
- One block with a length of 4
- One breach with a length of 1
- One block with a length of 1
- One breach with a length of 2
- One block with a length of 1

Consequently, there are two blocks with a length of 1, all other blocks occurring only once. Apply the run analysis to the exemplary bitstream *runanalysis.txt*. It consists of the example sequence repeated 27 times.

On clicking <**OK**> or <**Apply**>, go2ANALYSE opens a Measurement Display (unless already displayed) and shows the result for the current bitstream. The displayed result is as follows:

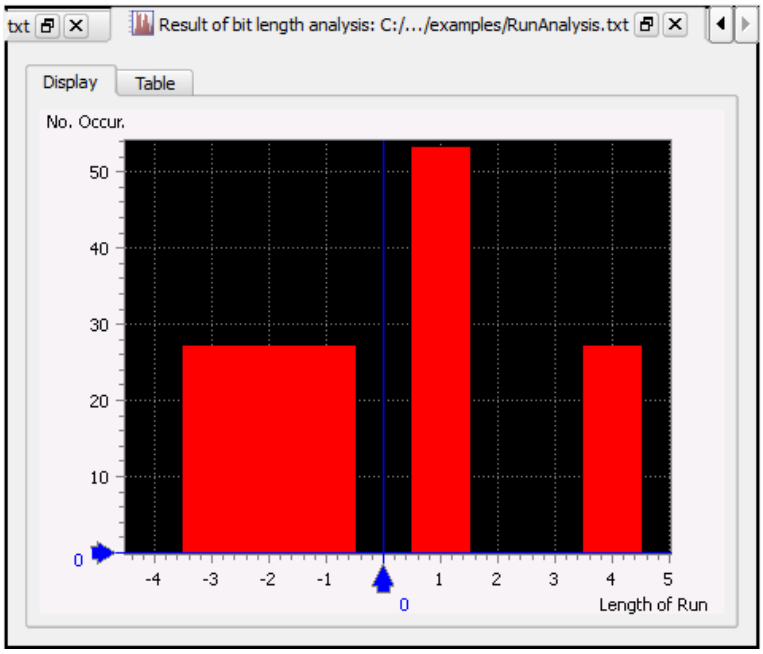


Figure 49: Exemplary Results of Bit Length Analysis

The display shows a "bundle" of bars printed on the X-axis. Each bar represents one run, either block or breach. The height of the bar reflects the absolute number of occurrences of this run (since <**Deviation to random distribution**> was left unchecked), the width of all bars is always one. The location of the bar on the X-axis indicates the length of this run.

Negative lengths represent breaches, positive lengths represent runs of blocks. The absolute value must be taken to obtain the real length. The zero value is always blank.

The blocks occurred are plotted toward positive X, the occurred breaches are plotted towards negative X. The null is always blank.

When measuring the bars, the results are:

- Number of breaches with lengths of 3, 2 and 1: 27

- Number of blocks with a length of 1 (bar at 1): 54 (because this breach occurred twice in the primary example sequence)
- Number of blocks with lengths of 2 and 3: 0
- Finally, the block with a length of 4 occurred 27 times

Changes in the displayed result change when checking <**Deviation to random distribution**>:

As stated above, the absolute result of the run analysis is compared to the run analysis of the bitstream with a sequence of random bits.

In a bitstream with a random sequence of bits, when each bit has been generated by a random generator, the probability of a one or a zero is $\frac{1}{2}$. Assuming a bitstream of random bits with a size of 500 bits, there should be 250 ones and 250 zeros in this stream. The probability of a block with a length of 2 can be calculated as:

$\frac{1}{2}$ * Size of Stream * Probability for a One * Probability for a One

thus $\frac{1}{2}$ * 250 * $\frac{1}{2}$ * $\frac{1}{2}$ = 31.25.

Generally, the probability of occurrence for a block or a breach of a length N can be calculated as:

$p(N) = \frac{0.5 * Bitstreamize}{2^{N+1}}$

If <**Deviation to random distribution**> is checked, go2ANALYSE will calculate the total number of occurrences for each run as it would occur in a bitstream with random bits.

The calculated results are then compared with the results of the run analysis of the current bitstream, and the deviation between these results is displayed.

When applying <**Deviation to random distribution**> to the example, the Measurement Display will show the following result:

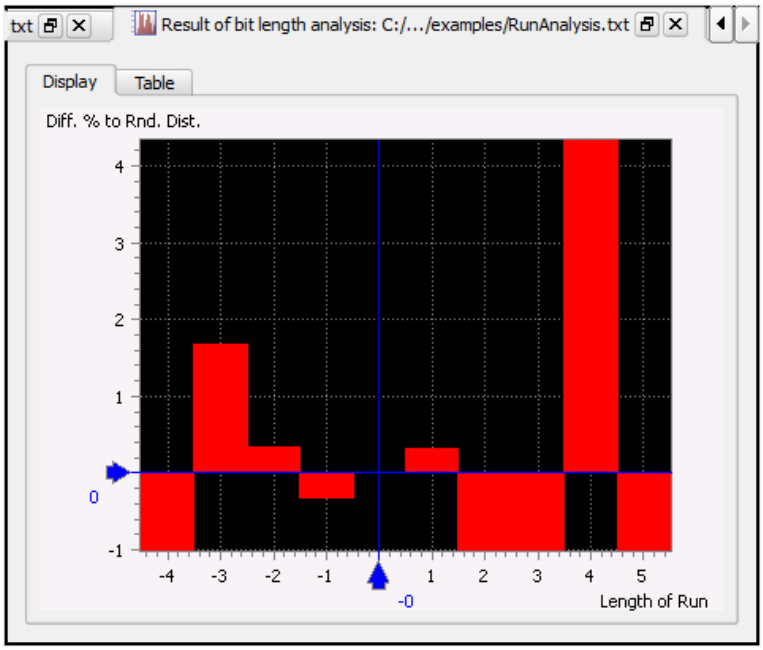


Figure 50: Exemplary Results with Option Deviation to Random Distribution Checked

Result Interpretation

The bars for blocks and breaches are printed at the same positions as with the option unchecked, but the value (height) of each bar indicates the difference (deviation) to the random bitstream.

A value of -1 indicates that this run has *never* occurred in this bitstream.

Bars with a value less than zero values indicate that the run in question has occurred less often in the current bitstream than it would in a random bitstream.

If a bar has a value of 0 (except the bar at zero which is always 0), this bar occurs in the current bitstream as many times as it would in a random one.

Bars with a value greater than zero indicate that the run in question has occurred more often in the current bitstream than it would in a random bitstream.

Note: If a run with a length greater than 32 bits shows in the bitstream, the value will always be 2!

4.3.1. Result Table for Bit Length Analysis

The result table for a bit length analysis looks as follows:

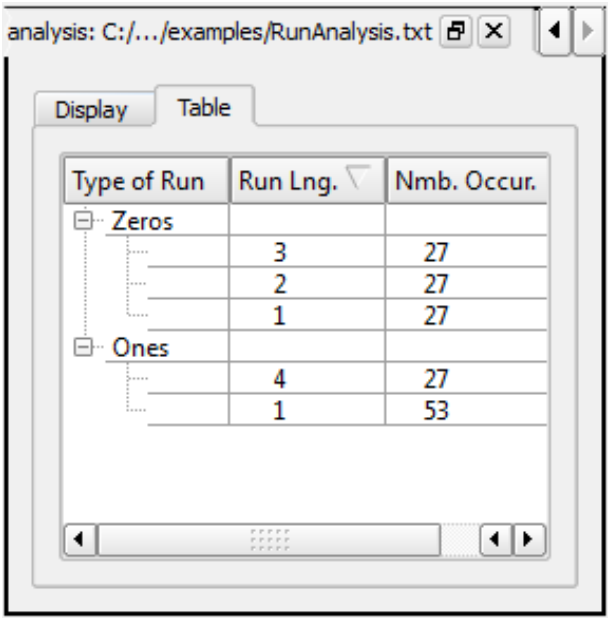


Figure 51: Result Table for Run Analysis

- The first column separates the blocks (Ones) from the breaches (Zeros)
- The second column contains the length of a run found
- The third column shows the total number of occurrences

When <**Deviation to random distribution**> is checked, the third column lists the deviation between the current result and the run analysis of a bitstream with random bits.

The interpretation of the results is the same as in the previous paragraph.

Note: Values for the total number of occurrences are limited to a maximum of 2^{24}

4.4. Measurement Display

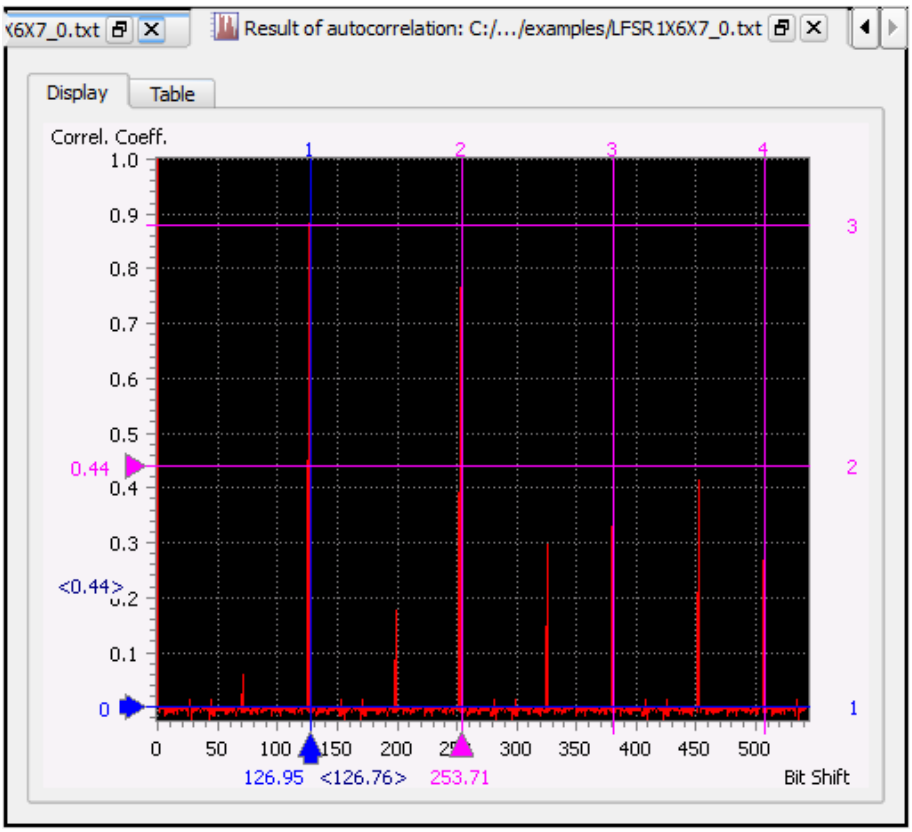


Figure 52: Measurement Display Example

4.4.1. Parameters

On the one hand, the displayed area of the measuring results changes when editing the listed parameters.

On the other, when applying a measurement function to the active display, the Measurement Display changes the current parameters to fit the current result.

Parameter	Valid Range	Does This
Minimum X	-1,000,000 – +999,999	Sets the start value of the X-axis
Maximum X	-999,999 – +1,000,000	Sets the end value of the X-axis. Is always greater than Minimum X.
Minimum Y	-1,000,000 – +999,999	Sets the start value of the Y-axis
Maximum Y	-999,999 – +1,000,000	Sets the end value of the Y-axis. Is always greater than Minimum Y.

Table 18: Measurement Display Parameters

4.4.2. Cursors

The cursor parameters are displayed on the <**Cursor**> tab. You may insert cursors into the display to measure the displayed results. The respective cursor positions are displayed and editable.

Parameter Group	Parameter	Does This
Cursor Options		Activate or deactivate cursor directions
	X-Cursors On/Off	Toggle X-Cursor On/Off (activated/deactivated)
	Y-Cursors On/Off	Toggle Y-Cursor On/Off (activated/deactivated)
Cursor Mode		Changes of cursor mode will apply to all active cursors
	Normal	For each activated cursor direction two cursors are displayed (cursor 1: blue, cursor 2: magenta). The lowest parameter field labeled Distance 1-2 shows the distance between the cursors. Each cursor is moved separately.
	Harmonic	For each activated cursor direction, up to 20 cursors are displayed at equidistant intervals. The cursors will move by dragging the first cursor. The width of the interval between the cursors is changed by dragging the second or any following cursor. The lowest parameter field labeled Period shows the width of the cursor interval. Harmonic cursors serve for measuring periodic results.
	X-Y Cursor	For each activated cursor direction only one cursor is displayed. The lowest parameter field is labeled Value X / Value Y and shows the current position of the X- or the Y-cursor respectively. The X- and Y-cursors are moved together (if both are activated) by dragging the intersection point of the cursors lines.
Cursor Positions and Measurement		Read out the current positions of the cursors and change them by editing the values of the parameters for Cursor 1 and Cursor 2.
	Cursor 1	Value of the first (blue) X-cursor or Y-cursor. When changing this value, the first cursor will change position. The cursors cannot be moved outside the visible area by editing this parameter.
	Cursor 2	Value of the second (magenta) X-cursor or Y-cursor. When changing this value, the first cursor will change position. The cursors cannot be moved outside the visible area by editing this parameter.

<table>
<tr><th>Parameter Group</th><th>Parameter</th><th>Does This</th></tr>
<tr><td></td><td>Difference 1-2
Period
Value X /
Value Y</td><td>This parameter changes the mode. The bits are coded to code symbols.
<table>
<tr><th>Cursor Mode</th><th>Name</th><th>Parameter Description</th></tr>
<tr><td>Normal</td><td>Difference 1-2</td><td>Difference between cursors 1 and 2</td></tr>
<tr><td>Harmonic</td><td>Period</td><td>Interval between cursors</td></tr>
<tr><td>X-Y</td><td>Value X/
Value Y</td><td>Position of X-cursor
Position of Y-cursor</td></tr>
</table></td></tr>
</table>

Table 19: Measurement Cursor Parameters

On double clicking the cursors, the current *Difference/Period/Value X* will be set as new circulation length of <**Bit Display**>. This is especially useful to measure a periodical result e.g. the result of an autocorrelation. All cursors can be moved using the mouse. To move the cursor, place the mouse pointer on the cursor handle (arrow or triangle) or on the cursor line, press the left mouse button and move the mouse. The cursor is dragged with the mouse until you release the left mouse button.

- Remember: In X-Y Mode, both cursors will follow the mouse, so if you move to the side and upwards the X-cursor is moved to the side while the Y-cursor is moved upwards.
- On clicking inside the Measurement Display, the cursor closest to the current position of the mouse pointer will be moved to the position clicked.

4.4.3. Zoom Display

The buttons <**Zoom X-Axis**> and <**Zoom Y-Axis**> are only active when the cursor mode is not set to *X-Y Cursor*. On selection of <**Zoom X-Axis**>, the Measurement Display is zoomed in showing the area between the first two cursors.

On selection of <**Zoom Y-Axis**>, the Measurement Display is zoomed in showing the area between the first two cursors.

4.4.4. Extras

<**Extras**> features the <**Color scheme**> setting which can be used to change the current color scheme for the display:

Color Scheme	Cursor Color	Data Color	Background
Standard	blue, magenta	various colors	black
Inverse	blue, magenta	various colors	white
Monochrome	shades of grey	different shades of grey	white

Table 20: Measurement Display Color Schemes

4.4.5. Measuring Result Table

Each result displayed in the Measurement Display is also listed in the built-in result table. The layout of this result table depends on the type of result shown in the Measurement Display.

4.5. Frame Statistics

This function serves to analyze the bitstream by defining frames and calculating statistical data for each frame, displaying the results in a special display which facilitates the comparison of the frame statistics.

On clicking <**Function Kit**><**Measurement**><**Frame Statistics**>, the following dialog box is shown:

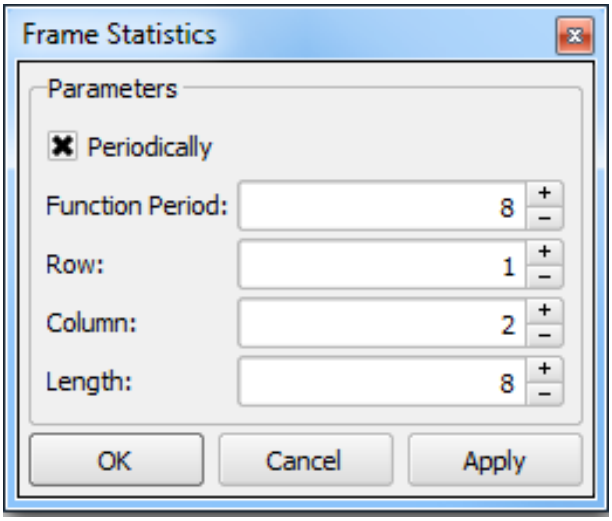


Figure 53: Dialog Box for Frame Statistics

The parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Length**> define the frames to which the function will be applied.

The parameter <**Function Period**> is only applied if the option <**Periodically**> is checked.

Example

The bitstream example used features a circulation length of 8.

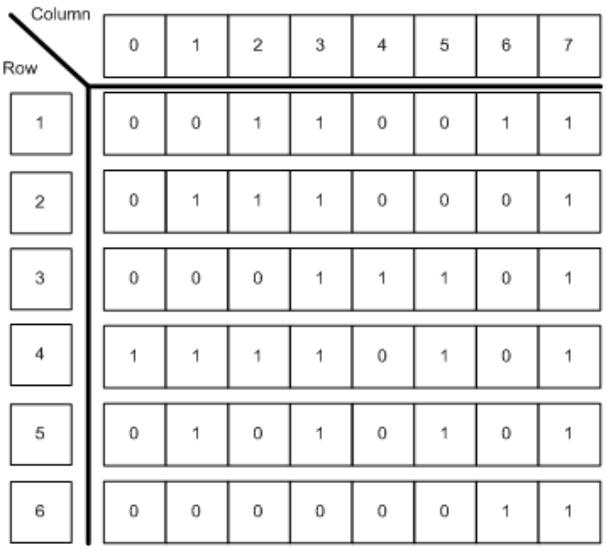


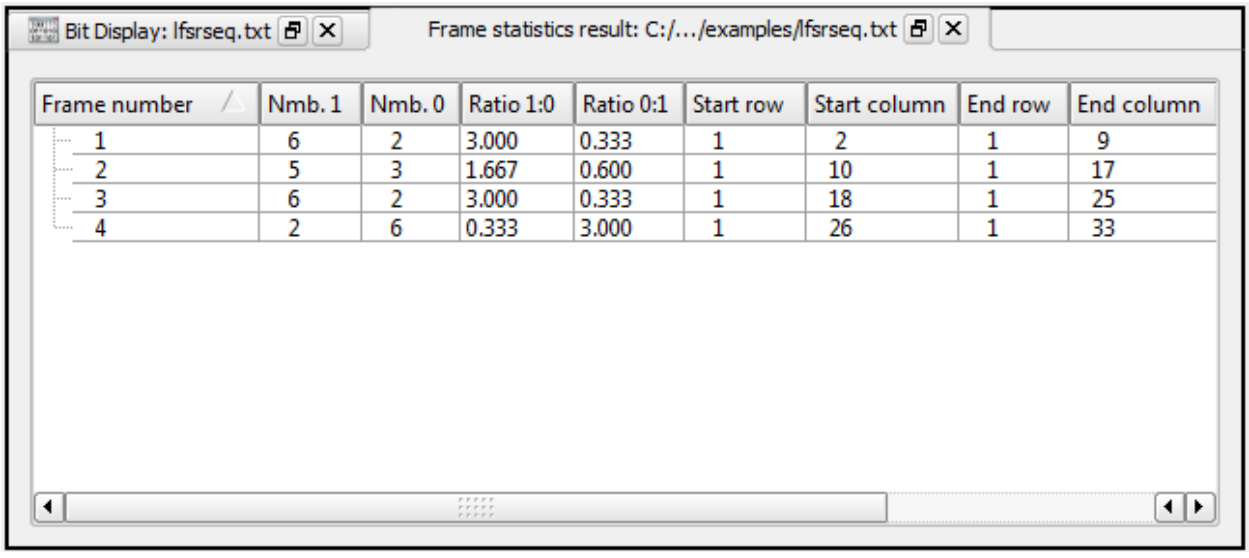
Figure 54: Bitstream Example

Apply the frame statistics function to the exemplary bitstream. To do so, enter the following parameter values:

- <**Periodically**> Checked
- <**Function Period**> 8
- <**Row**> 1
- <**Column**> 2
- <**Length**> 8

Press <**OK**> or <**Apply**>.

go2ANALYSE opens the Result Display. Do not confuse this display with the result table of the Measurement Display. The Result Display is an independent display which is not part of any other display:



Frame number	Nmb. 1	Nmb. 0	Ratio 1:0	Ratio 0:1	Start row	Start column	End row	End column
1	6	2	3.000	0.333	1	2	1	9
2	5	3	1.667	0.600	1	10	1	17
3	6	2	3.000	0.333	1	18	1	25
4	2	6	0.333	3.000	1	26	1	33

Figure 55: Exemplary Results of the Frame Statistics

Each frame analyzed is listed in the order of its occurrence in the bitstream. Consequently, the frame with number 0 must be the first frame analyzed, the frame with the highest number must be the last. In our example, frame number 0 covers the bits in row 1, frame number 5 the bits in row 6.

For each frame, the result table shows:

- The number of Ones in this frame
- The number of Zeros in this frame
- The ratio of Ones to Zeros in this frame
- The ratio of Zeros to Ones in this frame
- And the "coordinates" (start and end row, start and end column) of the frame indicating where each frame is located in the bitstream.

Note: When the current circulation length of <**Bit Display**> is changed while the Result Display is open, the "coordinates" for each frame will be adapted to the new circulation length.

4.6. Parity & Weight Statistics

This function serves to analyze the bitstream by defining frames, to calculate the parity bit for even and odd parities for each frame, and to show the results in a special display which facilitates the comparison with the statistics of other frames.

On clicking <**Function Kit**><**Measurement**><**Parity & Weight**>, the following dialog box is shown:

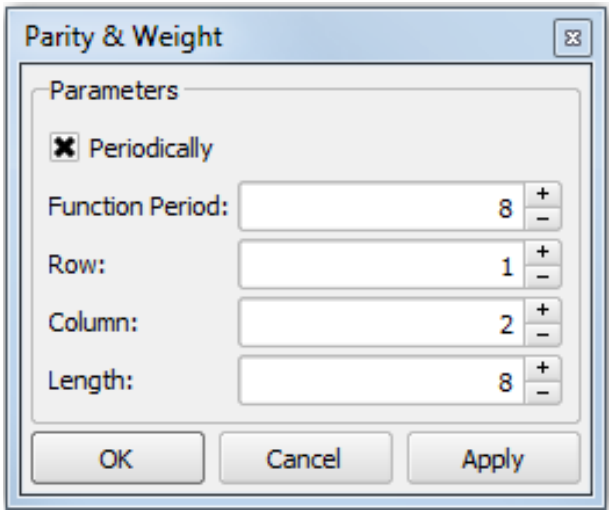


Figure 56: Dialog Box for Parity & Weight

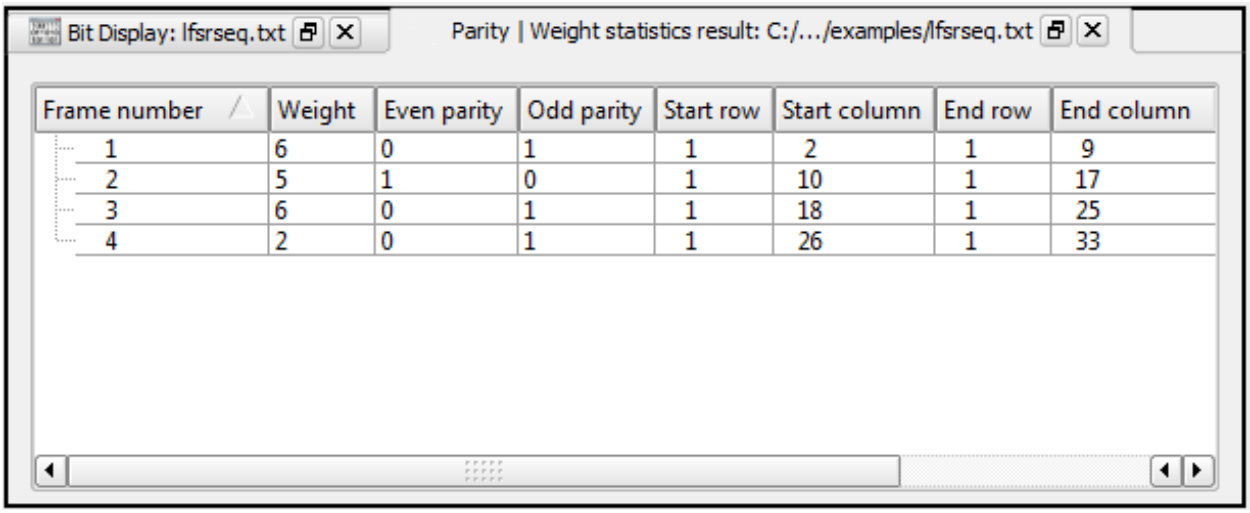
The parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Length**> define the frames the function is applied to.

The parameter <**Function Period**> is only applied if the option <**Periodically**> is checked.

Example

This function is applied to the bitstream used in the frame statistics example.

The Result Display shows the following results:



Frame number	Weight	Even parity	Odd parity	Start row	Start column	End row	End column
1	6	0	1	1	2	1	9
2	5	1	0	1	10	1	17
3	6	0	1	1	18	1	25
4	2	0	1	1	26	1	33

Figure 57: Exemplary Results of the Parity Weight Analysis Function

Each frame analyzed is listed in the order of its occurrence in the bitstream. Consequently, the frame with number 0 must be the first frame analyzed, the frame with the highest number must be the last. In the example, the frame number 0 covers the bits in row 1, frame number 5 the bits in row 6.

For each frame, the result table shows:

- The "weight" (weight meaning the sum of ones) of this frame
- The even parity bit, i.e. the bit which must be added to the sum of ones so the sum of ones is an even number
- The odd parity bit, i.e. the bit which must be added to the sum of ones so the sum of ones is an odd number
- The "coordinates" (start and end rows, start and end columns) of the frame indicating the position of each frame in the bitstream

Note: When changing the current circulation length of <**Bit Display**> while the Result Display is open, the "coordinates" for each frame will be adapted to the new circulation length.

4.7. Result Display

The third display in go2ANALYSE is the Result Display. The main element of this display is a table showing the various function results. Operation and handling of the Result Display are explained in detail in the respective chapters describing the go2ANALYSE functions whose results are displayed in this display.

These functions are:

- Frame Statistics function
- Parity & Weight Statistics function
- Search for bit sequences from LFSRs

5. Search Features

5.1. Search Menu

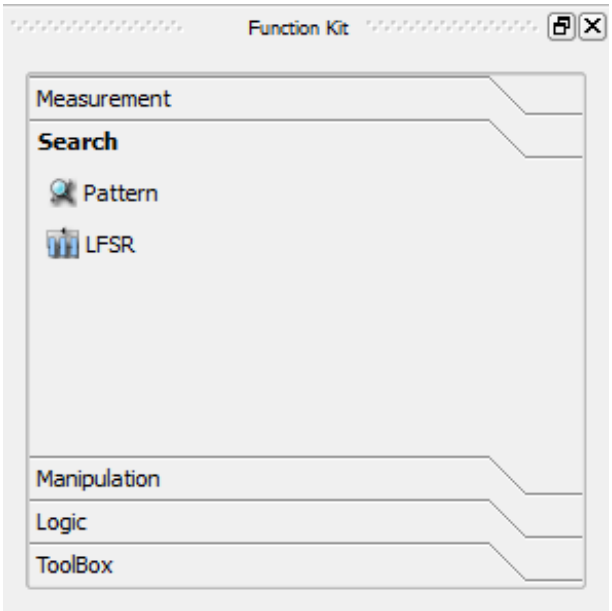


Figure 58: Function Kit - Search Menu

5.2. Pattern Search

This function searches all occurrences of the entered bit sequence in the bitstream and tags them with the chosen color.

On clicking the button <**Pattern**> in the Function Kit Search Window, the following dialog box is shown.

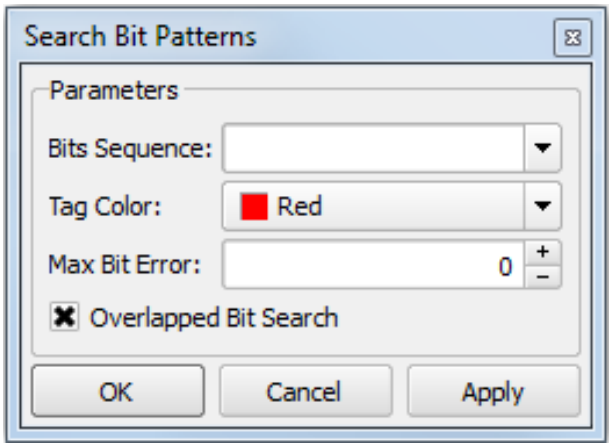


Figure 59: Dialog Box for Pattern Search

Enter the bit sequence to search for in the list box <**Bits Sequence**> (or choose an existing bit sequence on the property sheet and select an existing entry). Choose the color to tag the bits with on the property sheet with <**Tag Color**>.

On pressing <**OK**> or <**Apply**>, this function will find all occurrences of the entered sequence in the current bitstream.

Note: The characters entered in the list box <**Bits Sequence**> will be converted to bits using Table 21:

Character Input	Equivalent Bit
0	0
- (dash)	0
_ (underline)	0
L	0
. (Dot)	0
1	1
X	1
x	1
H	1

Table 21: Bit Sequence Conversion

Example

Open the file with the *LFSR1X6X7_0.txt*, copy the first 10 bits in row 2 to the clipboard and paste them in the field <**Bits Sequence**>, choose the tag color *Green* and press <**OK**>.

Subsequently, <**Bit Display**> shows Figure 60 (circulation length is 50):

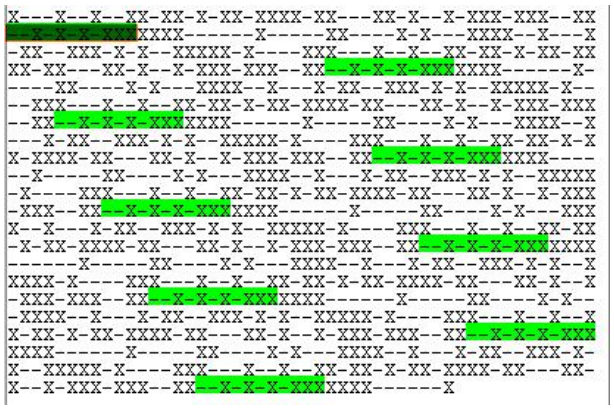


Figure 60: LFSR1X6X7_0.txt Bitstream after Search

Every bit in the bitstream can only have one tag color. Therefore, after searching the same pattern one more time with a different color, the bits will not be tagged with two colors or a mix of them but with the new color chosen.

5.3. Search for LFSR Sequences

This function serves to search for bit sequences generated by "linear feedback shift registers" (LFSR).

5.3.1. Introduction to LFSR Sequences and Berlekamp-Massey Algorithm

A LFSR is a shift register whose input is the Exclusive-OR (XOR) of some of its outputs. The outputs that influence the input are called *taps*. An LFSR with N taps (N is normally referred to as the linear complexity or length of the LFSR) is able to generate a pseudorandom binary sequence with a maximum length of 2^N-1 bits if it is constantly "fed" with zeros. As a precondition, the cells of the shift register have to be set to any non-zero state and then cycled. After the output of this bit sequence of 2^N-1 bits, the same sequence is repeated if the LFSR is continued to be fed with zeros. If the precondition is not met, the output bits of the LFSR are all zero if the input bits are zero.

The tap sequence of an LFSR can be represented as a polynomial modulo 2 - called the feedback polynomial. For example, if the taps are at the 1st and 3rd bits, the polynomial is 1+xˆ1+xˆ3 (see Figure 61).

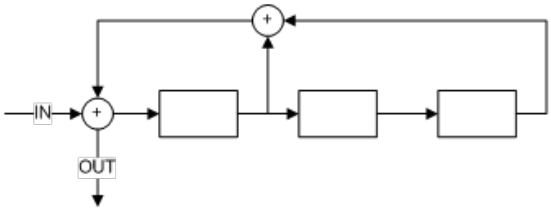


Figure 61: LFSR with Polynomial 1+xˆ1+xˆ3

The length of the pseudorandom binary sequence is maximal if the polynomial of the LFSR is a primitive one as it is called, i.e. it cannot be reduced to terms.

Given an output sequence of an LFSR you can reconstruct the generating LFSR of minimal size using the "Berlekamp-Massey Algorithm" (BMA). This algorithm can be used to find the shortest LFSR for a given output sequence.

To do so, the algorithm needs to test at least 2*N bits to reconstruct a generating LFSR with a linear complexity N, however, it is possible that the algorithm reconstruct only an LFSR with M < N because this LFSR was the shortest LFSR which could generate the tested sequence.

The algorithm always returns the linear complexity and the polynomial of the LFSR which generated the tested bit sequence.

Attention: It is fundamental to know that a primitive LFSR with a length X > N can generate the bit sequence of a primitive LFSR with a length N but the LFSR with a length N cannot generate the bit sequence of the LFSR with a length X.

5.3.2. Using LFSR Search

On clicking <**Function Kit**><**Search**><**LFSR**>, Figure 62 is displayed.

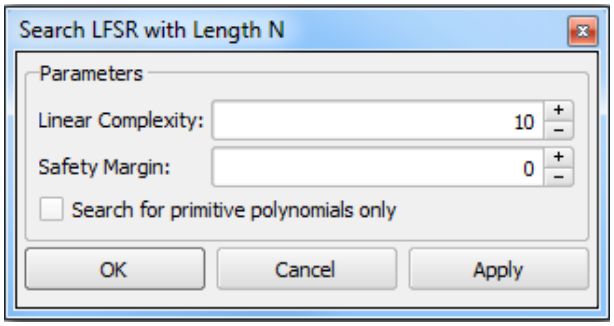


Figure 62: Dialog Box for LFSR Search

Adjust the maximum linear complexity of the LFSR to search for by entering the appropriate values in <**Linear Complexity**>. The parameter may range from 1 to 256.

The parameter <**Safety Margin**> allows for improving the reliability of the BMA's "statement" about the LFSR which generated the tested bit sequence.

Normally the BMA needs to test a number of bits equal to two times the entered linear complexity (N) to reconstruct the LFSR. If <**Safety Margin**> (we use *K* to refer to this parameter) is set to any value greater than zero, then 2*N + K bits will be tested by the BMA.

If the linear complexity returned after the test of 2*N + K bits is still N, then the reliability that this bit sequence was generated by the LFSR with length N and not with an LFSR with a length greater than N has improved.

Note: A value of 5 for *K* is a good compromise between improving the reliability and the required length of a bit sequence to test for LFSRs.

Example

To apply the LFSR to the exemplary bitstream *LFSR1X6X7_0.txt*, proceed as follows:

Load the bitstream, press <**LFSR**> and enter the following parameters:

- Linear complexity: 7
- Safety Margin BMA: 0
- Leave the option <**Search for primitive polynomials only**> unchecked
- Press <**OK**> or <**Apply**>.

go2ANALYSE will also open a new Result Display (unless already open) and display the results of the example:

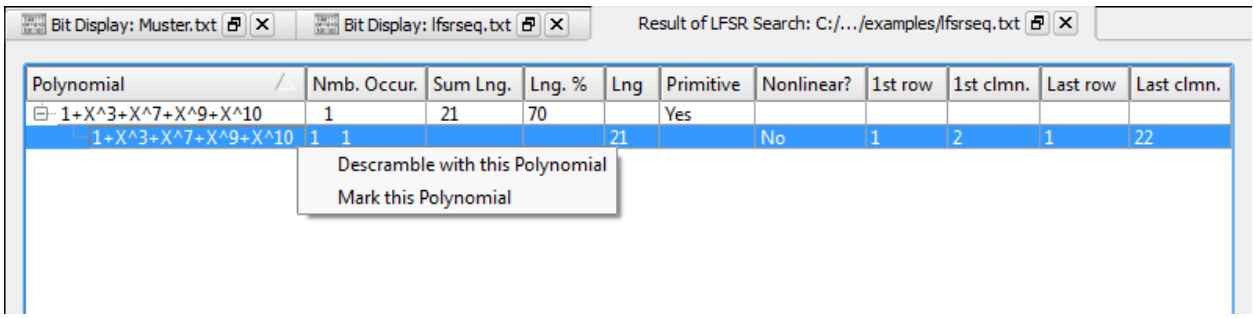


Figure 63: Exemplary Results of LFSR Search

Item	Explanation
Polynomial	The column *Polynomial* contains the polynomial of the LFSRs found. All polynomials with the same equation are grouped in one "family" and listed below a "parent" polynomial as it is called (see first row in Figure 63). Below each "parent" polynomial is a list of the "child" polynomials, these are all polynomials of the same order and with the same equation (the second and following rows in the screenshot show a list of the child polynomials listed in the first row). Once a new result of an LFSR search is displayed, all child polynomials are hidden. To see the list of child polynomials for a parent polynomial click the plus sign next to the equation of polynomial. If the list of child polynomials is open, the plus sign has changed to a minus (see first column in Figure 63).
Nmb. Occur.	Number of occurences the polynomial was found
Sum Lng.	The *Sum Lng.* column lists how many bits of this bitstream have been generated by LFSRs with this equation
Lng. %	The column value in *Lng. %* is calculated as: 100 * *Sum Lng.* divided by the summed length of all LFSRs found
Lng	The value in the column *Lng.* is the length of the bit sequence found in bits
Primitive	Indicates if polynomial is primitive or not
Nonlinear?	No in the column *Nonlinear?* indicates that the BMA did not find any unusual jumps in the linear complexity as it reconstructed the LFSR. Yes indicates that this LFSR is probably nonlinear.
1st row 1st clmn. Last row Last clmn.	All child polynomials are sorted by their start row (*1st row*). The "coordinates" of the start bit of each bit sequence the child polynomial belongs to is given by the values of start row and start column, the "coordinates" of the last bit of this sequence is given by the value of last row and last column.

Table 22: LFSR Parameters

Note: If the current circulation length of <**Bit Display**> is changed while the Result Display remains open, the "coordinates" for each frame are adapted to the new circulation length.

When right clicking any polynomial, a popup menu with the items <**Descramble with this Polynomial**> and/or <**Mark this Polynomial**> will be displayed. Clicking this item opens the dialog box for descrambling or for tagging, and the LFSR-Polynomial parameter is set to the right-clicked polynomial. For details on descrambling the bitstream see chapter Diff Bitstream on page 73.

6. Manipulation Features

6.1. Manipulation Menu

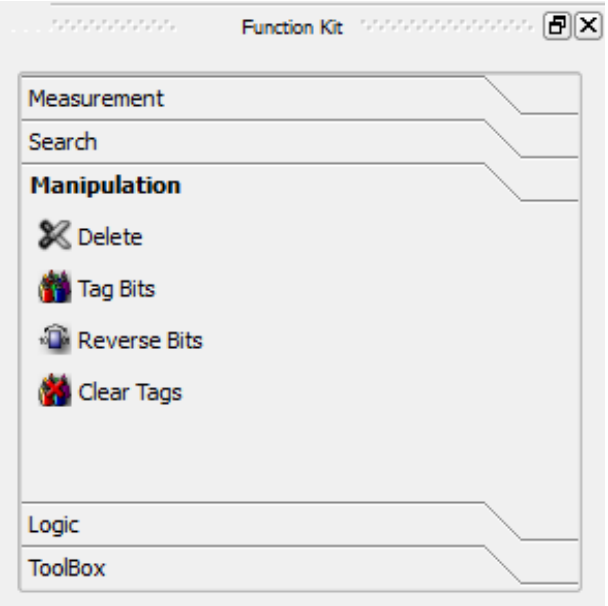


Figure 64: Function Kit - Manipulation Menu

6.2. Delete

Deletes the bits defined by the current function parameters from the bitstream. On clicking <**Function Kit**><**Manipulation**><**Delete**>, Figure 65 is shown.

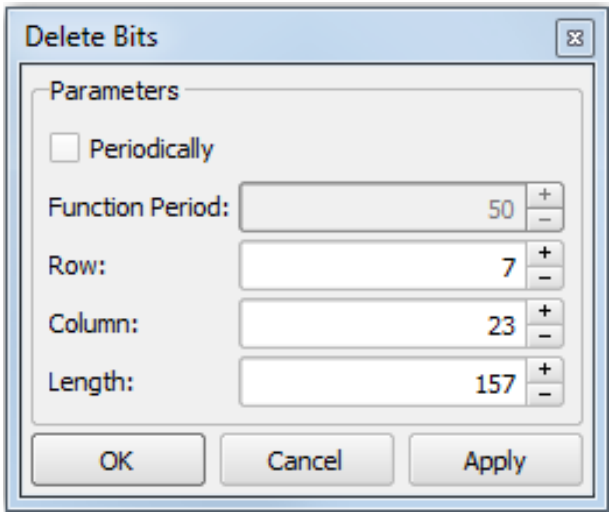


Figure 65: Dialog Box for Delete

The parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Length**> define which bits the function is applied to (see chapter Define Bits for Application of go2ANALYSE Functions on page 23 for details).

On clicking <**OK**> or <**Apply**> the function is applied to the stream. It may take a while until all bits defined have been deleted from the bitstream.

6.3. Tag Bits

Tags the bits defined by the current function parameters with the chosen color. On clicking <**Function Kit**><**Manipulation**><**Tag Bits**>, Figure 66 is shown.

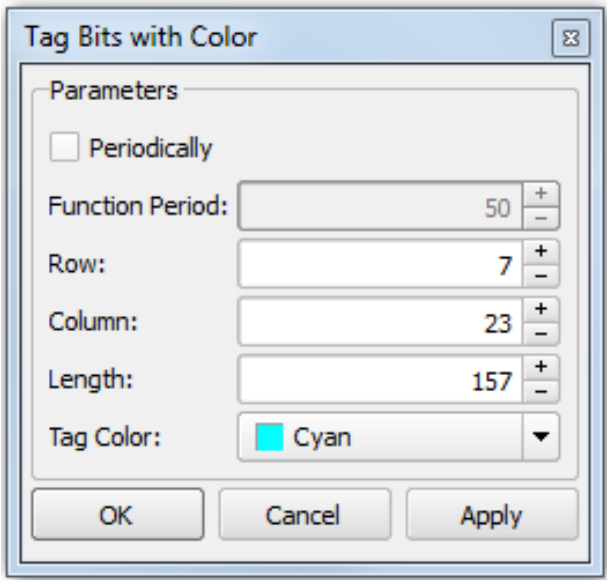


Figure 66: Dialog Box for Tagging Bits

The parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Length**> define which bits the function is applied to (see chapter Define Bits for Application of go2ANALYSE Functions on page 23 for details).

<**Tag Color**> defines the color in which the bits are tagged.

On clicking <**OK**> or <**Apply**> the function is applied to the stream.

Note: Each bit in the stream can only have one tag color at a time. Consequently, after searching the same pattern one more time with a different color the bits will not be tagged with two colors or a mix of them but with the new color chosen.

6.4. Reverse Bits

This function reverses bits in the frame which is defined by the parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Length**>. On clicking <**Function Kit**><**Manipulation**><**Reverse Bits**>, Figure 67 is shown.

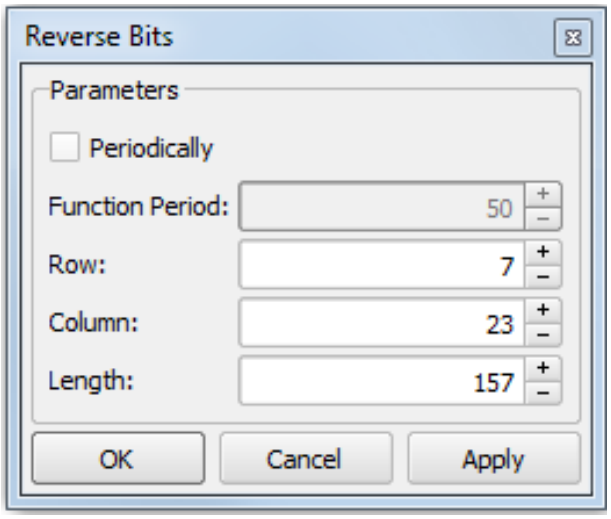


Figure 67: Dialog Box for Reverse Bits

The parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Length**> define which bits the function is applied to (see chapter Define Bits for Application of go2ANALYSE Functions on page 23 for details).

Example

The example uses the following bitstream with a circulation length of 4.

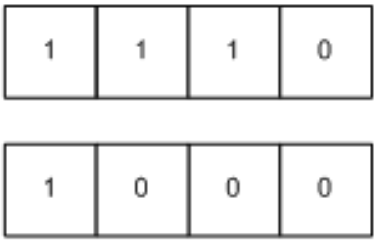


Figure 68: Bitstream before Reversing

Apply the reverse bits function with the parameters shown in Figure 67 (i.e. <**Periodically**> checked, Function period = 4, Row = 1, Column = 0, Length = 4).

Subsequently, the following bitstream is obtained.

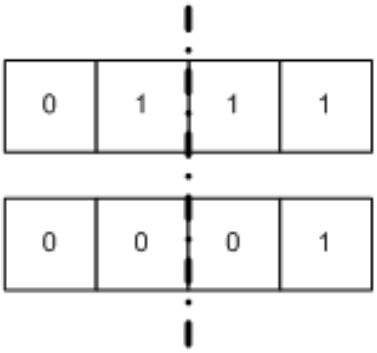


Figure 69: Bitstream after Reversing

Each row has been reversed across the dash-dotted line.

Note: If the parameter <**Length**> is even, all bits in the frame will be reversed, otherwise if <**Length**> is odd, the bit in the center of the frame will remain unaffected by the reverse bits function.

6.5. Clear Tags

Clears either all tags of the chosen color or all tags of any color. Figure 70 is shown on clicking <**Function Kit**><**Manipulation**><**Clear Tags**>.

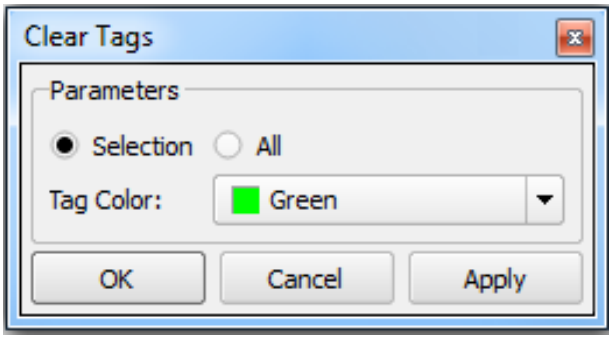


Figure 70: Dialog Box for Clearing Tags

If <**All**> is selected, then all tags regardless of their color are cleared from the bitstream. If <**Selection**> is enabled, only the tags with the color defined by <**Tag Color**> will be cleared.

6.6. Insert and Paste

The bits to be inserted into the current bitstream must be previously copied to the clipboard.

To do so, either highlight the bits in the stream using the mouse, click the right mouse button and select <**Copy**> from the popup menu or copy a number of characters from a text editor to the clipboard to insert them into the bitstream. However, please note that the only characters admissible for this purpose are the ones listed in the table in chapter Open Bitstream on page 18. Copying other characters to the clipboard will have the effect that no bits are inserted.

To insert the copied bits into the stream, select (highlight) the position at which to insert the bits, click the right mouse button and choose <**Insert**>. The bits are inserted into the bitstream, the highlighted bits are not replaced by the ones inserted.

Note: The <**Bit Display**>, as any other text editor, has no cursor. To select the position at which to insert the bits highlight least one bit with the mouse.

When choosing <**Replace**> instead of <**Insert**>, the bits highlighted in the bitstream are replaced with the bits from the clipboard.

7. Logic Features

7.1. Logic Menu

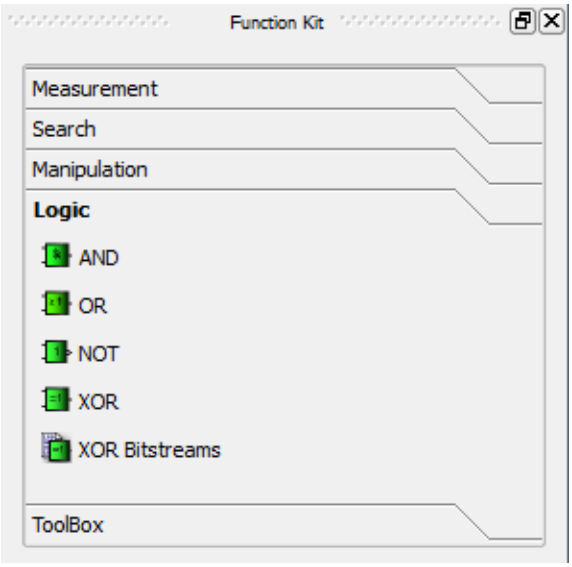


Figure 71: Function Kit - Logic Menu

7.2. AND

<**Function Kit**><**Logic**><**AND**> does a bitwise AND operation of the bits defined by the parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Bits Sequence**> using the entered bit sequence.

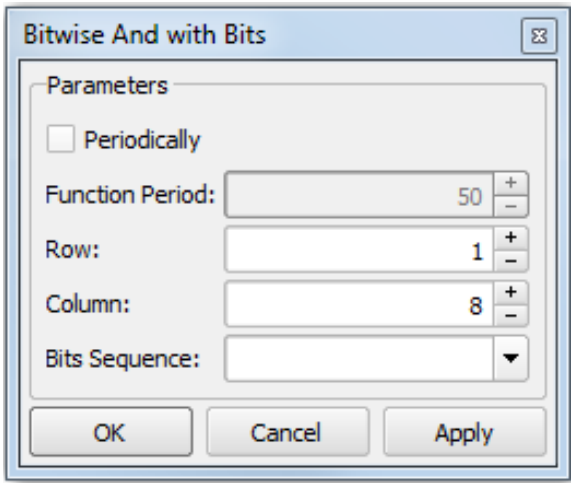


Figure 72: Function Kit - Bitwise And with Bits

The bits are combined by means of Table 23.

Bit of Sequence	Bit of Stream	Result of Operation
0	0	0
0	1	0
1	0	0
1	1	1

Table 23: Logic Table Bitwise AND Function

Example

The example uses a bitstream with two rows and a circulation length of 10 bits.

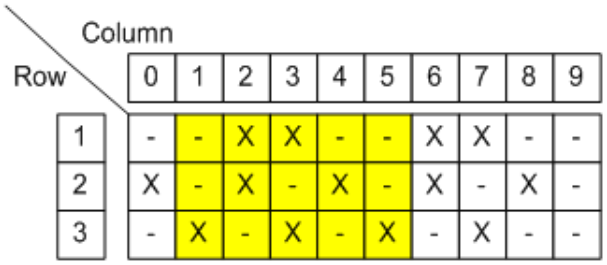


Figure 73: Bitstream before AND Operation, Area of Interest is Highlighted

It is intended to apply the sequence of bits –X-XX to the area of interest highlighted in yellow. To do so, enter the following parameters in the dialog box of the AND function:

- <**Periodically**> Checked
- <**Function Period**> 10
- <**Row**> 1
- <**Column**> 1
- <**Bits Sequence**> –X-XX

Now the AND operation is done periodically every 10 bits, starting in row 1, column 1. The sequence of bits is 5 bits long, so the bits in columns 1, 2, 3, 4 and 5 are affected by the function. On clicking <**OK**> in the dialog box, the bits in all rows have been changed to:

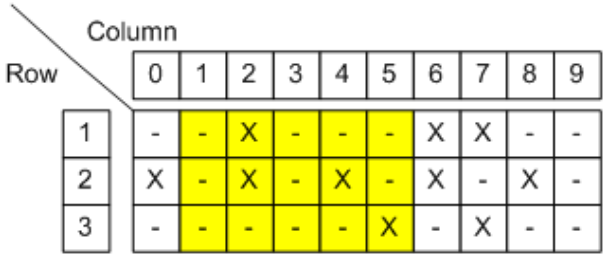


Figure 74: Bitstream after Applied AND Operation

7.3. OR

<**Function Kit**><**Logic**><**OR**> does a bitwise OR operation of the bits which are defined by the parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Bits Sequence**>, using the entered bit sequence.

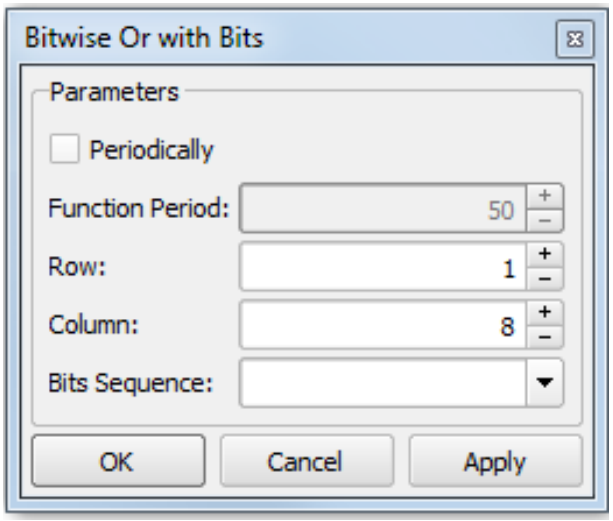


Figure 75: Function Kit - Bitwise Or with Bits

The bits are combined by means of Table 24.

Bit of Sequence	Bit of Stream	Result of Operation
0	0	0
0	1	1
1	0	1
1	1	1

Table 24: Logic Table Bitwise OR Function

Example

The example uses a bitstream with two rows and a circulation length of 10 bits.

Row \ Column	0	1	2	3	4	5	6	7	8	9
1	-	-	X	X	-	-	X	X	-	-
2	X	-	X	-	X	-	X	-	X	-
3	-	X	-	X	-	X	-	X	-	-

Figure 76: Bitstream before OR Operation

It is intended to apply the sequence of bits –X-XX to the area of interest highlighted in yellow. To do so, enter the following parameters in the dialog box of the OR function:

- <**Periodically**> Checked
- <**Function Period**> 10
- <**Row**> 1
- <**Column**> 1
- <**Bits Sequence**> –X-XX

Now the OR operation is done periodically every 10 bits, starting in row 1, column 1. The sequence of bits is 5 bits long, so the bits in columns 1, 2, 3, 4 and 5 are affected by the function. On clicking <**OK**> in the dialog box, the bits in all rows have been changed to:

Row \ Column	0	1	2	3	4	5	6	7	8	9
1	-	-	X	X	X	X	X	X	-	-
2	X	-	X	-	X	X	X	-	X	-
3	-	X	-	X	X	X	-	X	-	-

Figure 77: Bitstream after Applied OR Operation

7.4. NOT

<**Function Kit**><**Logic**><**NOT**> does a bitwise negation of the bits which are defined by the parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Lenght**>.

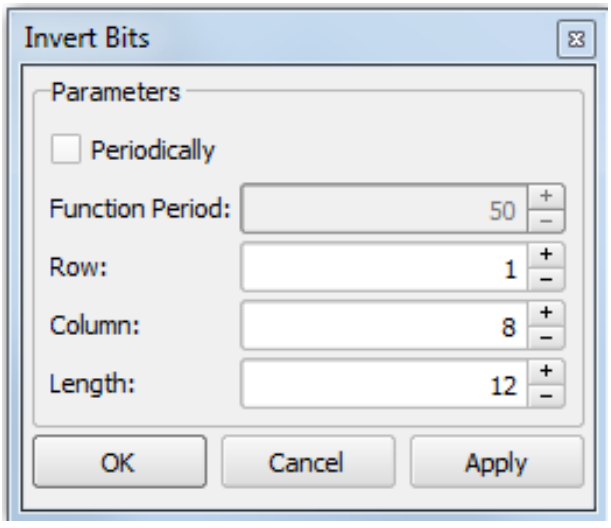


Figure 78: Function Kit - Inverse Bits

The bits are negated by means of Table 25.

Bit of Stream	Result of Operation
0	1
1	0

Table 25: Logic Table Bitwise NOT Function

Example

The example uses a bitstream with two rows and a circulation length of 10 bits.

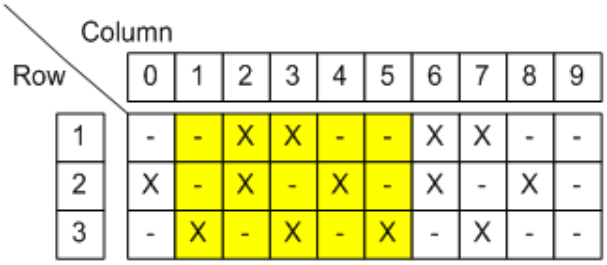


Figure 79: Bitstream before NOT Operation

It is intended to apply the sequence of bits –X-XX to the area of interest highlighted in yellow. To do so, enter the following parameters in the dialog box of the NOT function:

- <**Periodically**> Checked
- <**Function Period**> 10
- <**Row**> 1
- <**Column**> 1
- <**Bits Sequence**> 5

Now the NOT operation is done periodically every 10 bits, starting in row 1, column 1. The sequence of bits is 5 bits long, so the bits in columns 1, 2, 3, 4 and 5 are affected by the function. On clicking <**OK**> in the dialog box, the bits in all rows have been changed to:

Figure 80: Bitstream after Applied NOT Operation

7.5. XOR

<**Function Kit**><**Logic**><**XOR**> does a bitwise Exclusive OR (XOR) operation of the bits defined by the parameters <**Periodically**>, <**Function Period**>, <**Row**>, <**Column**> and <**Bits Sequence**>, using the entered bit sequence.

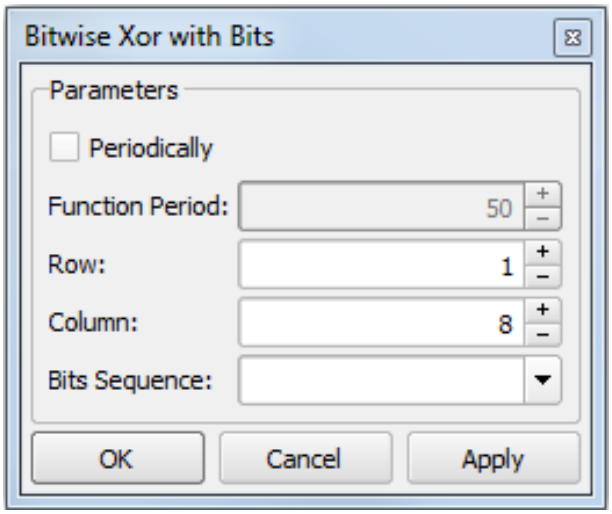


Figure 81: Function Kit - Bitwise Xor with Bits

The bits are combined by means of Table 26.

Bit of Sequence	Bit of Stream	Result of Operation
0	0	0
0	1	1
1	0	1
1	1	0

Table 26: Logic Table Bitwise XOR Function

Example

The example uses a bitstream with two rows and a circulation length of 10 bits.

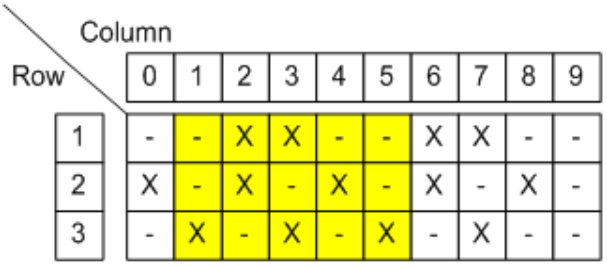


Figure 82: Bitstream before XOR Operation

It is intended to apply the sequence of bits –X-XX to the area of interest highlighted in yellow. To do so, enter the following parameters in the dialog box of the XOR function:

- <**Periodically**> Checked
- <**Function Period**> 10
- <**Row**> 1
- <**Column**> 1
- <**Bits Sequence**> –X-XX

Now the XOR operation is done periodically every 10 bits, starting in row 1, column 1. The sequence of bits is 5 bits long, so the bits in columns 1, 2, 3, 4 and 5 are affected by the function. On clicking <**OK**> in the dialog box, the bits in all rows have been changed to:

Row \ Column	0	1	2	3	4	5	6	7	8	9
1	-	-	-	X	X	X	X	X	-	-
2	X	-	-	-	-	X	X	-	X	-
3	-	X	X	X	X	-	-	X	-	-

Figure 83: Bitstream after Applied XOR Operation

7.6. XOR Bitstream

<**Function Kit**><**Logic**><**XOR Bitstream**> does a bitwise exclusive XOR operation of two complete bit streams available in different bit displays. The input bit streams shall be selected as shown in Figure 84.

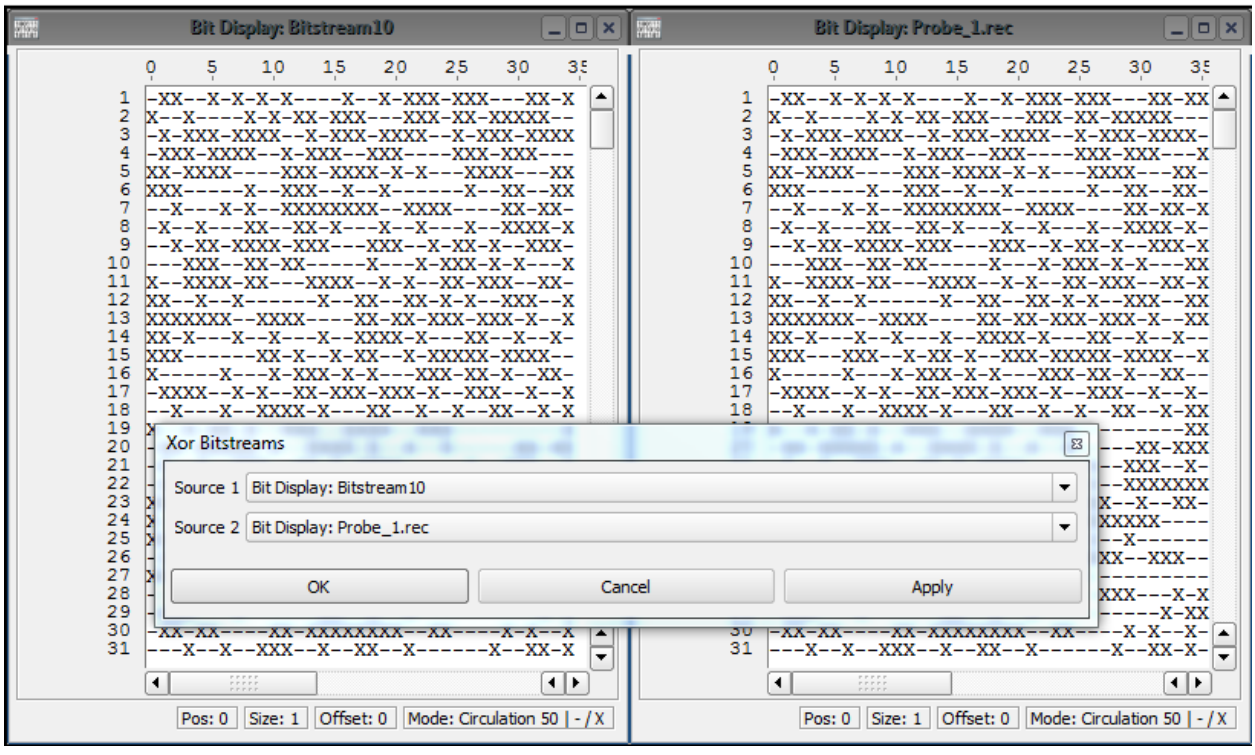


Figure 84: XOR Bitstream dialog

The result will be shown in a new created bit display (see Figure 85).

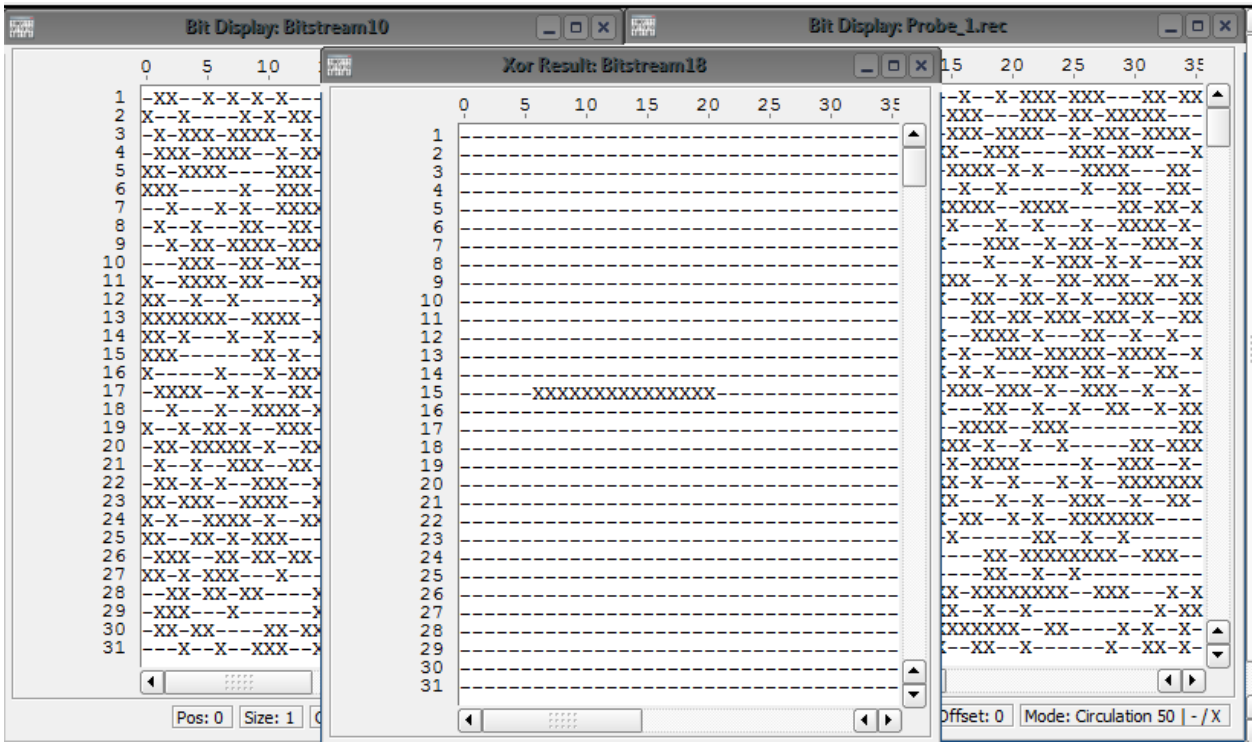


Figure 85: XOR Bitstream result

8. Toolbox Features

8.1. Toolbox Menu

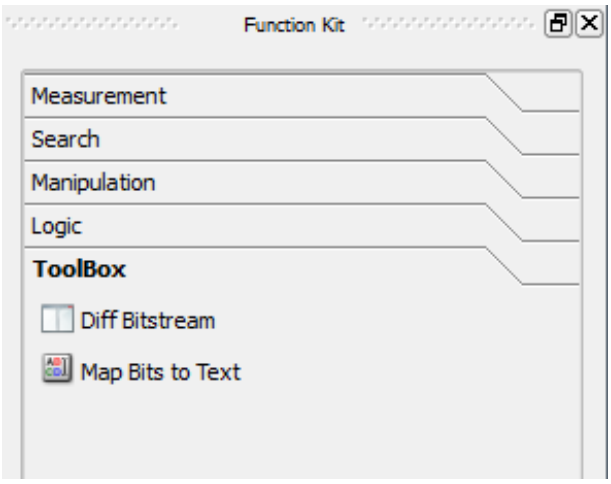


Figure 86: Function Kit - Toolbox Menu

8.2. Diff Bitstream

<**Function Kit**><**Toolbox**><**Diff Bitstream**> does a bitwise Exclusive OR (XOR) operation of maximum two bitstream files. The different bits are marked red.

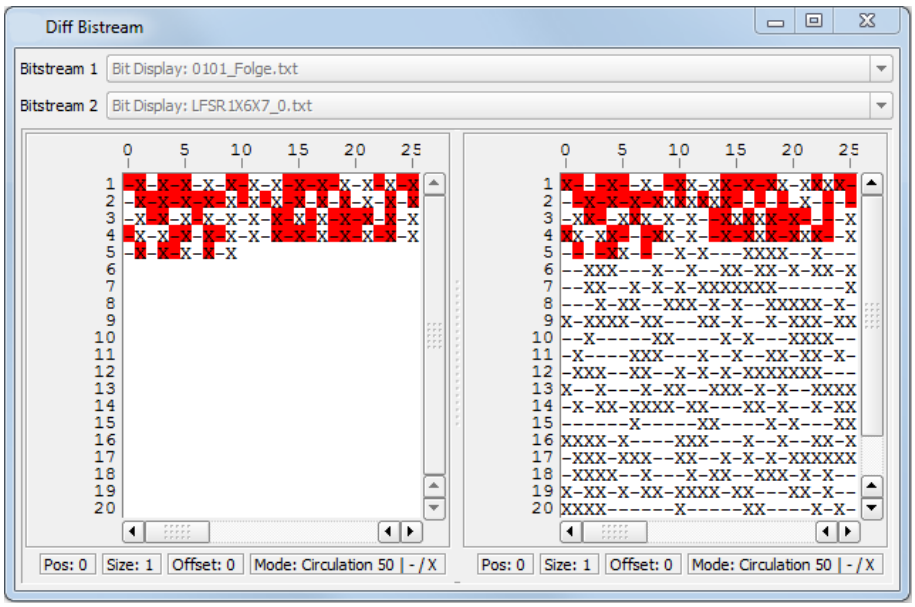


Figure 87: Diff of two Bitstreams

8.3. Map Bits to Text

Bits can be mapped to text manually in go2ANALYSE. Click <**Function Kit**><**Toolbox**><**Map Bits to Text**>. The Text Display is opened showing the text result. The output of the mapping operation varies with the parameter setting on the display's property sheet.

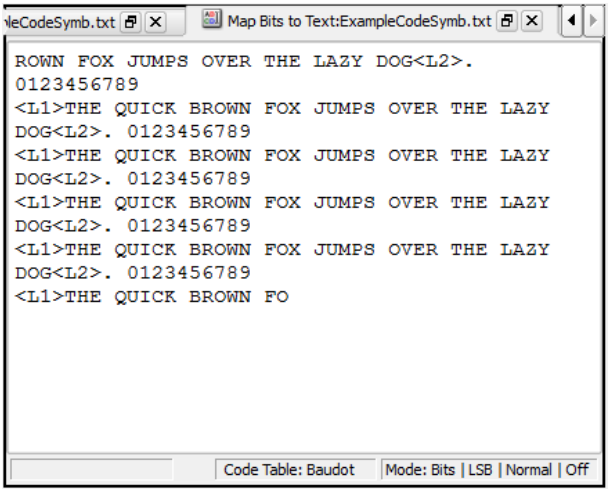


Figure 88: Bit Display Showing Mapping Result

Code tables are created and edited by use of the item <**Extras**><**Configure Code Tables**> (see chapter View and Edit Code Tables on page 100).

9. User Functions

The Decoder Description Language (DDL) is a programming language for the implementation of software decoders. These software decoders convert the data from a bitstream into the output information.

DDL is a simple programming language developed especially for decoding tasks. It includes the individual commands which are processed in the application of a decoder in the appropriate sequence. A compiler translates this text into a code which is interpreted quickly and easily during the runtime of the decoder. Source code and compiled decoders are stored in files.

If you are interested in developing your own decoders then please contact service@procitec.de so that we can provide special tools and trainings.

go2ANALYSE provides the option to apply compiled software decoders to a bitstream. The decoder can supply different output types such as bitstream output, graphic output, marker output, progress bar output and text output, which are displayed with go2ANALYSE. The decoder also has the capability to receive user-defined parameters.

9.1. Use of Decoders

The decoders can be used on record-based bitstreams (see chapter Open Bitstream on page 18). Text-based bitstreams can be decoded with the software decoders as well. To apply a decoder to the current bitstream, select the decoder on <**User Functions**>. A decoder dialog box is displayed with or without defined parameters.

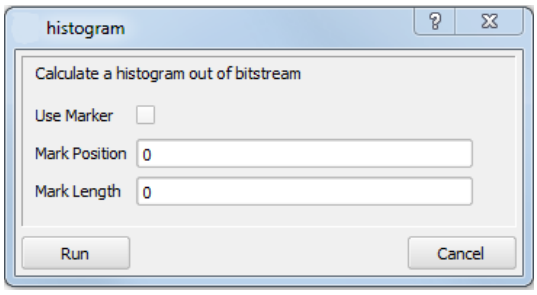


Figure 89: Exemplary Histogram Decoder Dialog Box

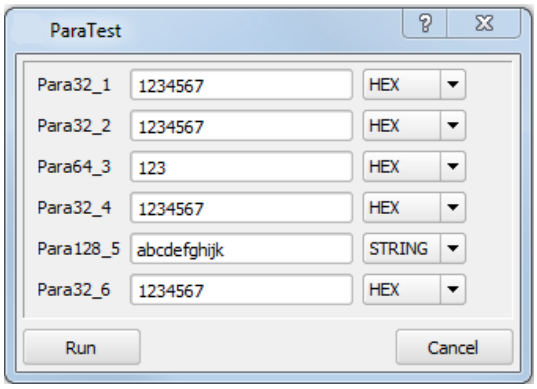


Figure 90: Exemplary ParaTest Decoder Dialog Box

Note: <**User Functions**> cannot be used unless at least one decoder is available in the decoder directory of your go2ANALYSE installation. Having chosen the correct decoder,press <**Run**> to start decoding or analyzing. The <**Cancel**> button closes the dialog box. On successful decoding of the file, the decoder output is shown in go2ANALYSE. If the specific output is not displayed before applying the decoder, go2ANALYSE will open the output, otherwise the content will be overwritten by the new decoder output.

9.2. Configure User Functions

The <**User Functions**> are configured using an XML configuration file.

```
<go2ANALYSE version="19.2.0">
<guielement class="textbox">
<format>hex</format>
<name>Parameter</name>
<value>01234567</value>
<size>1</size>
</guielement>
...
<guielement class="textbox">
<format>string</format>
<name>Search String</name>
<value>abcdefghijk</value>
<size>4</size>
</guielement>
</ go2ANALYSE>
```

The first line must be a root element of configuration:
<go2ANALYSE> or, with optional attribute <go2ANALYSE version="19.2.0">

The next line defines the type of a GUI element:
<guielement class="textbox">

Following types can be defined by assigning to class:

Class	Description
textbox	General Input field for different formats
markstart	Like textbox, but the start position of a marker in an active <**Bit Display**> is automatically set here

Class	Description
marklength	Like textbox, but the length of a marker in an active <**Bit Display**> is automatically set here
checkbox	A check box, which can be just set on off
comment	A text in the head of the dialog box, typically to explain the function

Table 27: GUI Elements

The next child element defines the default format of the text box.
<format>hex</format>

The following formats for text boxes are supported:

Format Definition	Description	Valid Input Characters
dec	Decimal number	0 – 9
hex	Hexadecimal number	0 – 9 | a – f | A – F
oct	Octal number	0 – 7
bin	Binary number	0 | 1
string	ASCII text	ASCII code

Table 28: GUI Text Box Formats

This element defines the name of the parameter:
<name>Para32_1</name>

The next element defines the default value of the parameter:
<value>01234567</value>

The next element defines the length of the parameter in bits (size x 32 bits = parameter length):
<size>4</size>

The next element closes the GUI element definition:
</guielement>

The next line can either be a GUI element definition or the last closing root element.

This element is the last closing root element:
</go2ANALYSE>

The result of this exemplary definition is a dialog box like this:

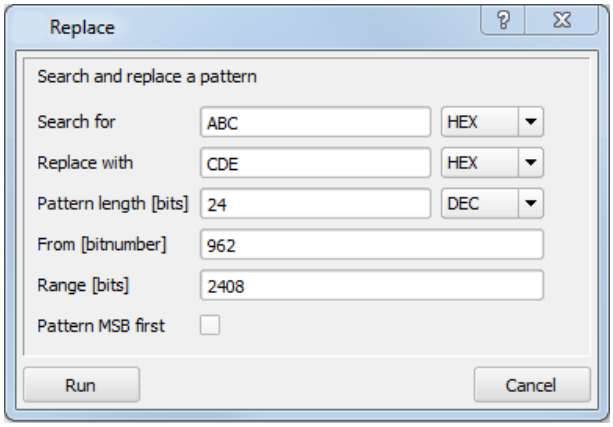


Figure 91: Exemplary Replace Dialog Box

Decoders without any parameters have the following XML definition:
<go2ANALYSE version="19.2.0">
</go2ANALYSE>

9.3. User Functions Language Description

<**User Functions**> can control different output types in go2ANALYSE, and this output feature can also be used with DDL. The DDL function *OutVal* serves to supply a specific output type.

9.3.1. Text Output

The standard text output for <**User Functions**> is the Text Display.

The status bar of the Text Display shows the name of the decoder the current bitstream was decoded with, and *Mode: Text | ASCII*.

The parameters <**Serialisation**>, <**Bitstream**>, <**Code Table**>, <**Force Level**> and <**Start Level**> and the <**Sync Mode**> button are disabled.

The text shown in the Text Display is saved by means of the <**Save Text As...**> button (see chapter Text Display on page 37 for details).

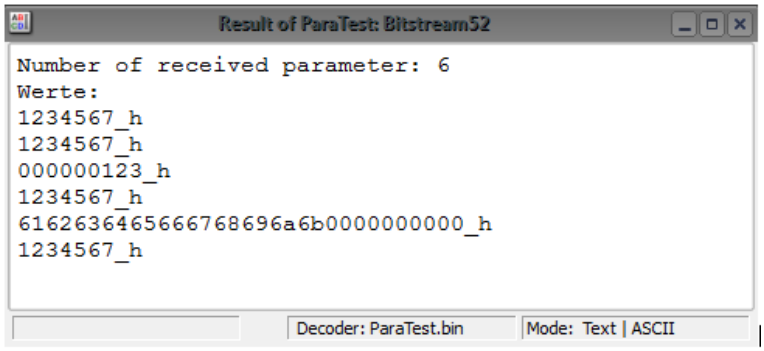


Figure 92: Text Display

9.3.1.1. OutText

OutText (Text, Addr)

Description

Direct output of data package which consists of ASCII symbols (character set defined by target system / output system)

Parameter	Direction	Description	Valid Range
Text	Input	Variable containing 0-terminated ASCII string	Any value
Addr	Input	Output address	All output commands refer to an output address, which is defined by the parameter *Addr*.
			Addr = 0 No output will be transferred. However, all remaining testing, validations etc. are done. *Addr > 0* The output will be directed to the standard output (display, data base etc.). The value of *Addr* is application dependent, e.g. a channel, type of information, or similar. The following values are admissible: • Values 1 to 14: The output will be framed within XML tags <text1> to <text14> • Strings up to 16 characters: The XML tag contains this string Consequently, output addresses e.g. *1* and *text1* have the same result. *Addr = -1* The output is directed to a variable serving as an output buffer. The variable must be defined using SetOutBuf. The output uses string format, i.e. outputs of succeeding calls are appended. On reaching the maximum string length (given by variable length), any further output to this address is stopped until the variable is reset (assigned) to 0.

Table 29: OutText Parameters

9.3.2. Graphic Output

The decoder can plot graphics in one or two dimensions. For 2D, *plot2dx* and *plot2dy* must be used in pairs otherwise *ploty* will be used for 1D.

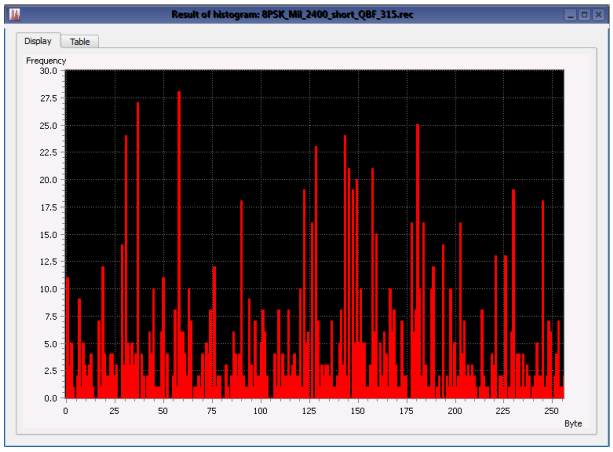


Figure 93: Exemplary Dialog Box

9.3.2.1. graphcolor

OutVal (*(Format, DisplayStyle,* "graphcolor")

Description:

Set display style of graphic output. Values are color, inverse color or monochrome color.

Parameter	Description	Admissible Values
Format	Output format	d decimal h hexadecimal o octal b binary
DisplayStyle	DisplayStyle	0 standard 1 inverse 2 monochrome

Table 30: Graph Color Parameters

9.3.2.2. graphxmin

OutVal *(Format, Xmin,* "graphxmin")

Description:

Set minimum of X-axis.

Parameter	Description	Admissible Values
Format	Output format	d decimal h hexadecimal o octal b binary

Parameter	Description	Admissible Values
Xmin Xmax Ymin Ymax	Minimum of X-axis Maximum of X-axis Minimum of Y-axis Maximum of Y-axis	-2.000.000 to 2.000.000

Table 31: Graph Axis Parameters

9.3.2.3. graphxmax

OutVal *(Format, Xmax,* "graphxmax")

Description:

Set maximum of X-axis.

9.3.2.4. graphymin

OutVal *(Format, Ymin,* "graphymin")

Description:

Set minimum of Y-axis.

9.3.2.5. graphymax

OutVal *(Format, Ymax,* "graphymax")

Description:

Set maximum of Y-axis.

9.3.2.6. graphxunit

OutVal *(Format, AxisUnit,* "graphxunit")

Description:

Set unit of X-axis.

Parameter	Description	Admissible Values
Format	Output format	s string
AxisUnit	Unit of X/Y-axis	Any character

Table 32: Graph Unit Parameters

9.3.2.7. graphyunit

OutVal *(Format, AxisUnit,* "graphyunit")

Description:

Set unit of Y-axis.

9.3.2.8. plot2dx

OutVal *(Format, x, "plot2dx")*

Description:

Set X-coordinate of 2D graphical plot. It is used in pairs with *plot2dy.*

Parameter	Description	Admissible Values
Format	Output format	d decimal h hexadecimal o octal b binary
x y	x-cordinate y-coordinate	0 to 2.000.000

Table 33: Graph Coordinate Parameters

9.3.2.9. plot2dy

OutVal *(Format, y, "plot2dy")*

Description:

Set Y-coordinate of 2D graphical plot. It is used in pairs with *plot2dx.*

9.3.2.10. ploty

OutVal *(Format, y, "ploty")*

Description:

Set Y-coordinate of graphical plot. The X-coordinate is automatically incremented by one.

9.3.3. Bitstream Output

The decoder can output a bitstream with a variable bit block length. The default block length is 32 bits.

9.3.3.1. bitlen

OutVal *(Format, Length, "bitlen")*

Description:

Set length of bit block. The default block length is 32 bits.

Parameter	Description	Admissible Values
Format	Output format	d decimal h hexadecimal o octal b binary
Length	Bit packet length	1 to 65536

Parameter	Description	Admissible Values

Table 34: Bitlen Parameters

9.3.3.2. bit

OutVal (*Format, BitValue,* "bit")

Description:

Output a bit block with a defined length.

Parameter	Description	Admissible Values
Format	Output format	d decimal h hexadecimal o octal b binary
BitValue	Bit block	Any value

Table 35: Bit Parameters

9.3.4. Mark Output

The decoder can highlight single or continuous bits in the bit view.

9.3.4.1. markstart

OutVal (*Format, StartPosition,* "markstart")

Description:

Set start position of highlighting.

Parameter	Description	Admissible Values
Format	Output format	d decimal h hexadecimal o octal b binary
StartPosition EndPosition	Bit position to start highlighting Bit position to stop highlighting	Any value

Table 36: Mark Position Parameters

9.3.4.2. markend

OutVal (*Format, EndPosition,* "markend")

Description:

Set end position of highlighting.

9.3.4.3. markcolor

OutVal (*Format*, *Color*, "markcolor")

Description:

Set highlighting color from go2ANALYSE color table.

Parameter	Description	Admissible Values
Format	Output format	d decimal h hexadecimal o octal b binary
Color	Highlighting color	Any value

Table 37: Mark Color Parameters

9.3.5. Progress Bar Output

The decoder can open and control a progress bar.

9.3.5.1. progress

OutVal *(Format, Value,* "progress")

Description:

Set progress bar position between 0 and 100.

Parameter	Description	Admissible Values
Format	Output format	d decimal h hexadecimal o octal b binary
Value	Progress bar position	0 to 100

Table 38: Progress Parameters

9.4. Implemented User Functions

The following user functions have been implemented and shall serve as exemplary models for customer solutions. Some of these functions use additional bit stream attributes, which are only present if files of format "*.rec" are used for input, i.e. no "*.txt" files and no intermediate results.

9.4.1. User Functions Menu

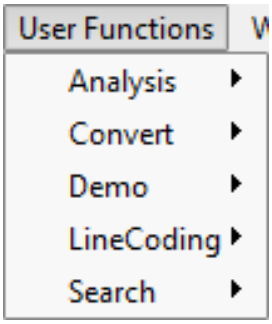


Figure 94: User Functions Menu

The menu bar will offer the following groups of user functions:

User Function Group	Description
<**Analysis**>	General functions to analyze the bit stream
<**Convert**>	General functions to modify the bit stream
<**Demo**>	Applied examples of various additional DDL user function commands
<**LineCoding**>	Function to recover line codings
<**Search**>	Function to search for specific bit stream characteristics

Table 39: User Functions Overview

9.4.2. Analysis Functions

9.4.2.1. PoynomialCheck

Call: <**User Functions**><**Analysis**><**PolynomialCheck**>

Description:

This function is helpful when searching of a bit sequence in the bitstream generated by a linear feedback shift register (LFSR) with the generator polynomial (p0 + p1*z-1 + p2*z-2 + ….. pn*z-n). The function uses the DDL-command "SearchPolynom" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Polynomial [binary]**>	See DDL Command "SearchPolynom" parameter *Polynom*
<**Sequence length [bits]**>	See DDL Command "SearchPolynom" parameter *Len*
<**Tolerance [bits]**>	See DDL Command "SearchPolynom" parameter *FaultMax*
<**MARK range start only**>	Mark bit position of range start

Table 40: User Function <PolynomialCheck> Parameters

Result <**Bit Display**>: Found LFSR sequences and/or sequence starts will be marked with two different colors.

9.4.2.2. ShowQuality

Call: <**User Functions**><**Analysis**><**ShowQuality**>

Description:

Shows the mean symbol quality of a marked bit range. The symbol qualities are delivered by the APC demodulator for each symbol ranging from 0 to 100. The range parameters can be set by bit display marker or can be annotated directly.

Parameter	Description
<**Start position**>	Bit position of range start
<**Range [bits]**>	Range length in number of bits

Table 41: User Function <ShowQuality> Parameters

Result Text Display: "Mean quality of marked symbols: Q" (Q = 0 . . . 1900)

Restrictions: Applicable for unmodified bit streams only. The input file must be of type "*.rec".

9.4.2.3. ShowTimeRange

Call: <**User Functions**><**Analysis**><**ShowTimeRange**>

Description:

Shows the start time and the time duration of a marked bit range. Time information is delivered by the APC demodulator. The range parameters can be set by bit display marker or can be annotated directly.

Parameter	Description
<**Start position**>	Bit position of range start
<**Range [bits]**>	Range length in number of bits

Table 42: User Function <ShowTimeRange> Parameters

Result Text Display:
"Range Start: hours : minutes : seconds : milliseconds"
"Range Length: MMM. ms " (MMM = range in milliseconds)

Restrictions: Applicable for unmodified bit streams only. The input file must be of type "*.rec".

9.4.2.4. ViterbiCorrection

Call: <**User Functions**><**Analysis**><**Viterbi Correction**>

Description:

Applies a Viterbi correction for of a marked bit range. The range parameters can be set by bit display marker or can be annotated directly. The function uses the DDL command "Correct Viterbi" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Frame start**>	Bit position of range start
<**Frame length [bits]**>	Range length in number of bits
<**Polynomial 1**>	See DDL Command "CorrectViterbi" parameter *Poly1*
<**Polynomial 2**>	See DDL Command "CorrectViterbi" parameter *Poly2*
<**Polynomial 3**>	See DDL Command "CorrectViterbi" parameter *Poly3*
<**Polynomial 4**>	See DDL Command "CorrectViterbi" parameter *Poly4*
<**Rate**>	See DDL Command "CorrectViterbi" parameter *Rate*
<**Constraint length**>	See DDL Command "CorrectViterbi" parameter *RegLen*
<**Depuncturing mask**>	See DDL Command "CorrectViterbi" parameter *DePunctMask*
<**Depunct. mask length**>	See DDL Command "CorrectViterbi" parameter *MaskLen*
<**Register start value**>	See DDL Command "CorrectViterbi" parameter StartVal

Table 43: User Function <ViterbiCorrection> Parameters

Result <**Bit Display**>: Corrected bit stream, corresponding to DDL Command "CorrectViterbi" result *DatOut*.

9.4.3. Convert Functions

9.4.3.1. ChannelSelect

Call: <**User Functions**><**Convert**><**ChannelSelect**>

Description:

Single channels can be extracted from a demodulator output. The selected channels are designated by a bit pattern. Each bit represents a channel symbol: 1/0 = selected / omitted. Annotation LSB-first: ... Ch3,Ch2,Ch1,Ch0. This functions applies the DDL command "PPExtractChannels" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Channel pattern**>	Set a bit for each channel to be selected

Table 44: User Function <ChannelSelect> Parameters

Result <**Bit Display**>: Bit stream containing selected channels only.

Restrictions: Applicable for unmodified bit streams only. The input file must be of type "*.rec".

9.4.3.2. DeInterleaving

Call: <**User Functions**><**Convert**><**Deinterleaving**>

Description:

Applies bit-deinterleaving on a marked bitstream section. The function uses the DDL-command "ExtractInterlLong" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Start position**>	Bit position of range start
<**Block length [bits]**>	Range length in number of bits
<**Character length [bits]**>	See DDL Command "ExtractInterlLong" parameter *CharSize*
<**Bit distance [bits]**>	See DDL Command "ExtractInterlLong" parameter *BitDist*
<**Character distance [bits]**>	See DDL Command "ExtractInterlLong" parameter *CharDist*
<**Modulo block size**>	See DDL Command "ExtractInterlLong" parameter *BlockLen*
<**No of characters**>	Number of characters to be interleaved

Table 45: User Function <DeInterleaving> Parameters

Result <**Bit Display**>: Bit stream containing deinterleaved characters.

9.4.3.3. DeStuff

Call: <**User Functions**><**Convert**><**DeStuff**>

Description:

Applies de-stuffing on a marked bitstream section. The function uses the DDL-command "Destuff" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Frame start**>	Bit position of range start
<**Frame length [bits]**>	Range length in number of bits
<**Stuff length [bits]**>	See DDL Command "ExtractInterlLong" parameter *ChainLim*
<**Sequence polarity**>	See DDL Command "ExtractInterlLong" parameter ChainPol

Table 46: User Function <DeStuff> Parameters

Table 38: User Function <DeStuff> Parameters

Result <**Bit Display**>: Bit stream containing de-stuffed characters.

9.4.3.4. Descramble

Call: <**User Functions**><**Convert**><**Descramble**>

Description:

Performs a self-synchronized descrambling of a marked bitstream section. The function uses the DDL-command "PPDescramble" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Start position**>	Bit position of range start
<**Range [bits]**>	Range length in number of bits
<**Polynomial [binary]**>	See DDL Command "PPDescramble" parameter *Polynom*)

Table 47: User Function <Descramble> Parameters

Result <**Bit Display**>: Bit stream containing descrambled bitstream.

9.4.3.5. Extract

Call: <**User Functions**><**Convert**><**Extract**>

Description:

Extracts a marked section.

Parameter	Description
<**Frame start**>	Bit position of range start
<**Frame length [bits]**>	Range length in number of bits

Table 48: User Function <Extract> Parameters

Result <**Bit Display**>: Extracted bit stream.

9.4.3.6. RotateLeft

Call: <**User Functions**><**Convert**><**RotateLeft**>

Description:

Left-rotates a marked section within a bit stream.

Parameter	Description
<**Start Position**>	Start position of section to rotate
<**Range [bits]**>	Length of the section to rotate
<**Number of Rotations**>	Number of bit rotations

Table 49: User Function <RotateLeft> Parameters

Result <**Bit Display**>: Complete bit stream including rotated section.

9.4.3.7. RotateRight

Call: <**User Functions**><**Convert**><**RotateRight**>

Description:

Right-rotates a marked section within a bit stream.

Parameter	Description
<**Start position**>	Start position of section to rotate
<**Range [bits]**>	Length of the section to rotate
<**Number of rotations**>	Number of bit rotations

Table 50: User Function <RotateRight> Parameters

Result <**Bit Display**>: Complete bit stream including rotated section.

9.4.3.8. Symbol2Convert

Call: <**User Functions**><**Convert**><**Symbol2Convert**>

Description:

Converts demodulated symbols of length 2 bits (e.g. PSK4) within a selected range. Bits outside the marked sections remain unchanged.

Parameter	Description
<**MSB first**>	This checkbox set will interpret the values MSB-first otherwise the interpretation is LSB-first
<**Symbol 00**>	Replacement for symbol 00
<**Symbol 01**>	Replacement for symbol 01
<**Symbol 10**>	Replacement for symbol 10
<**Symbol 11**>	Replacement for symbol 11
<**From [bitnumber]**>	Range start position
<**Range [bits]**>	Range length

Table 51: User Function <Symbol2Convert> Parameters

Result <**Bit Display**>: Complete bit stream including converted section.

9.4.3.9. Symbol3Convert

Call: <**User Functions**><**Convert**><**Symbol3Convert**>

Description:

Converts demodulated symbols of length 3 bits (e.g. PSK8) within a selected range. Bits outside the marked sections remain unchanged.

Parameter	Description
<**MSB first**>	This checkbox set will interpret the values MSB-first otherwise the interpretation is LSB-first
<**Symbol 000**>	Replacement for symbol 000
<**Symbol 001**>	Replacement for symbol 001

Parameter	Description
<**Symbol 010**>	Replacement for symbol 010
<**Symbol 011**>	Replacement for symbol 011
<**Symbol 100**>	Replacement for symbol 100
<**Symbol 101**>	Replacement for symbol 101
<**Symbol 110**>	Replacement for symbol 110
<**Symbol 111**>	Replacement for symbol 111
<**From [bitnumber]**>	Range start position
<**Range [bits]**>	Range length

Table 52: User Function <Symbol3Convert> Parameters

Result <**Bit Display**>: Complete bit stream including converted section.

9.4.3.10. SymbolBitReversal

Call: <**User Functions**><**Convert**><**SymbolBitReversal**>

Description:

Applies bit reversal for all input symbols. Symbol length and limits are delivered by the APC demodulator and must not be specified.

No parameter.

Result <**Bit Display**>: Complete bit stream with bit reversed symbols.

Restrictions: Applicable for unmodified bit streams only. The input file must be of type "*.rec".

9.4.4. Demo Functions

Functions under <**User Functions**><**Demo**> shall serve just as example to demonstrate the use of all types of language elements described in chapter User Functions Language Description on page 78.

9.4.5. LineCoding Functions

9.4.5.1. BIPH

Call: <**User Functions**><**LineCoding**><**BIPH**>

Description:

Performs a reconstruction of biphase line coding. The function uses the DDL-command "PPBitCodeBIPH" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Invert**>	This checkbox set will perform a BIPH-S reconstruction, otherwise it will be BIPH-M. This corresponds to the parameter *Mode* in DDL command *PPBitCodeBIPH*

Parameter	Description

Table 53: User Function <BIPH> Parameters

Result <**Bit Display**>: Reconstructed bit stream.

9.4.5.2. Manchester

Call: <**User Functions**><**LineCoding**><**Manchester**>

Description:

Performs a reconstruction of Manchester line coding the function uses the DDL-command "PPBitCode-Manch" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Invert**>	This checkbox set will invert the input before applying the function

Table 54: User Function <Manchester> Parameters

Result <**Bit Display**>: Reconstructed bit stream.

9.4.5.3. NRZ

Call: <**User Functions**><**LineCoding**><**NRZ**>

Description:

Performs a reconstruction of NRZ line coding The function uses the DDL-command "PPBitCodeNRZ" as described in the DDL manual or under the Decoder-Editors "Help".

Parameter	Description
<**Invert**>	This checkbox set will perform a NRZ-M reconstruction, otherwise it will be NRZ-S. This corresponds to the parameter *Mode* in DDL command PPBitCo-deNRZ

Table 55: User Function <NRZ> Parameters

Result <**Bit Display**>: Reconstructed bit stream.

9.4.6. Search Functions

9.4.6.1. Replace

Call: <**User Functions**><**Search**><**Replace**>

Description:

Replaces a specified pattern with another pattern.

Parameter	Description
<**Search for**>	Bit pattern to Search for
<**Replace with**>	Bit pattern to replace with
<**Pattern length [bits]**>	Length of the bit pattern to search for
<**From [bitnumber]**>	Range start position
<**Range [bits]**>	Range length
<**Pattern MSB first**>	This checkbox set will interpret the pattern MSB-first otherwise the interpretation is LSB-first

Table 56: User Function <Replace> Parameters

Result <**Bit Display**>: Complete bit stream with bit replaced patterns.

Result Text Display: Number of pattern found.

9.4.6.2. Search_4Phase

Call: <**User Functions**><**Search**><**Search4Phase**>

Description:

When applying an absolute PSK4 demodulation, the absolute phase might be ambiguous, as there are 4 possible variants. This function searches for a pattern, trying all 4 variants. The symbol table is supposed to be a "natural coding", i.e.: 0°: 00 / 90°: 01 / 180°: 10 / 270°: 11. Other symbol configurations must be set via the functions DDL source code.

Parameter	Description
<**Pattern**>	Pattern to search for
<**Length [bits]**>	Length of the pattern in bits
<**Tolerance [bits]**>	Number of faulty bits to be tolerated
<**Pattern MSB first**>	This checkbox set will interpret the pattern MSB-first otherwise the interpretation is LSB-first

Table 57: User Function <Search_4Phase> Parameters

Result <**Bit Display**>: Found patterns will be marked. The mark color corresponds to phase shift.

9.4.6.3. Search_8Phase

Call: <**User Functions**><**Search**><**Search8Phase**>

Description:

When applying an absolute PSK8 demodulation, the absolute phase might be ambiguous, as there are 8 possible variants. This function searches for a pattern, trying all 8 variants. The symbol table is supposed to be a "natural coding", i.e.: 0°: 000 / 45°: 001 / 90°: 010 / 135°: 011 / 180°: 100 / 255°: 101 / 270°: 110 / 315°: 111. Other symbol configurations must be set via the functions DDL source code.

Parameter	Description
<**Pattern**>	Pattern to search for
<**Length [bits]**>	Length of the pattern in bits
<**Tolerance [bits]**>	Number of faulty bits to be tolerated
<**Pattern MSB first**>	This checkbox set will interpret the pattern MSB-first otherwise the interpretation is LSB-first

Table 58: User Function <Search_8Phase> Parameters

Result <**Bit Display**>: Found patterns will be marked. The mark color corresponds to phase shift.

9.4.6.4. Search_InterlVal

Call: <**User Functions**><**Search**><**Search_InterlVal**>

Description:

Searches an interleaved character. The input comprises the non-interleaved character and the interleaving parameters. The found interleaved bits will be marked.

Parameter	Description
<**Pattern MSB first**>	This checkbox set will interpret the pattern MSB-first otherwise the interpretation is LSB-first
<**Character size [bits]**>	Size of the character in bits.
<**Bit distance [bits]**>	The distance (in bits) of two succeeding bits of a character after interleaving.
<**Character distance [bits]**>	The distance of the first bit of a character to the first bit of the next character after interleaving
<**Tolerance [bits]**>	The number of faulty bits to be tolerated
<**Mark color [index]**>	Numerical index of the mark color

Table 59: User Function <Search_InterlVal> Parameters

Result <**Bit Display**>: Found patterns will be marked.

9.4.6.5. Search_Val

Call: <**User Functions**><**Search**><**Search_Val**>

Description:

Searches a pattern given in any base annotation (hexadecimal, decimal etc.). Found patterns will be marked.

Parameter	Description
<**Include overlapping patterns**>	This checkbox set will find overlapping patterns as well. Otherwise only isolated patterns shall be searched.

Parameter	Description
<**Pattern MSB first**>	This checkbox set will interpret the pattern MSB-first otherwise the interpretation is LSB-first
<**Pattern**>	The pattern to search (any base annotation)
<**Pattern length [bits]**>	Size of one the searched pattern in bits.
<**Tolerance [bits]**>	The number of faulty bits to be tolerated
<**Mark color [index]**>	Numerical index of the mark color

Table 60: User Function <Search_Val> Parameters

Result <**Bit Display**>: Found patterns will be marked.

9.4.6.6. ShowBurstStart

Call: <**User Functions**><**Search**><**ShowBurstStart**>

Description:

APC Demodultors working with burst detectors (“burst mode”) supply informations about the burst start. So within this bitstreams the first bit of a burst can be identified. This function marks the first bit of each burst in the Bit Display.

No parameter.

Result <**Bit Display**>: Complete bit stream with bit marked burst starts.

Restrictions: Applicable for unmodified bit streams only. The input file must be of type “*.rec”.

9.4.6.7. ShowInterleaving

Call: <**User Functions**><**Search**><**ShowInterleaving**>

Description:

The function shows the position of the bits of one character, bit-interleaved over the bit stream.

Parameter	Description
<**Character length [bits]**>	Length of one character in bits
<**Bit distance [bits]**>	The distance (in bits) of two succeeding bits of a character after interleaving. See DDL Command “ExtractInterlLong” parameter *BitDist.*
<**Character distance [bits]**>	The distance of the first bit of a character to the first bit of the next character after interleaving.
<**Modulo block size**>	The block size in case of a modulo-block-interleaving. If no modulo interleaving is performed, this value should be at least the complete interleaving range.
<**Character Idx to show**>	Index of character to show (0, 1, 2 ... = first, second, third ... character)

Table 61: User Function <ShowInterleaving> Parameters

Result <**Bit Display**>: Bit stream with marked character bits.

9.4.6.8. ShowSymbolStart

Call: <**User Functions**><**Search**><**ShowBurstStart**>

Description:

APC Demodulators supply the symbol boundaries for all modulation orders across the complete bit stream. This function marks the first bit of each demodulation symbol in <**Bit Display**>.

No parameter.

Result <**Bit Display**>: Complete bit stream with bit marked burst starts.

Restrictions: Applicable for unmodified bit streams only. The input file must be of type "*.rec".

9.4.6.9. ZipSignatures

Call: <**User Functions**><**Search**><**ZipSignatures**>

Description:

This function shall serve as an example to identify and mark special signatures (patterns) of a known file format. In this case several known patterns of ZIP-files will be marked.

No parameter.

Result <**Bit Display**>: Complete bit stream with marked ZIP-specific pattern.

Result Text Display: Position and description of signatures found.

10. Function Workflow

All analyzing steps applied to the current bitstream are logged by go2ANALYSE in a command history, as it is called.

go2ANALYSE can show the history of the workflow for the current modification steps, replay (repeat) all steps logged in the workflow history, and load and save the workflow history in a file.

10.1. View Function Workflow for Current Bitstream

Each analyzing step applied to the current bitstream is logged by go2ANALYSE. All logged steps are viewed in <**Function Workflow**><**History**> dock window.

Example

Open the file *Numb0-15asBits.txt*. Change the circulation length to 64. Apply the following go2ANALYSE functions:

- Tag the first four bits in row 1 with red color
- Autocorrelate the file (<**Autocorrelation**> and <**OK**>)
- Apply OR to the first four bits in row 3 with this pattern: X-X-
- Tag the first four bits in row 18 with blue color

Now the window in the <**Function Workflow**><**History**> should look as shown in Figure 95.

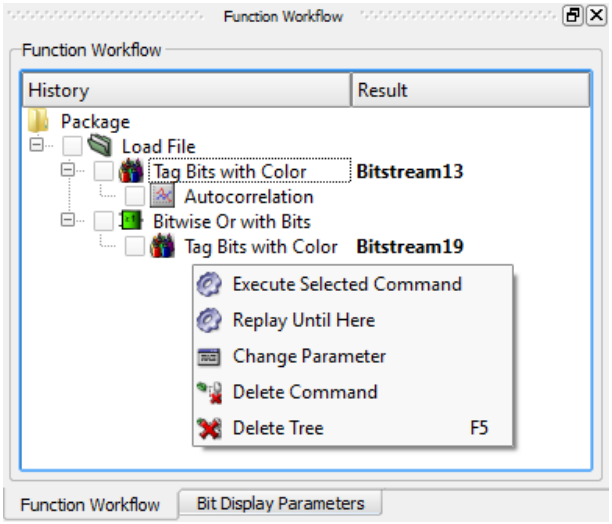


Figure 95: Function Workflow History

10.2. Replay Analysis Steps to File

It is possible to replay the current history partly or completely via the right mouse button context menu. In the same way parameters of each step can be changed before replaying.

10.3. Load and Save Workflow History

<**Function Workflow**><**History**> can be saved and loaded under <**File**>.

11. Code Tables

Code tables are used to convert bits of a bitstream into readable output. They consist of a code that pairs a set of characters (or code symbols) with a set of bits. Common examples include the code table for Morse, Baudot and ASCII, which encodes letters, numerals, and other symbols as 7-bit binary set of bits.

go2ANALYSE provides the option to create the code tables required to analyze bitstreams, to modify existing code tables or simply view the definition of available code tables.

All code table operations are controlled by means of the code table dialog which opens on selecting <**Extras**><**Configure Code Tables**> or the shortcut <**Alt**>+<**F7**>.

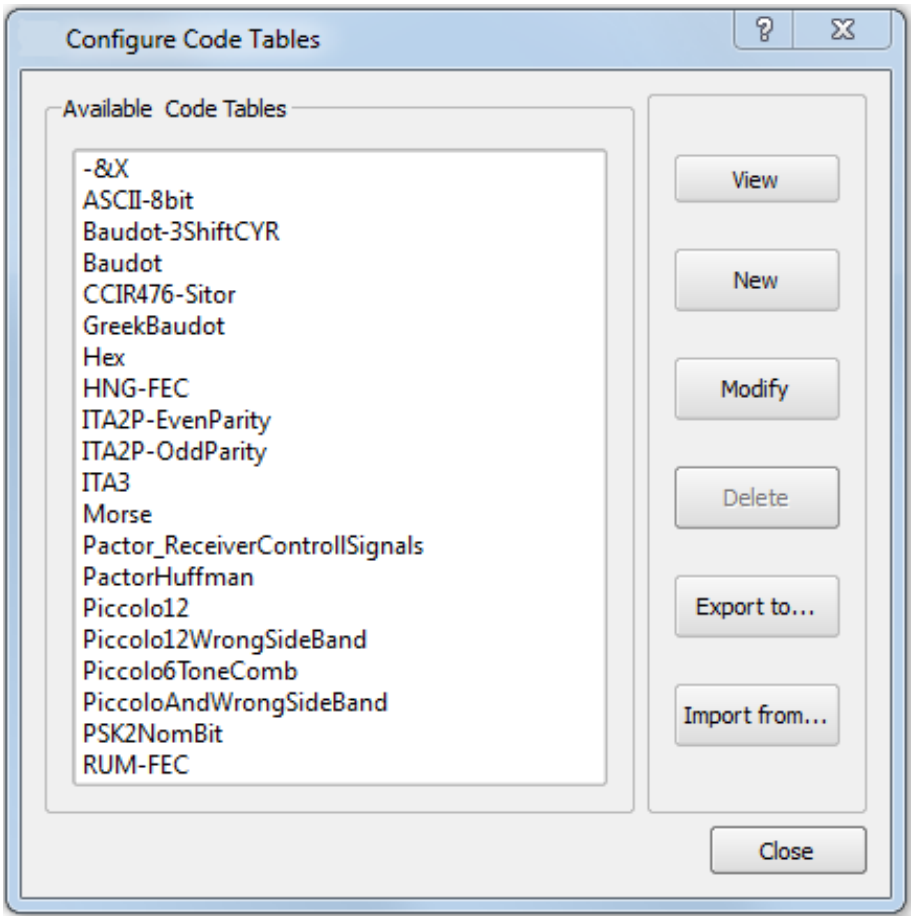


Figure 96: Configure Code Tables Dialog Box

11.1. View and Edit Code Tables

11.1.1. View Code Table

The code tables listed are displayed by selecting the desired table and pressing <**View**>. The following dialog is displayed.

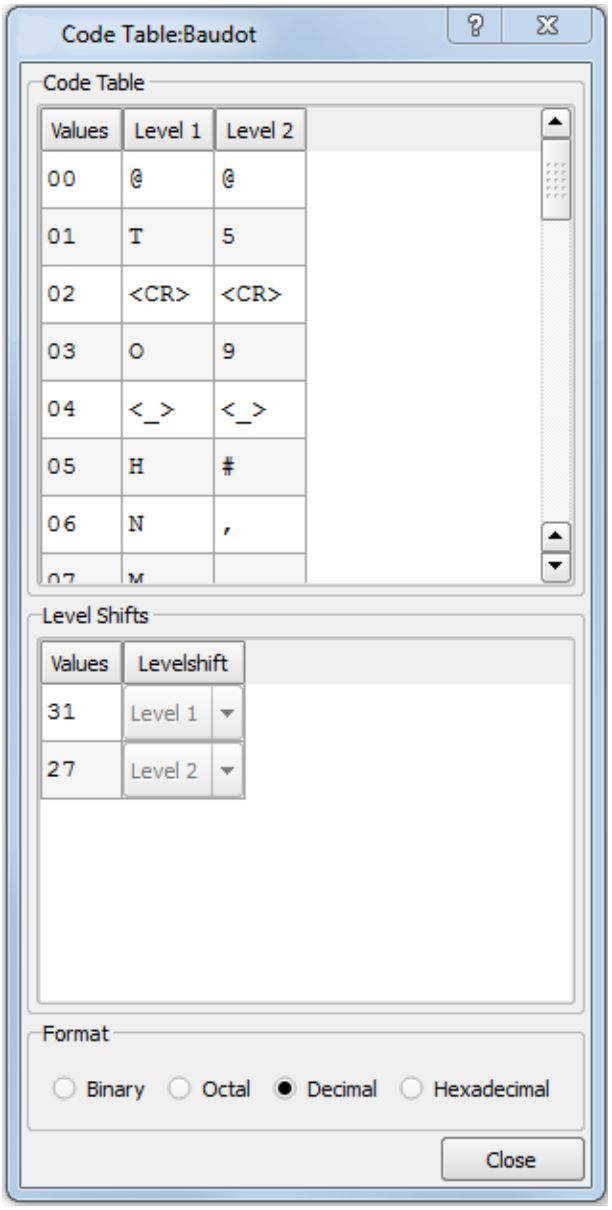


Figure 97: Viewing the Baudot Code Table

Code Table

The group box *Code Table* shows the details of the selected code table. As stated above, code tables consist of a set of bits which is paired to a set of characters.

The set of bits belonging to one set of characters is shown as an integer value in the column "Values".

The columns *Level 1* and *Level 2* show the set of characters belonging to the bits.

For each set of bits defined (shown as decimal values in this example), at least one set of characters has been defined.

The following reserved character sequences have a predefined function:

Reserved Character	Defined Function
<>	No output

Reserved Character	Defined Function
<_>	Space character
<**LF**>	Line feed
<**CR**>	Carriage return
<**L1**>	Switch to output level 1
<**L2**>	Switch to output level 2
<**LN**>	Switch to output level 3
{value}	Unicode value of symbol (0-65535)
{44}	comma
{59}	semicolon

Table 62: Code Table Reserved Characters

Note: If a code table is used for converting bits to text, then every time a symbol value is found which is paired with one of the reserved character sequences, this special symbol is appended to the text.

Level Shifts

Some code tables have more than one defined output level (code table subset). In each subset, all sets of bits defined in this code table have a set of characters assigned.

There must be at least one special symbol, i.e. a level shift as it is called, which switches from one output level to another. The values assigned to these level shifts are displayed in the group box *Level Shifts*. This group box is only shown if the code table currently displayed has more than one level defined.

Format

Changing the options in the group box *Format* enables to view the values either in decimal, hexadecimal, octal or binary annotation.

When selecting <**Binary**>, each value shows the exact set of bits assigned to this value.

11.1.2. Edit Existing Code Table

To modify an existing code table, select the respective code table from the list shown by selecting <**Extras**> <**Configure Code Tables**> and press <**Modify**>.

Existing code tables cannot be edited on pressing <**View**>.

The dialog box in Figure 98 is displayed (with ASCII-8bit code table selected).

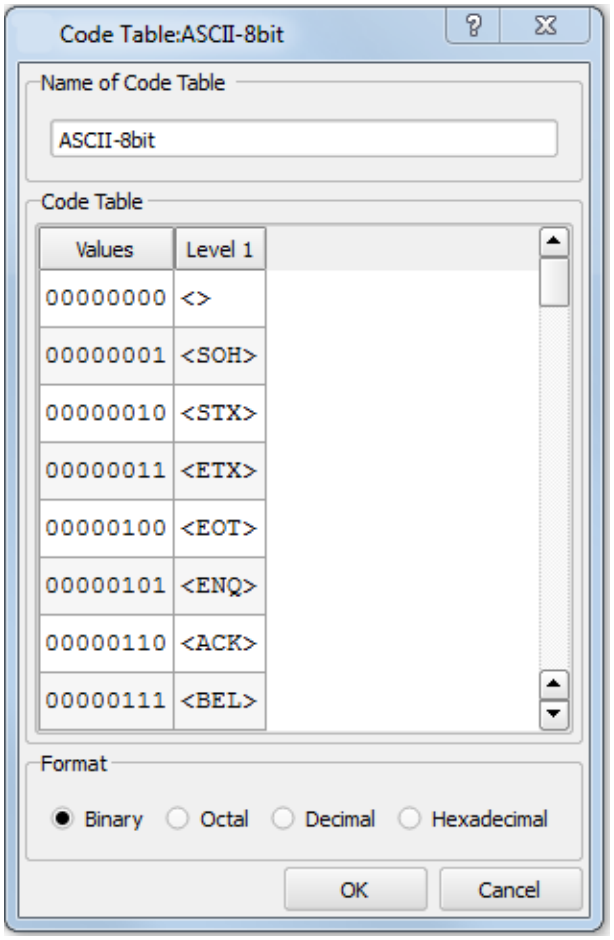


Figure 98: Modifying the ASCII-8bit Code Table

11.1.2.1. Rename Code Table

To rename the code table, edit the entry in the text box in the group box *Name of Code Table*.

Note: When editing the name of the current code table, the new name will be applied to this code table on activating the <**OK**> button. The code table file will be renamed as well. The file with the code table prior to renaming will be deleted.

11.1.2.2. Change Values and Characters

To modify any value simply double click the value to change and enter the value.

Press <**Return**> after editing the value. go2ANALYSE will verify the entered string and show an alert message:

- If the entered string is not a valid value
- If the same value is currently used in this code table in another row

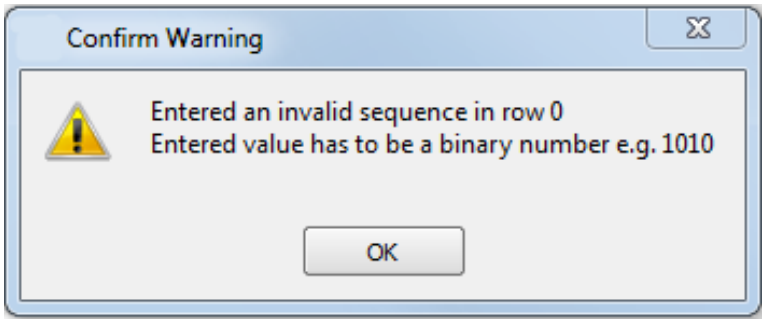


Figure 99: Example Alert Message - Change Values

go2ANALYSE will restrict the entered value to the maximum admissible value for this code table if the code table has a fixed number of bits per symbol. Otherwise the entered values are not restricted to a maximum value.

Example

Assume an 8-bit ASCII code table. With 8 bits, one can define 2^8 -1 = 255 code symbols.

When entering a value of 256 or higher, go2ANALYSE will restrict the entered value to 255. Once the entered value has been restricted to 255, go2ANALYSE verifies if this value is already assigned to another code symbol. If so, go2ANALYSE will display an alert message saying this value is used more than once.

To change a character set (symbol) of the code table, select the symbol to change and enter the characters using the keyboard. Mind the special characters mentioned in chapter View Code Table on page 100.

11.1.2.3. Insert Non-Latin Characters Into Code Tables

Some cases may require inserting characters from non-Latin alphabets such as Hebrew, Cyrillic or Arabic into a code table. If you have a need for such characters in a code table, enter them by means of a special dialog which provides the option to choose any character from the Unicode character table.

Select the cell in which to insert the special character and click the right mouse button. A popup menu appears (see Figure 100).

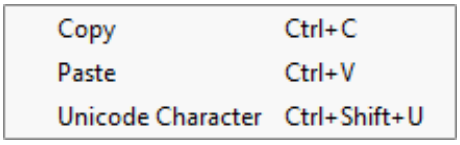


Figure 100: Configure Tables - Choose Unicode Character

On selecting the item <**Unicode Character**>, the following dialog box is show.

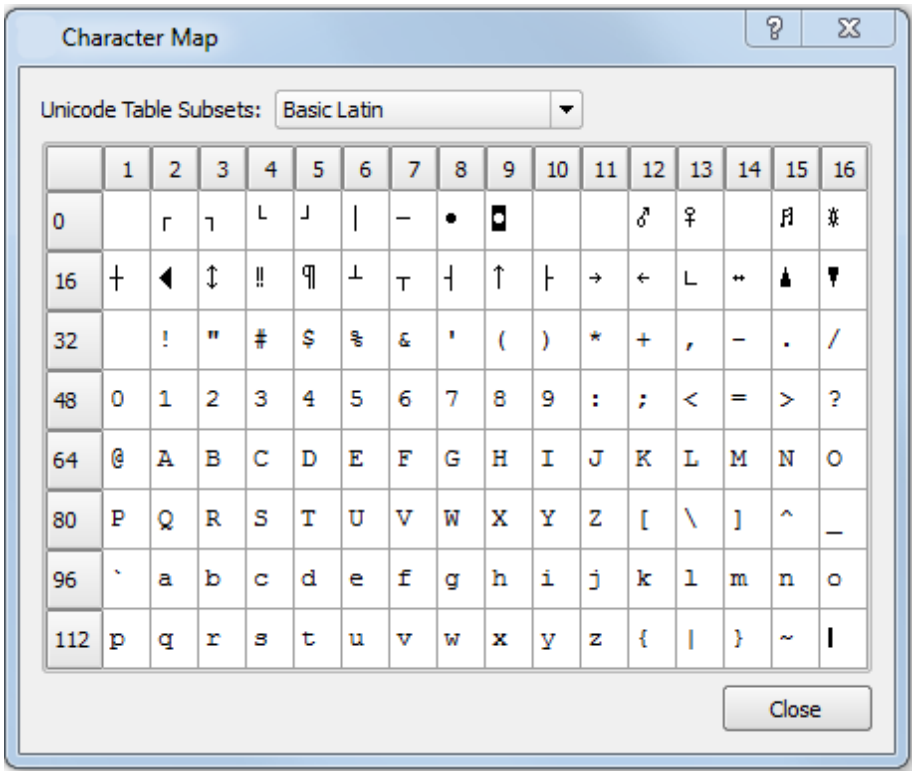


Figure 101: Dialog Box to Insert Unicode Characters into Code Tables

The drop-down list <**Unicode table subsets**> enables to choose alphabets such as

- Greek and Coptic
- Cyrillic
- Arabic
- Chinese Japanese Korean (CJK) Unified Ideographs

To insert Unicode characters to the code table, double click the Unicode character to insert.

To change the cell in which to insert the Unicode characters, click the mouse on the desired cell in the <**Code Table**> dialog box.

To stop inserting Unicode characters into the code table, click the button <**Close**>.

11.2. Complete Editing

To complete the editing of a code table, either press <**OK**> or <**Cancel**>. On pressing <**Cancel**> all modifications applied to the selected code table will be discarded, otherwise the modifications will be saved to the code table file, and the edited code table is ready for use.

11.3. Create and Delete Code Tables

11.3.1. Create Code Table

go2ANALYSE can be used to create new code tables. To create a code table, press the <**New**> button in the <**Extras**><**Configure Code Tables**> dialog box. The following dialog box is shown.

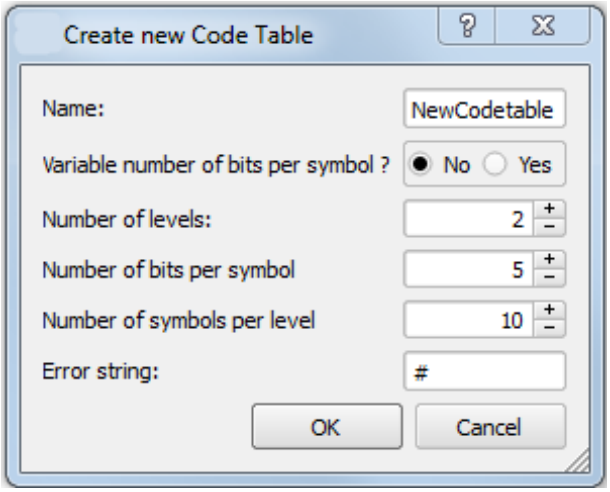


Figure 102: Dialog Box to Set the Parameters of Code Tables to be Created

Parameter	Defined Function
<**Name**>	Name of new code table
<**Variable number of bits per symbol?**>	Defines if the code table has a fixed number of bits per symbol or a variable number of bits per symbol. Example of a code table with a variable number of symbols is a Huffman code table.
<**Number of levels**>	Number of code table subsets (levels) to be created. Note: If the option <**Variable number of bits per symbol?**> is set, the code table is only allowed to have a single level.
<**Number of bits per symbol**>	Defines the number of bits for each code table. Example: To define a Baudot code table this parameter must be set to 5.
<**Number of symbols per level**>	Defines the number of symbols in each level of the code table. Example: To define a Baudot code table with a third shift this parameter must be set to 3.
<**Error string**>	Defines the error string for this code table. The error string is used if bits are mapped to text via this code table. If the symbol value of the bits does not match any symbol of the code table, the error string will be inserted into the text.

Table 63: Code Table Creation Parameters

On pressing <**OK**> the code table created is shown in accordance with the parameters set in the <**Create**

new Code Table> dialog box. The code table created can now be edited as described in chapter Edit Existing Code Table on page 102.

When terminating the editing of the newly created code table by pressing <**OK**>, the code table created is saved to the code table directory of go2ANALYSE and the name of the code table is added to the list of available code tables.

Example

Create Code Table with fixed number of Bits per Symbol

Press the <**New**> button.

Enter the name *CreatedCodetable*.

Enter the following parameters in the text and spin boxes of the <**Create new Code Table**> dialog box:

Parameter	Value
<**Name**>	CreatedCodetable
<**Variable number of bits per symbol?**>	No
<**Number of levels**>	2
<**Number of bits per symbol**>	5
<**Number of symbols per level**>	10

Table 64: Code Table Creation - Example

After clicking <**OK**> the created code table is displayed.

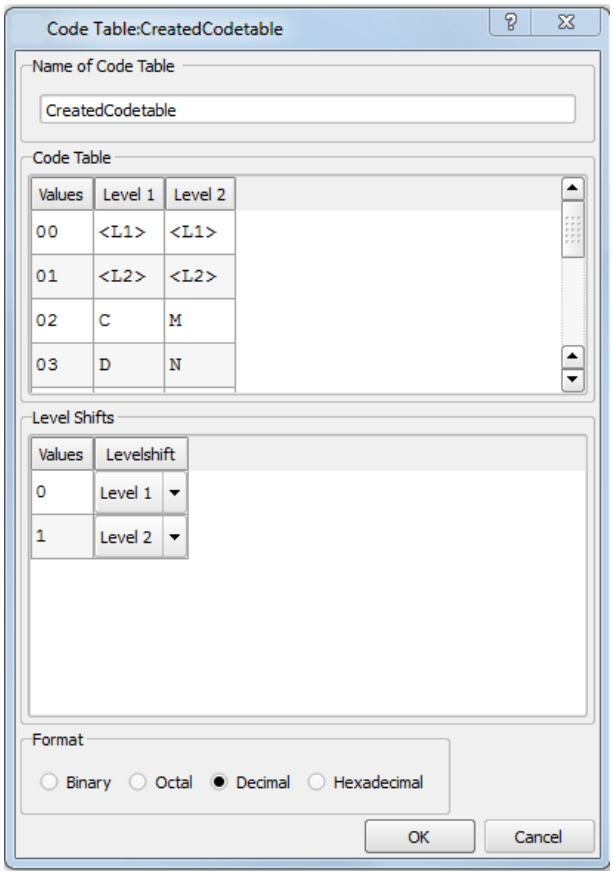


Figure 103: Example of Created Code Table

Example

Create Code Table with variable number of Bits per Symbol.

Assume a *Huffman* code table must be created. To do so, it must be possible to assign a varying number of bits to each symbol.

Enter the following parameters in the text and spin boxes of the <**Create New Code Table**> dialog box.

Parameter	Value
<**Name**>	CreatedVarCodetable
<**Variable number of bits per symbol?**>	Yes
<**Number of symbols per level**>	10

Table 65: Code Table Creation - Example with variable Number of Bits per Symbol

Press <**OK**>.

The <**Create new code table**> dialog is closed and the created code table is displayed:

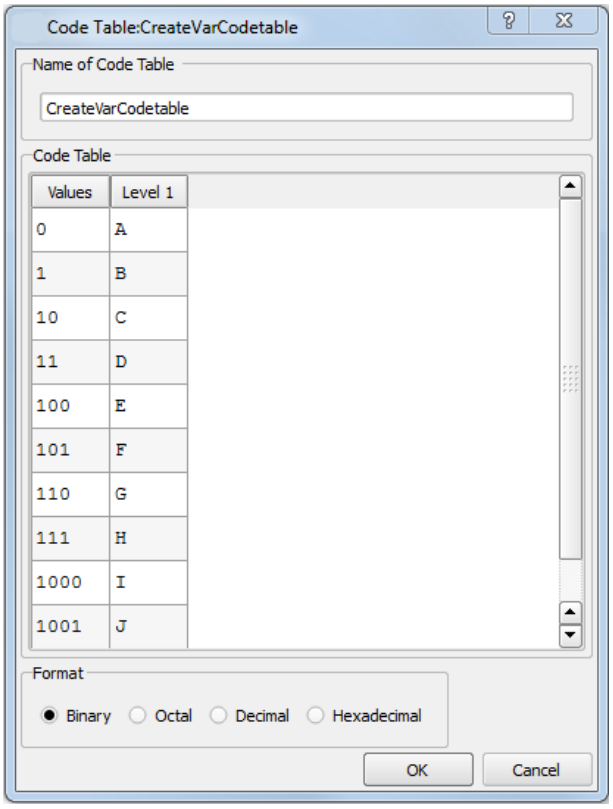


Figure 104: Example of Code Table Created with a variable Number of Bits per Symbol

Change the option in the group box *Format* to <**Binary**>. As you will see, the symbols have a variable length of symbols.

11.3.2. Delete Code Table

To delete a code table, select the code table to delete and press the <**Delete**> button on the <**Configure Code Tables**> dialog box.

Note: go2ANALYSE will delete the code table file from the directory where go2ANALYSE is installed.

11.4. Import and Export Existing Code Tables

11.4.1. Export go2ANALYSE Code Tables

go2ANALYSE can backup all available code tables by means of the <**Export**> button on the <**Configure Code Tables**> dialog box.

To do so, press the button <**Export to...**> on the *Configure Code Tables* dialog box. Then another dialog box is shown from go2ANALYSE. This dialog box is used to select any directory in the file system, e.g. *C:\CodeTableBackup*.

On clicking the <**OK**> button all files listed in the <**Configure Code Tables**> dialog box will be saved to the selected directory.

11.4.2. Import Code Tables from other directories

go2ANALYSE can import code tables as files ("*.ctb" files, see specification of Code Table File Format for details) from other directories into go2ANALYSE. To do so, press the <**Import from...**> button on the *Configure Code Tables* dialog box.

An import dialog box is displayed in which "*.ctb" files can be selected in all directories of your file system.

Having selected one or several "*.ctb" files to import, click the <**OK**> button. The selected files are copied to the current directory.

Each file is loaded by go2ANALYSE and the list of code tables is updated.

Note: go2ANALYSE will display an alert message when attempting to import a code table whose name is identical to the name of a code table already loaded by go2ANALYSE.

When the alert message is acknowledged by <**OK**> go2ANALYSE will show the directory with all code tables available and provide the option to enter a name for the code table to be imported. Please enter a unique name.

12. Linking External Applications

Any displayed bit stream can be passed to external applications in a quick way. As go2ANALYSE is optimized for pure bit level processing, useful add-ons may be text editors, hex-editors, disassemblers, decompression tools, etc. External applications can be configured by <**Extras**><**Configure External Tools...**>.

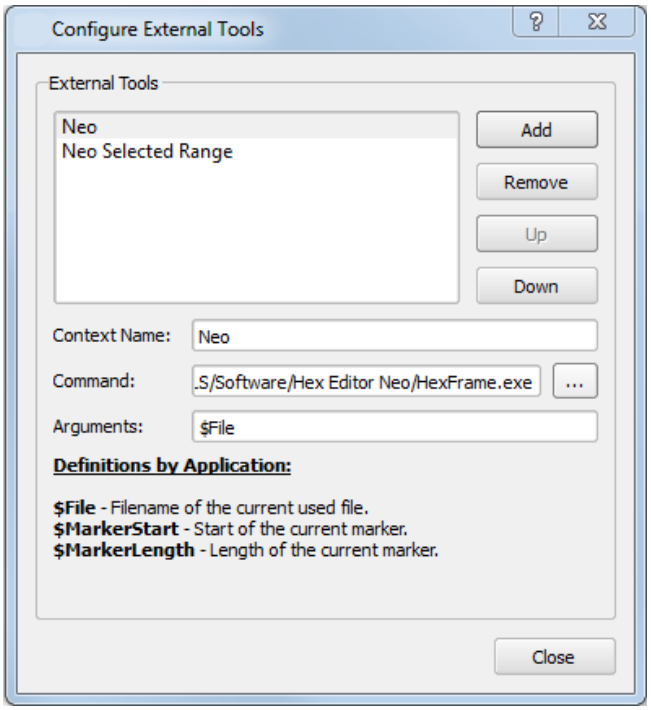


Figure 105: Configuring External Applications

The displayed input fields are used as follows:

Function	Description
<**Context Name**>	The application name, which shall be displayed as an additional menu item
<**Command**>	Put in the command as you would start the application via command line
<**Arguments**>	Additional command line arguments needed for application start. The following replacement expressions can be used to pass go2ANALYSE internal parameters: $File — This will include the name of an internal temporary file generated to transfer the contents of <**Bit Display**> $MarkerStart — The bit displays marker start position (byte number) will be added as an argument $MarkerLength — The bit displays marker length position (number of bytes) will be added as an argument

Table 66: Configuring External Applications Parameters

Figure 106 shows a setting for the Hex-Editor "Neo", to pass the bit displays contents as well as the marker selection.

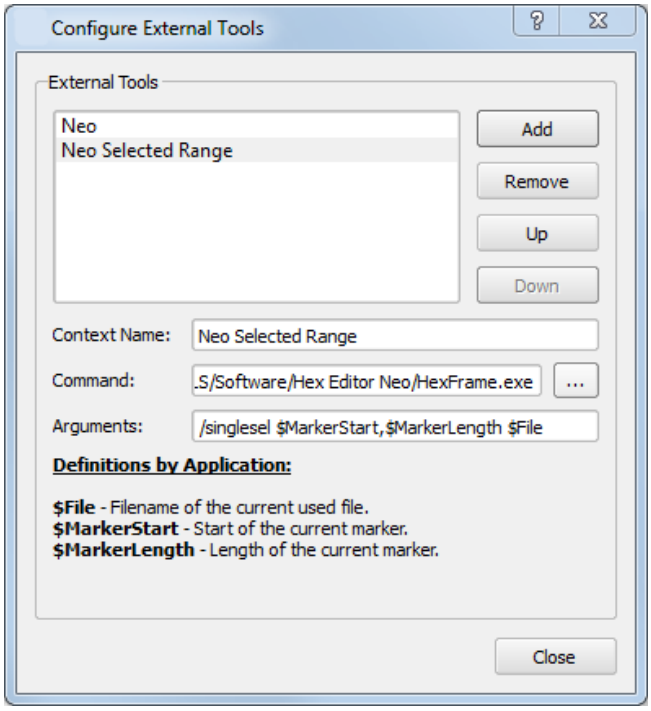


Figure 106: Configuring External Applications with Marker Values

The application can be called from <**Extras**>.

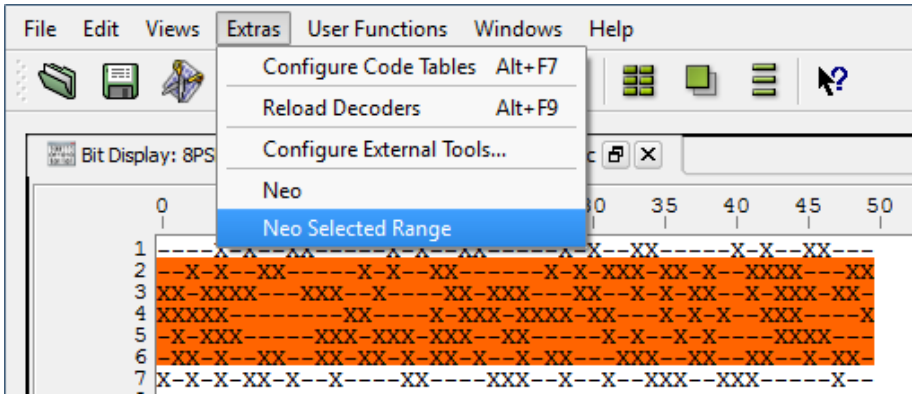


Figure 107: Calling External Functions

13. Technical Reference

13.1. Keyboard Shortcuts

Function	Shortcut
<Open...>	<Ctrl>+<O>
<Save...> / <Save selection>	<Ctrl>+<S>
<Copy>	<Ctrl>+<C >
<Replace>	<Ctrl>+<V >
<Insert>	<Ctrl>+<I>
<Select all>	<Ctrl>+<A>
<Undo>	<Ctrl>+<Z>
<Redo>	<Ctrl>+<Y>
<Decoder Editor>	<Ctrl>+<E>
<Attributes & Statistics>	<Alt>+<T>
<Close File>	<Alt>+<C>
<Exit>	<Alt>+<F4>
<Configure Code Tables>	<Alt>+<F7>
<What's This?>	<Shift>+<F1>

Table 67: Keyboard Shortcuts

13.2. Built-In Code Tables

13.2.1. Baudot Code Table

Bit-Code	Decimal	Hex	Letter	Figure
00000	0	00	Ignore (@)	
00001	1	01	T	5
00010	2	02	Carriage Return (CR)	Carriage Return (CR)
00011	3	03	O	9
00100	4	04	Space	Space
00101	5	05	H	Space
00110	6	06	N	,
00111	7	07	M	.
01000	8	08	Line feed (LF)	Line feed (LF)
01001	9	09	L	)
01010	10	0a	R	4
01011	11	0b	G	&
01100	12	0c	I	8
01101	13	0d	P	0
01110	14	0e	C	:
01111	15	0f	V	=
10000	16	10	E	3
10001	17	11	Z	+
10010	18	12	D	$
10011	19	13	B	?
10100	20	14	S	\
10101	21	15	Y	6
10110	22	16	F	!
10111	23	17	X	/
11000	24	18	A	-
11001	25	19	W	2
11010	26	1a	J	~
11011	27	1b	Figure Shift >	Figure Shift >
11100	28	1c	U	7
11101	29	1d	Q	1
11110	30	1e	K	(

Bit-Code	Decimal	Hex	Letter	Figure
11111	31	1f	Letter Shift <	Letter Shift <

Table 68: Baudot Code

13.2.2. ITA2P, ITA3, CCIR476 Code Tables

No.	Letter	Figure	ITA2P ARQ1A	ITA3	CCIR476 SITOR
1	A	-	0110001	0011010	0001110
2	B	?	0100110	0011001	1011000
3	C	:	0011100	1001100	0100011
4	D	$	0100101	0011100	0011010
5	E	3	0100000	0111000	1001010
6	F	%	0101100	0010011	0010011
7	G	&	0010110	1100001	0101001
8	H	#	0001011	1010010	0110100
9	I	8	0011001	1110000	0100110
10	J	@	0110100	0100011	0001011
11	K	(	0111101	0001011	1000011
12	L	)	0010011	1100010	0101100
13	M	.	0001110	1010001	0110001
14	N	-	0001101	1010100	0110010
15	O	9	0000111	1000110	0111000
16	P	0	0011010	1001010	0100101
17	Q	1	0111011	0001101	1000101
18	R	4	0010101	1100100	0101010
19	S	~	0101001	0101010	0010110
20	T	5	0000010	1000101	1101000
21	U	7	0111000	0110010	1000110
22	V	=	0011111	1001001	1100001
23	W	2	0110010	0100101	0001101
24	X	/	0101111	0010110	1010001
25	Y	6	0101010	0010101	0010101
26	Z	+	0100011	0110001	0011100
27	Carriage return (cr)	cr	0000100	1000011	1110000
28	Line feed (lf)	lf	0010000	1011000	1100100
29	Letter shift >	>	0111110	0001110	1010010
30	Figure shift <	<	0110111	0100110	1001001
31	Space (sp)	sp	0001000	1101000	1100010
32	Idle α	α	0000001	0000111	1010100
	RQ, Ph 2 β	β	1110000	0110100	1001100

No.	Letter	Figure	ITA2P ARQ1A	ITA3	CCIR476 SITOR
	Idle, beta θ	θ	1001001	0101001	0011001
	Idle, alpha, Ph 1 π	π	1000110	0101100	0000111
	CS1 γ (gamma)	γ			0101100
	CS2 δ	δ			1010100
	CS3 ϵ	ε			0110010
	CS4 ζ	ζ			0101001
	CS5 η	η			0110100
	Not defined (§)	§	All others	All others	All others

Table 69: ITA2P, ITA3 and CCIR476 Codes

13.3. Code Table File Format

The code tables used by the Text Display are stored in files with an extension "*.ctb" using the following conventions:

- Each code table starts with this qualifier: ALPHA_DEF(N)
- N represents the table identification number, an arbitrary decimal number

Note: Only the first code table of each file will be read by go2ANALYSE. It ends with this qualifier: END_ALPHA_DEF

13.3.1. Table Attributes and Table Switches

The following notations are used to store the parameters of the code tables to a file (N always representing a decimal value in full).

BitNo = N length of input code in bits (<= 32)
N = 0 indicates that the length is variable (e.g. Huffman code)
NoLevels = N number of symbol levels (e.g. letters/figures)
ErrSymb = # defines the output-symbol in case of a non-defined input code

13.3.2. List of Table Values

Table values require the following structure of annotation:

Example

User defined code table

```
TABLE
; Input Letter Figure
0 , <> ,<> ; here, comments are allowed
1 , T , 5
2 ,<CR> ,<CR>
3 , O , 9
4 ,<_> ,<_>
5 , H ,<>
6 , <IDLE> ,<44>
7 , M , .
8 ,<LF> ,<LF>
9 , L , )
10 , R , 4
11 , G , &
12 , I , 8
13 , P , 0
14 , C , :
15 , V , =
16 , Number16 , 3
17 , Z , +
18 , D ,<WRU>
19 , B , ?
20 , S , ′
21 , Y , 6
22 , F ,<>
23 , X , /
24 , A , -
25 , W , 2
26 , J ,<BEL>
27 ,<L2> ,<L2> ; figure shift
28 , U , 7
29 , Q , 1
30 , K , (
31 ,<L1> ,<L1> ; letter shift
ENDTABLE
```

The table input values are placed in the first column either in decimal, hexadecimal, octal or binary annotation. The number of additional output columns must be first predefined by means of *NoLevels* in the table attribute section. Output values may consist of up to 50 characters. Any comments in a "*.ctb" file will be ignored by go2ANALYSE.

The following reserved character sequences have predefined functions:

Reserved Character	Defined Function
<>	No output
<_>	Space character
<LF>	Line feed
<CR>	Carriage return
<L1>	Switch to output level 1
<L2>	Switch to output level 2
<LN>	Switch to output level 3
{value}	ASCII value of symbol (0-255)
{44}	comma
{59}	semicolon

Table 70: Code Table Reserved Characters

For reasons of general syntax, the characters “,” (comma) and “;” (semicolon) must be edited with their ASCII annotation {44} and {59} exclusively.

13.4. Interfaces

13.4.1. Format for Input/Output Bitstreams

The format for input/output bitstreams is the “*.txt” format with characters according to the following conversion table:

Character	Converted Bit
0	0
- (Dash)	0
_ (Underline)	0
L	0
. (Dot)	0
1	1
X	1
x	1
H	1

Table 71: Bitstream Conversion Table

Note: Any other characters in the text file will be ignored!

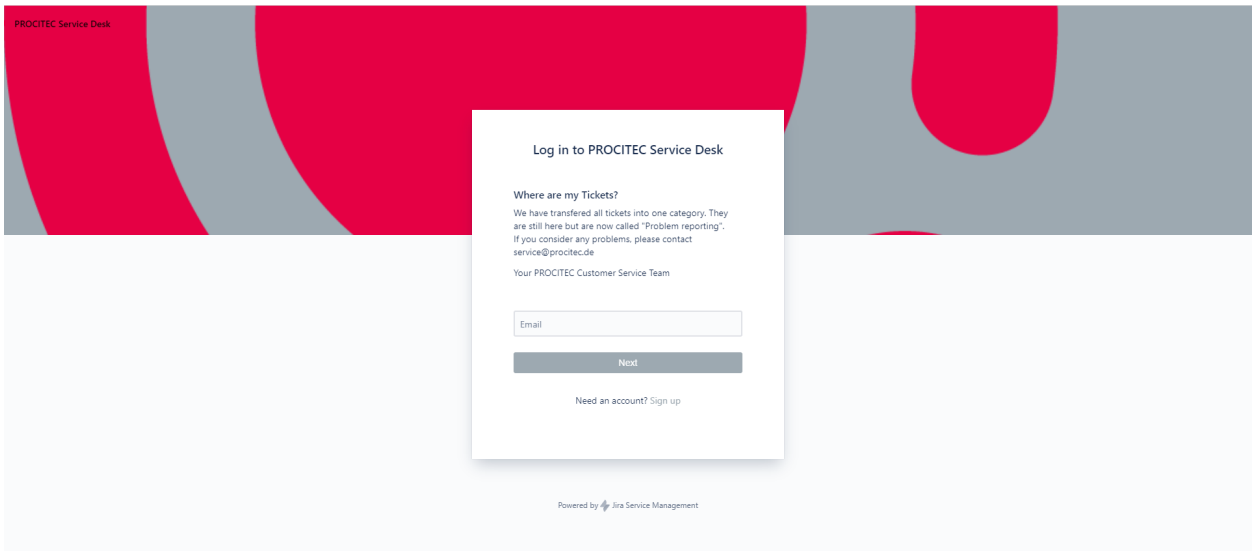
A. Support

Requests and suggestions?

All requests or suggestions regarding our go2signals product-range are very much appreciated; we would be delighted to hear from you.

Any questions? We are happy to assist you!

If you have any further questions, please do not hesitate to contact our Support Team for rapid assistance – just raise a service request at: **http://servicedesk.procitec.com**.



PROCITEC GmbH
Rastatter Straße 41
D-75179 Pforzheim
Phone: +49 7231 15561 0
Web: www.procitec.com
Email: service@procitec.com

List of Figures

List of Tables