BitView Manual V2.3.00

WAVECOM W-BV

by WAVECOM ELEKTRONIK AG

WAVECOM®

NACHRICHTENTECHNIK

PUBLISHED BY
WAVECOM ELEKTRONIK AG
Hammerstrasse 8
CH-8180 Buelach
Switzerland

Phone	+41-44-872 70 60
Fax	+41-44-872 70 66
Email:	info@wavecom.ch
Internet:	http://www.wavecom.ch

Printed: Wednesday, February 11, 2009, 12:49:05

Contents

Introduction

Introduction

WAVECOM-BitView (W-BV) enables the user to analyze any bit stream. The range of functions extends from the display of a bit stream in various formats, simple bit stream manipulations, over statistical functions to complex mathematical functions and functions based on coding theory. The tools are targeted at users with experience in bit stream analysis. To understand some of the functions a comprehensive mathematical knowledge is a prerequisite.

Direct bit stream input from W51PC, W61PC and W-CODE is supported.

BitViewTool supports the Windows 2000, XP, 2003 Server and Vista operating systems.

The product that you bought incorporates the latest technology in data decoding together with the latest software release available at the time of shipment.

Please, check our website http://www.wavecom.ch for software updates.

Always check the latest documentation on the installation CD or on our website.

We thank you for choosing a WAVECOM decoder and look forward working with you in the future.

This chapter introduces WAVECOM, the field of activity of the company, and how you may benefit from the expertise of WAVECOM.

Throughout this document the terms "BitView", "W-BV" and "BitViewTool" all designate "WAVECOM-BitView".

Training

WAVECOM offers all our customers a complete, professional training program covering all the key features of our products.

Depending on your skills (if you are an expert or a beginner), together we will work out a special training program for you.

Training is available on your location or in Switzerland.

Source Code

Source code is available for government bodies. Please, inquire an offer from WAVECOM, if you plan to add your own modes.

Company Profile

WAVECOM ELEKTRONIK GmbH was founded in 1985 in Hohentengen, Germany, close to the Swiss border. In 1991 the company moved to Switzerland and established itself as WAVECOM ELEKTRONIK AG. Now located in Buelach it is within close vicinity of Zurich airport.

The company has focused on decoding and analysis systems for wireless data transmissions. The wide product range spans from professional, high performance systems to devices for private and amateur radio use.

The very high quality standards combined with high system performance are appreciated by all customers worldwide. A global network of authorized sales partners ensures that local assistance and basic level support can be provided in most places. More than 95% of all units sold are exported. The majority of the customers are government agencies, defense organizations and the telecommunication industry.

About 40% of the turnover is invested in research and development. The employees at WAVECOM ELEKTRONIK AG are mainly engineers with experience in DSP technology, computer and RF hardware development, and software engineering and radio data transmission. Access to external know-how and human resources enlarges the capabilities for realizing projects. Manufacturing is outsourced to specialized companies within Switzerland which can handle today's needs for processing surface mount components and fine-pitch structures.

WAVECOM ELEKTRONIK AG does not have any juridical or financial links or connections to other companies or official bodies and is completely owned by Mr. Christian Kesselring.

Revisions

Version	Date	Changes
1.1	10.May.2007	
2.0	05.Nov.2007	Installation folders changed 'Hide on close' preference added Layout settings removed from context menu Layout settings now in the property grid Graphic layout added 'Inversion' function name inversion changed to 'polarity' Bit Sync Analysis added Custom library updated
2.1	14.Mar.2008	Hexadecimal view added Enhanced printer dialog MatLab custom libraries
2.1.01	15.Apr.2008	General overwork : ➢ Improved readability ➢ Extended explanations 'Parameters' window changed to 'Properties' window
2.2.00	22.Jul.2008	Works with W51PC, W61PC and W-CODE New functions: ➢ Unzip ➢ Autocalculation ➢ Enable function
2.3.00	7.Feb.2009	New functions: ➢ AND/OR/XOR/NOT Range ➢ Extraction (Range) ➢ General Reed-Solomon Decoding ➢ Parity (Even/Odd/Mark/Space) ➢ Parity from H-Matrix ➢ Parity form polynomial ➢ Descrambler (Pseudo Noise) ➢ Pager Numeric Custom libraries with source code Additional information in the status line of the main window Enhanced performance of the Auto-Calculation function Functions may be enabled or disabled in History Explorer

Requirements

.NET Framework version 2.0 must be installed. The framework is included in the setup and is installed if missing on the system.

Limitations

In this version of BitView, the maximum number of bits that can be imported is limited to 500,000.

Bear in mind that some formatting functions such as bit highlighting consume a lot of CPU power and may require considerable time to complete, especially on less powerful systems. Reducing the number of imported bits will speed up the application.

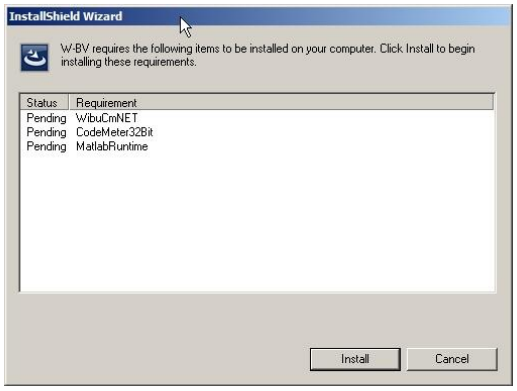
Bit streams are imported completely raw and unsynchronized, i.e., BitView will not recognize additional information like confidence levels from soft-decoders or symbol boundaries for m-ary modulation types. Any such information must be removed using the BitView toolbox functions.

Installation

W-BV Software Installation

To install the application, click **SetupBitViewTool.exe.** Files are then unpacked and copied to the installation folder. Ini files are not generated.

By default BitViewTool will be installed in the WAVECOM folder, where other WAVECOM products may be installed.



W-BV requires the following items to be installed on your computer. Click Install to begin installing these requirements.

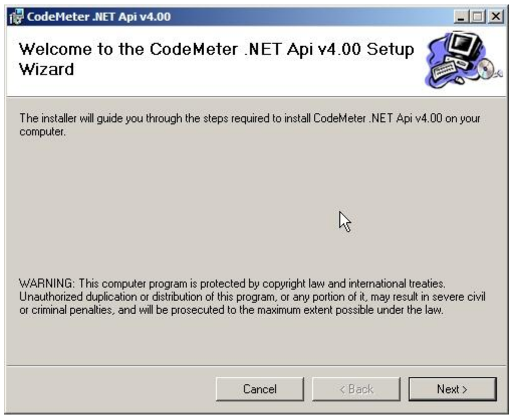
Status	Requirement
Pending	WibuCmNET
Pending	CodeMeter32Bit
Pending	MatlabRuntime

Press **Install** to continue the installation of the four components which constitute BitView,

- CodeMeter .NET API
- CodeMeter Runtime
- MatLab Runtime
- BitView application

The installation is self-explanatory and to complete it follow the instruction on the screen.

CodeMeter .NET API installation

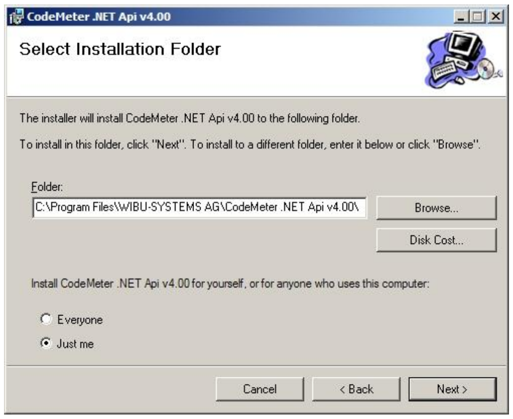


Welcome to the CodeMeter .NET Api v4.00 Setup Wizard

The installer will guide you through the steps required to install CodeMeter .NET Api v4.00 on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Pressing **Next** will start the installation of this component.


CodeMeter .NET Api v4.00

Select Installation Folder

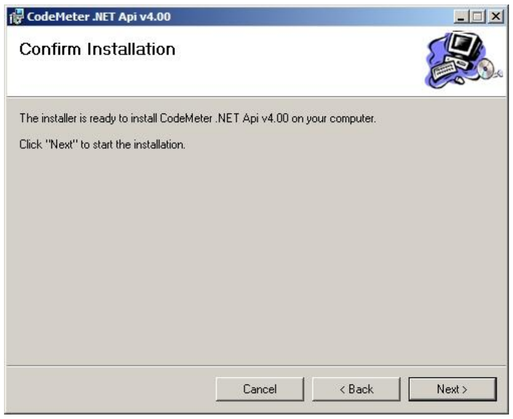
The installer will install CodeMeter .NET Api v4.00 to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:\Program Files\WIBU-SYSTEMS AG\CodeMeter .NET Api v4.00\

Install CodeMeter .NET Api v4.00 for yourself, or for anyone who uses this computer:

Everyone [✓ Just me]

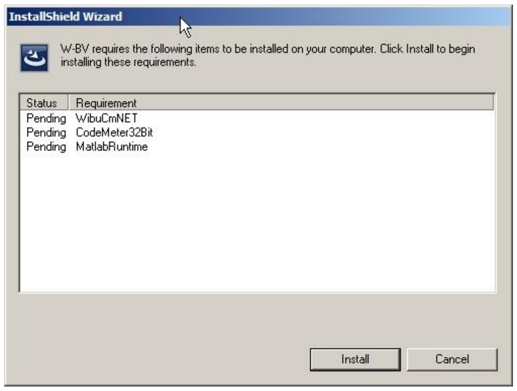

CodeMeter .NET Api v4.00

Confirm Installation

The installer is ready to install CodeMeter .NET Api v4.00 on your computer.

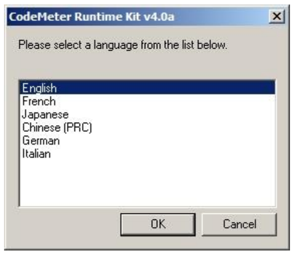
Click "Next" to start the installation.

CodeMeter Runtime installation


InstallShield Wizard

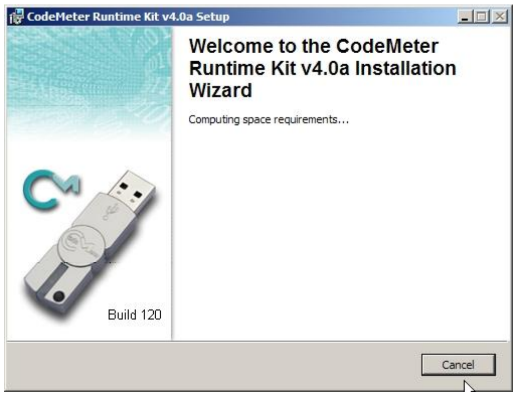
W-BV requires the following items to be installed on your computer. Click Install to begin installing these requirements.

Status	Requirement
Pending	WibuCmNET
Pending	CodeMeter32Bit
Pending	MatlabRuntime


CodeMeter Runtime Kit v4.0a

Please select a language from the list below.

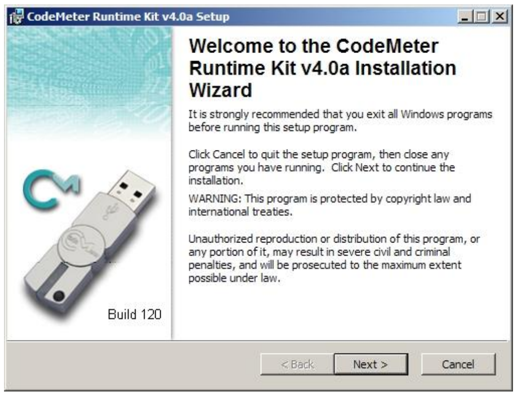
[✓ English]
French
Japanese
Chinese (PRC)
German
Italian


CodeMeter Runtime Kit v4.0a Setup

Welcome to the CodeMeter Runtime Kit v4.0a Installation Wizard

Computing space requirements...

Build 120


CodeMeter Runtime Kit v4.0a Setup

Welcome to the CodeMeter Runtime Kit v4.0a Installation Wizard

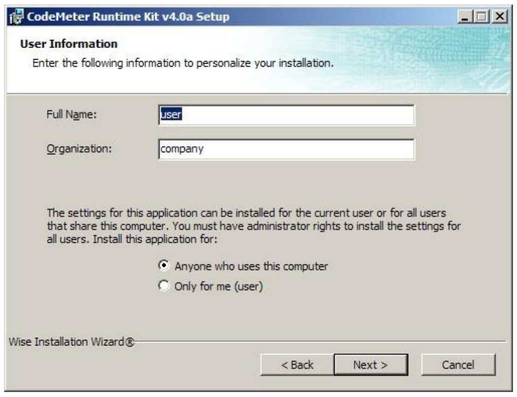
It is strongly recommended that you exit all Windows programs before running this setup program.

Click Cancel to quit the setup program, then close any programs you have running. Click Next to continue the installation.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Build 120


CodeMeter Runtime Kit v4.0a Setup

User Information

Enter the following information to personalize your installation.

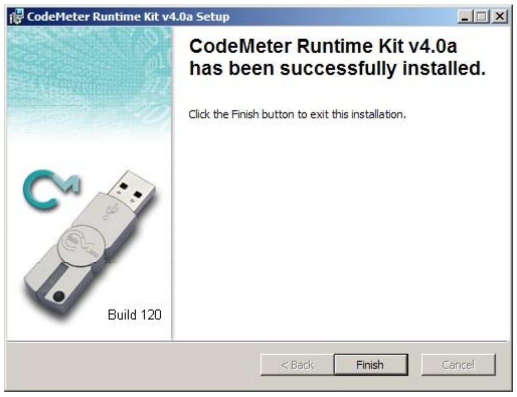
Full Name: user

Organization: company

The settings for this application can be installed for the current user or for all users that share this computer. You must have administrator rights to install the settings for all users. Install this application for:

● Anyone who uses this computer
○ Only for me (user)

Wise Installation Wizard®

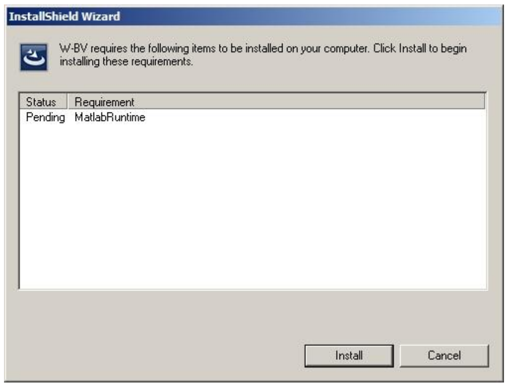

CodeMeter Runtime Kit v4.0a Setup

CodeMeter Runtime Kit v4.0a has been successfully installed.

Click the Finish button to exit this installation.

Build 120

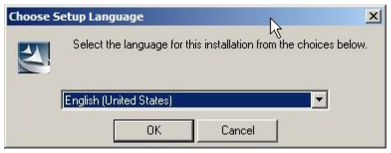
MatLab Runtime installation



InstallShield Wizard

W-BV requires the following items to be installed on your computer. Click Install to begin installing these requirements.

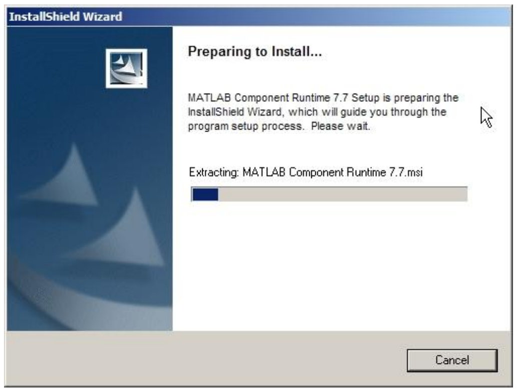
Status	Requirement
Pending	MatlabRuntime



Choose Setup Language

Select the language for this installation from the choices below.

English (United States)

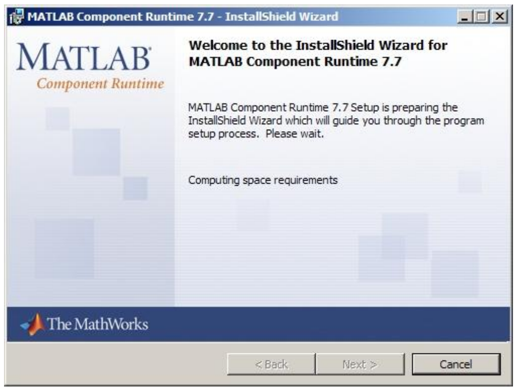


InstallShield Wizard

Preparing to Install...

MATLAB Component Runtime 7.7 Setup is preparing the InstallShield Wizard, which will guide you through the program setup process. Please wait.

Extracting: MATLAB Component Runtime 7.7.msi

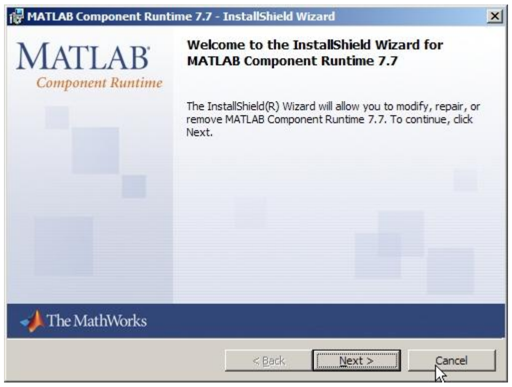


MATLAB Component Runtime 7.7 - InstallShield Wizard

Welcome to the InstallShield Wizard for MATLAB Component Runtime 7.7

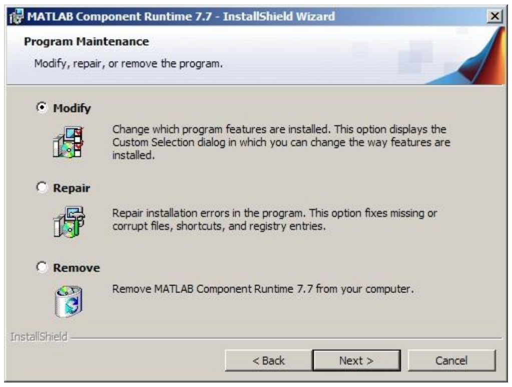
MATLAB Component Runtime 7.7 Setup is preparing the InstallShield Wizard which will guide you through the program setup process. Please wait.

Computing space requirements


MATLAB Component Runtime 7.7 - InstallShield Wizard

Welcome to the InstallShield Wizard for MATLAB Component Runtime 7.7

The InstallShield(R) Wizard will allow you to modify, repair, or remove MATLAB Component Runtime 7.7. To continue, click Next.


MATLAB Component Runtime 7.7 - InstallShield Wizard

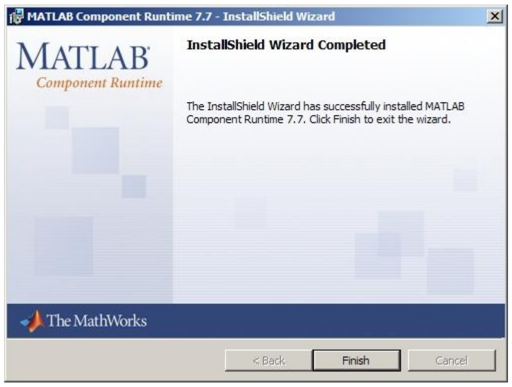
Program Maintenance

Modify, repair, or remove the program.

[✓ Modify] Change which program features are installed. This option displays the Custom Selection dialog in which you can change the way features are installed.

Repair: Repair installation errors in the program. This option fixes missing or corrupt files, shortcuts, and registry entries.

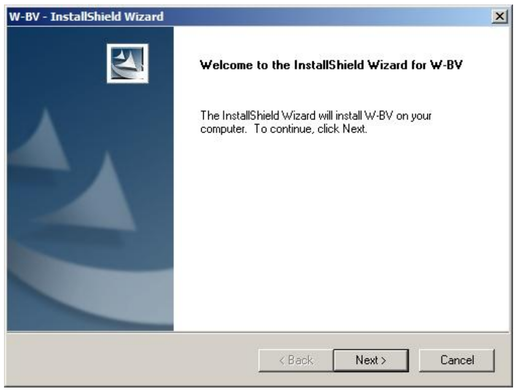
Remove: Remove MATLAB Component Runtime 7.7 from your computer.


MATLAB Component Runtime 7.7 - InstallShield Wizard

InstallShield Wizard Completed

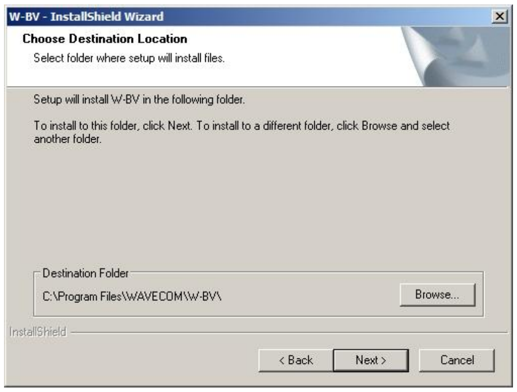
The InstallShield Wizard has successfully installed MATLAB Component Runtime 7.7. Click Finish to exit the wizard.

BitView installation


W-BV - InstallShield Wizard

Welcome to the InstallShield Wizard for W-BV

The InstallShield Wizard will install W-BV on your computer. To continue, click Next.


W-BV - InstallShield Wizard

Choose Destination Location

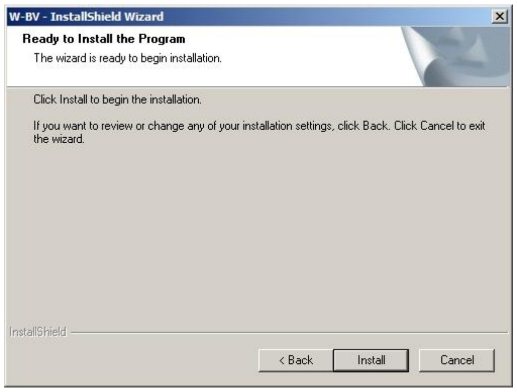
Select folder where setup will install files.

Setup will install W-BV in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\Program Files\WAVECOM\W-BV\

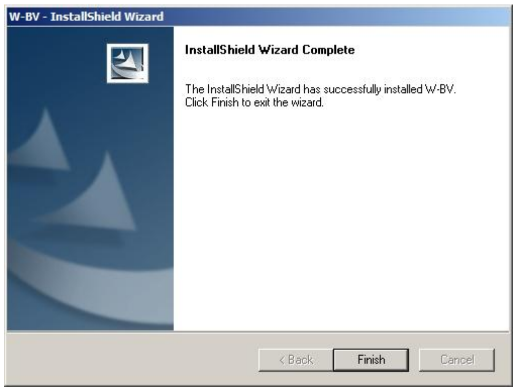

W-BV - InstallShield Wizard

Ready to Install the Program

The wizard is ready to begin installation.

Click Install to begin the installation.

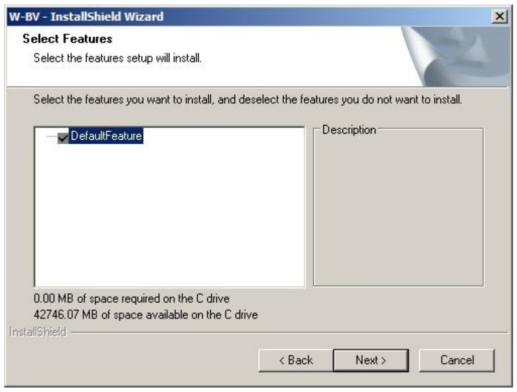
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.


W-BV - InstallShield Wizard

InstallShield Wizard Complete

The InstallShield Wizard has successfully installed W-BV. Click Finish to exit the wizard.

BitView defaults installation


W-BV - InstallShield Wizard

Select Features

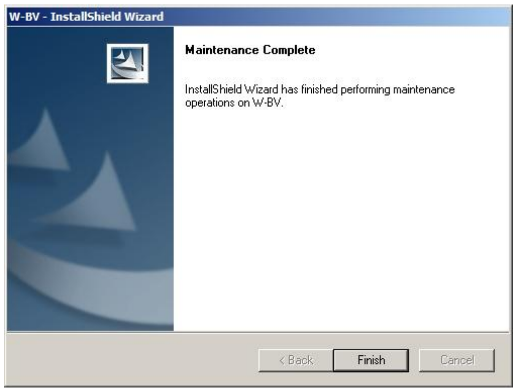
Select the features setup will install.

Select the features you want to install, and deselect the features you do not want to install.

DefaultFeature

Description

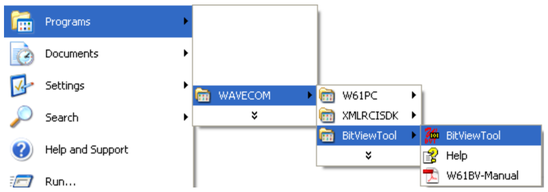
0.00 MB of space required on the C drive
42746.07 MB of space available on the C drive


W-BV - InstallShield Wizard

Maintenance Complete

InstallShield Wizard has finished performing maintenance operations on W-BV.

Press **Finish** to complete the installation of all BitView components.



BitViewTool may be uninstalled by using the **Add/Remove Programs** item found in the **Control Panel** menu.

Paths

Examples and CustomLib files are copied to the following folder:

Windows XP and earlier:

- Documents and Settings\All Users\Documents \WAVECOM\BitViewTool\
- Documents and Settings\All Users\Shared Documents \WAVECOM\BitViewTool\

Windows Vista:

- Users\Public\Public Documents\WAVECOM\BitViewTool\

W-BV Hardware Installation

Apart from a Windows PC and software protection hardware, no additional hardware is required to operate the software.

CmStick

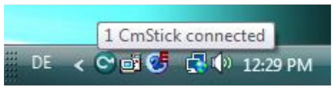
Software protection hardware has to be connected to the computer. The hardware device used for the software protection is called the CmStick and is available as:

- A small USB device
- A PC Card (CmCard/M, Cardbus, 32 Bit)
- An Express Card|34 (CmCard/E)

The W-BV application will not start if the appropriate and valid licenses are not found on a CmStick connected to the system.

After the installation of the software on the computer, the CmStick icon will be displayed in the tray icon area.

Note: when a CmStick is plugged into an USB socket of a LCD monitor, the CmStick will no longer be detected by the software protection server if the monitor is switched off. Hence a running application relying on the licenses stored on such a CmStick will stop working or disable its features.



W-BV Licensing

By default, two license models are available

- W-BV with an update period of 12 months
- W-BV with an update period of 36 months

After the initial update period has elapsed, it is possible to order an extension for the update period by another 36 months.

On request, we can generate different license models, even complex ones

- Single user license
- Evaluation licenses (number of starts, limited usage period, activation and/or expiration time)
- Modular licenses (activation of additional functions)
- Also licenses from other software manufacturers can be stored on a CmStick containing WAVECOM licenses

W-BV Software Options & Updates

Additional functions or services may be licensed to work with your tool, e.g.:

- W-CODE decoder
- W61PC decoder
- Classifier
- an extension by another 36 months of the software update period

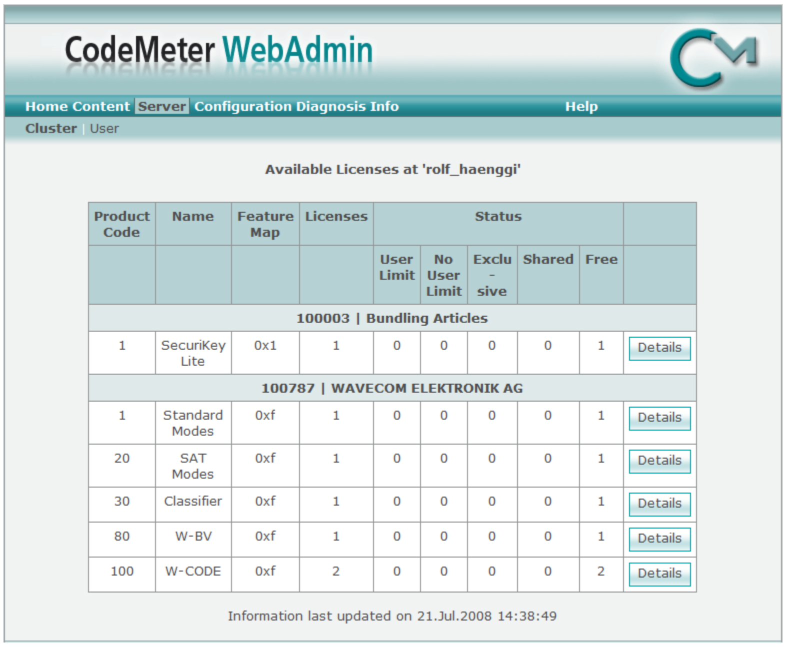
To process an order, the following information is required:

- full address
- ordered items
- email or post delivery
- serial number of CmStick
- remote context file of CmStick to be updated, if applicable

W-BV license checking

To check the license(s) on the CmStick follow these steps:

Click the icon in the tray icon area.



Available Licenses at 'rolf_haenggi'

Product Code	Name	Feature Map	Licenses	Status					
				User Limit	No User Limit	Exclu-sive	Shared	Free	
100003 | Bundling Articles									
1	SecuriKey Lite	0x1	1	0	0	0	0	1	Details
100787 | WAVECOM ELEKTRONIK AG									
1	Standard Modes	0xf	1	0	0	0	0	1	Details
20	SAT Modes	0xf	1	0	0	0	0	1	Details
30	Classifier	0xf	1	0	0	0	0	1	Details
80	W-BV	0xf	1	0	0	0	0	1	Details
100	W-CODE	0xf	2	0	0	0	0	2	Details

Information last updated on 21.Jul.2008 14:38:49

Item	Remarks
Product Code	Displays the Product Code
Name	Displays the name of the Product Item, normally the name of the product
Feature Map	Displays the Feature Map. WAVECOM uses the Feature Map to control the software upgrade period
Licenses	Displays the total number of network licenses
User Limit	Displays the number of licenses, which are currently used in the User Limit mode
No User Limit	Displays the number of licenses that are currently used in the No User Limit mode
Exclusive	Displays the number of licenses that are currently used in the Exclusive mode
Shared	Displays the number of licenses that are currently used in the Shared mode
Free	Displays the number of licenses that are currently free
Details	Displays detailed information about the respective network licenses in use

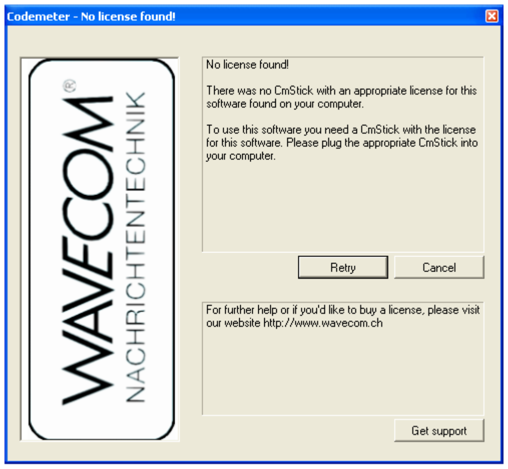
Important: If you have multiple CmSticks plugged into computers in a local network, then read the **CodeMeter and CmStick** chapter included in the "Appendix" on page 70.

Getting Started

Program Start

Starting the program will introduce a license check procedure.

If a valid license key was not found, the following message appears on the screen.


Codemeter - No license found!

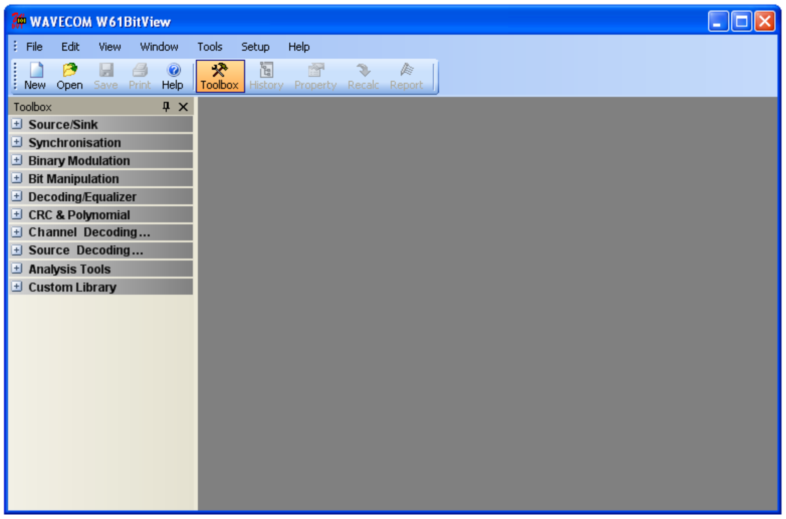
No license found!

There was no CmStick with an appropriate license for this software found on your computer.

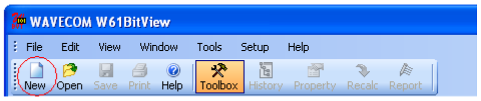
To use this software you need a CmStick with the license for this software. Please plug the appropriate CmStick into your computer.

For further help or if you'd like to buy a license, please visit our website http://www.wavecom.ch

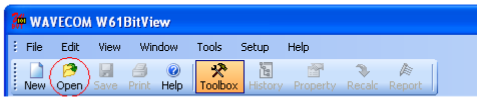
If a valid license key was found the application is started. The **Toolbox**, which contains the function library, is displayed and enables the user to import a bit stream from a selection of different sources.



Alternatively, using the **New** button from the **Toolbar**, an empty document window is opened which allows the user to manually create a bit stream, or copy and paste a bit stream from another source.

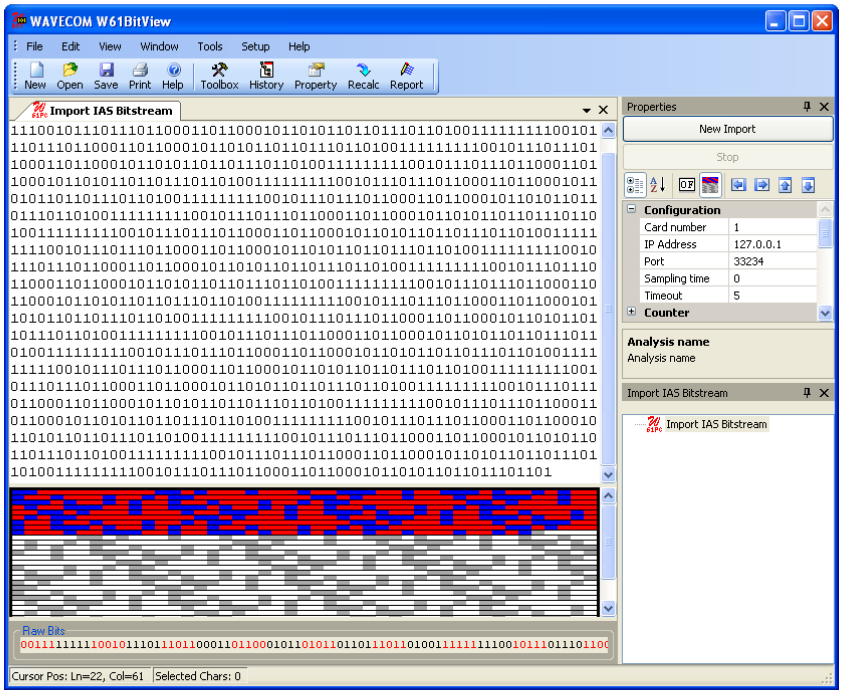


Another option, using the **Open** button from the **Toolbar,** allows a previously saved **Analysis Set** (stored in an .XML file) to be opened (see later in this manual for details).



Bit Stream Import

An imported bit stream is shown in a document window, and the **Properties** and **History Explorer** windows are opened. In general, the **Properties** window displays all the properties of a selected function, and the **History Explorer** window shows the dependencies of all functions in a tree view.


WAVECOM W61BitView
File Edit View Window Tools Setup Help
New Open Save Print Help Toolbox History Property Recalc Report
Import IAS Bitstream
Properties
New Import
Stop
Import IAS Bitstream
Raw Bits

```
1110010111011101100011011000101101011011011101101001111111100101
1101110110001101100010110101101101110110100111111111001011101101
1000110110001011010110110111011010011111111100101110111011000110 1
1000101101011011011101101001111111110010111011101100011011000101 1
0101101101110110100111111111001011101110110001101100010110101101 1
0111011010011111111100101110111011000110110001011010110111011 0110
1001111111110010111011101100011011000101101011011011101101001111 1
1111001011101110110001101100010110101101101110110100111111110010
1110111011000110110001011010110110111011010011111111100101110111 0
1100011011000101101011011011101101001111111110010111011101100011 0
1100010110101101101110110100111111111001011101110110001101100010 1
1010110110111011010011111111100101110111011000110110001011010110 1
1011101101001111111110010111011101100011011000101101011011011101 1
0100111111111001011101110110001101100010110101101101110110100111 1
1111100101110111011000110110001011010110110111011010011111111100 1
0111011101100011011000101101011011011101101001111111110010111011 1
0110001101100010110101101101110110100111111111001011101110110001 1
0110001011010110110111011010011111111100101110111011000110110001 0
1101011011011101101001111111110010111011101100011011000101101011 0
1101110110100111111111001011101110110001101100010110101101101110 1
1010011111111100101110111011000110110001011010110110111011101101
```

Raw Bits: 0011111111100101110111011000110110001011010110110111011010011111111100101110111011100

Configuration	
Card number	1
IP Address	127.0.0.1
Port	33234
Sampling time	0
Timeout	5
Counter	

Analysis name
Analysis name

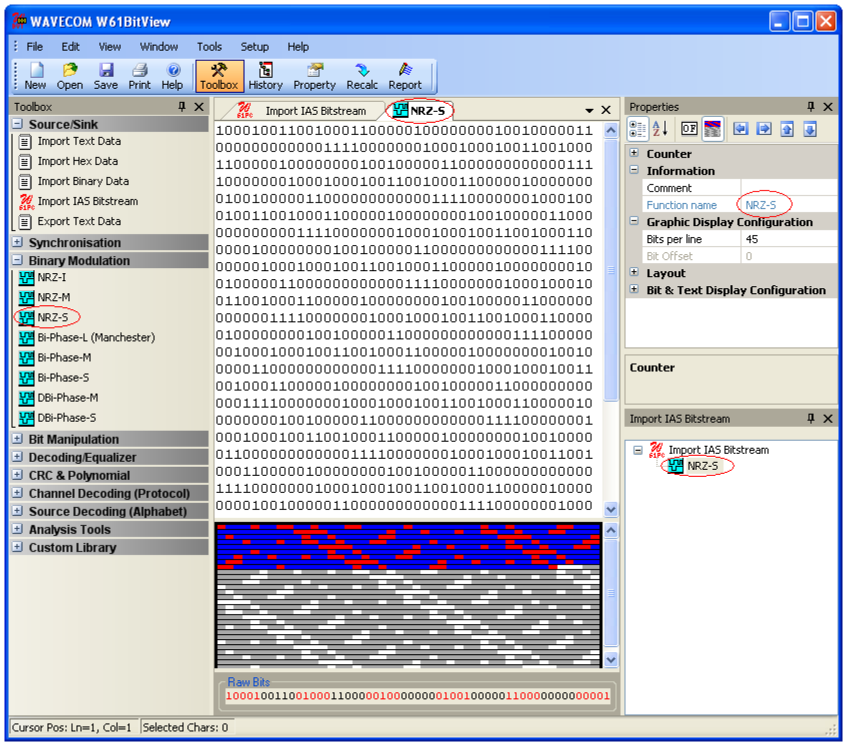
Import IAS Bitstream

Cursor Pos: Ln=22, Col=61 | Selected Chars: 0

Menu

Bit Stream Processing

An imported bit stream may be processed using any of the functions found in the library.


WAVECOM W61BitView
File Edit View Window Tools Setup Help
New Open Save Print Help Toolbox History Property Recalc Report
Toolbox
Import IAS Bitstream
NRZ-S
Properties
Import IAS Bitstream
Raw Bits

Toolbox
- Source/Sink
  - Import Text Data
  - Import Hex Data
  - Import Binary Data
  - Import IAS Bitstream
  - Export Text Data
- Synchronisation
- Binary Modulation
  - NRZ-I
  - NRZ-M
  - NRZ-S
  - Bi-Phase-L (Manchester)
  - Bi-Phase-M
  - Bi-Phase-S
  - DBi-Phase-M
  - DBi-Phase-S
- Bit Manipulation
- Decoding/Equalizer
- CRC & Polynomial
- Channel Decoding (Protocol)
- Source Decoding (Alphabet)
- Analysis Tools
- Custom Library

```
1000100110010001100000100000000010010000011
0000000000001110000000010001000100110010000
1100000100000000100100000110000000000000111
1000000010001000100110010001100000100000000
0100100000110000000000000111100000000100010 0
0100110010001100000100000000010010000011000
0000000001111000000001000100010011001000110
0000100000000010010000011000000000000011100
0000010001000100110010001100000100000000010
0100000110000000000000111100000000100010010
0110010001100000100000000010010000011000000
0000001111000000001000100010011001000110000
0100000000010010000011000000000000011100000
0010001000100110010001100000100000000010010
0000110000000000000111100000000100010001001 1
0010001100000100000000010010000011000000000
0001111000000001000100010011001000110000010
0000000100100000110000000000000111100000001
0001000100110010001100000100000000010010000
0110000000000000111100000000100010001001100 1
0001100000100000000010010000011000000000000
1111000000001000100010011001000110000010000
0000100100000110000000000000111100000001000
```

Raw Bits: 10001001100100011000001000000000100100000110000000000001

Counter	
Information	
Comment	
Function name	NRZ-S
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

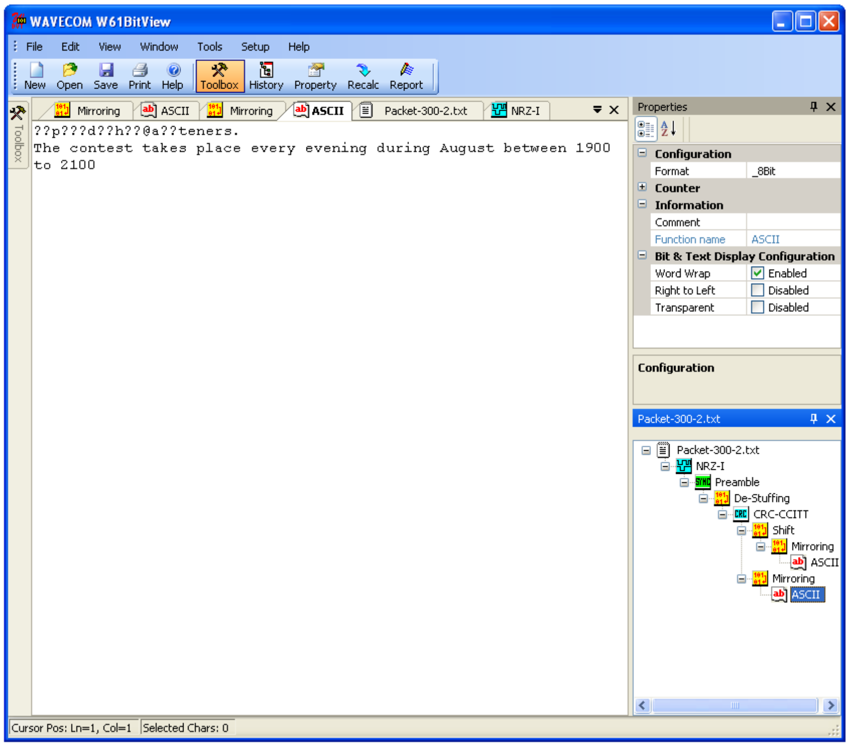
Counter

- Import IAS Bitstream
  - NRZ-S

Cursor Pos: Ln=1, Col=1 | Selected Chars: 0

The position of the cursor (line, column and index) in the bit display is continuously indicated in the bottom line of the application window. To select a number of bits, press and hold the left mouse button. The selected bits are marked in blue and the number of selected bits, and the decimal values of these bits interpreted as big-endian or little-endian values are displayed in the bottom line. Left-clicking anywhere in the display window will cancel the selection.

The processed bit stream is shown in a new document window. All document windows are shown as tabbed windows.


WAVECOM W61BitView
File Edit View Window Tools Setup Help
New Open Save Print Help Toolbox History Property Recalc Report
Mirroring ASCII Mirroring ASCII Packet-300-2.txt NRZ-I
Toolbox
Properties
Packet-300-2.txt
NRZ-I
Preamble
De-Stuffing
CRC-CCITT
Shift
Mirroring
ASCII
Mirroring
ASCII

```
??p???d??h??@a??teners.
The contest takes place every evening during August between 1900
to 2100
```

Configuration	
Format	_8Bit
Counter	
Information	
Comment	
Function name	ASCII
Bit & Text Display Configuration	
Word Wrap	✓ Enabled
Right to Left	Disabled
Transparent	Disabled

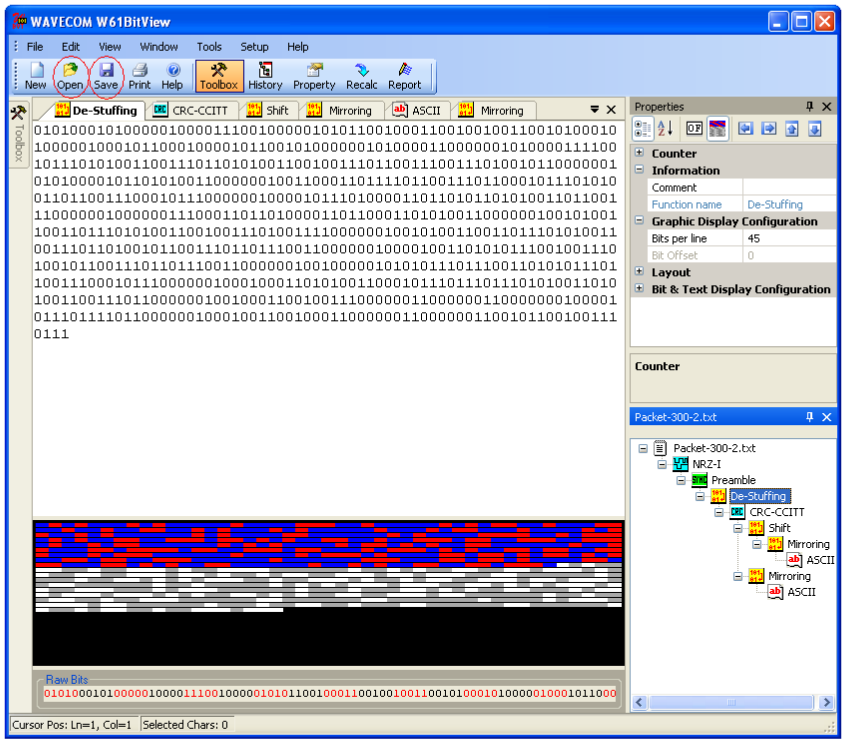
Configuration

Cursor Pos: Ln=1, Col=1 Selected Chars: 0

Analysis Sets

Functions and data may be combined to form a so-called **Analysis Set**, which contains an imported bit stream and the configured functions applied to the bit stream. The user may define and create different analysis paths, as may be seen in the **History Explorer**. The imported bit stream is processed according to the configuration settings of the selected functions.

Using the **Save** button in the toolbar, **Analysis Sets** may be saved in an XML file. Using the **Open** button, an **Analysis Set** may be reloaded at any time.



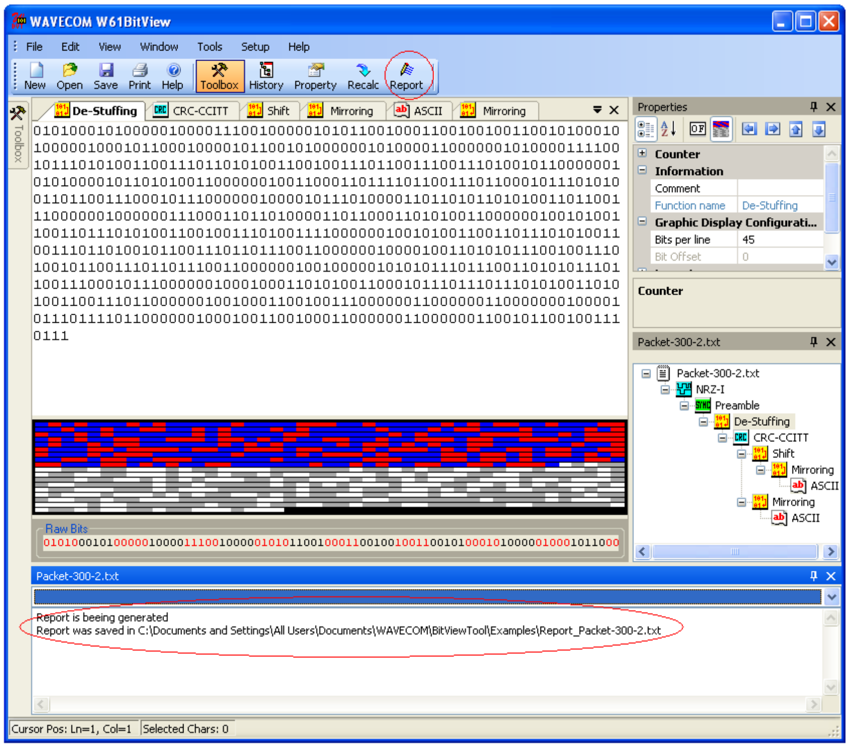
```
0101000101000000100001110010000001010110010001100100100110010100010
10000010001011000100001011001010000001010000110000001010000111100
1011101010011001110110101001100100111011001110011101001011000001
01010000101101010011000000100110001101111011001110110001011101010
011011001110001011100000001000010111010000110110101101010011011001
1100000010000000111000110110100001101100011010100110000000100101001
100110111010100110010011101001111000000010010100110011011101010011
001110110100101100111011011100110000000100001001101010111001001110
1001011001110110111001100000001001000000101010111011100110101011101
100111000101110000000100010001101010011000101110111011101010011010
100110011101100000001001000110010011100000011000000110000000100001
0111011110110000000100010011001000110000000110000001100101100100111
0111
```

Raw Bits: 0101000101000001000011100100000101011001000110010010011001010001010000010001011000

Function name: De-Stuffing
Bits per line: 45
Bit Offset: 0

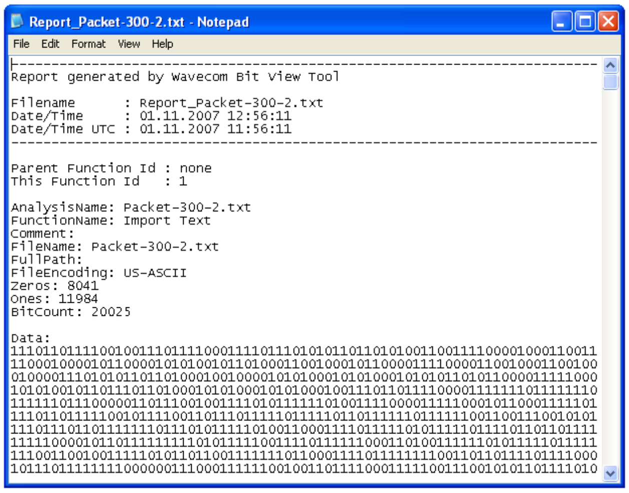
Reports

Using the **Report** button a complete **Analysis Set** may be generated and saved as a **text** file or a **XML** file.



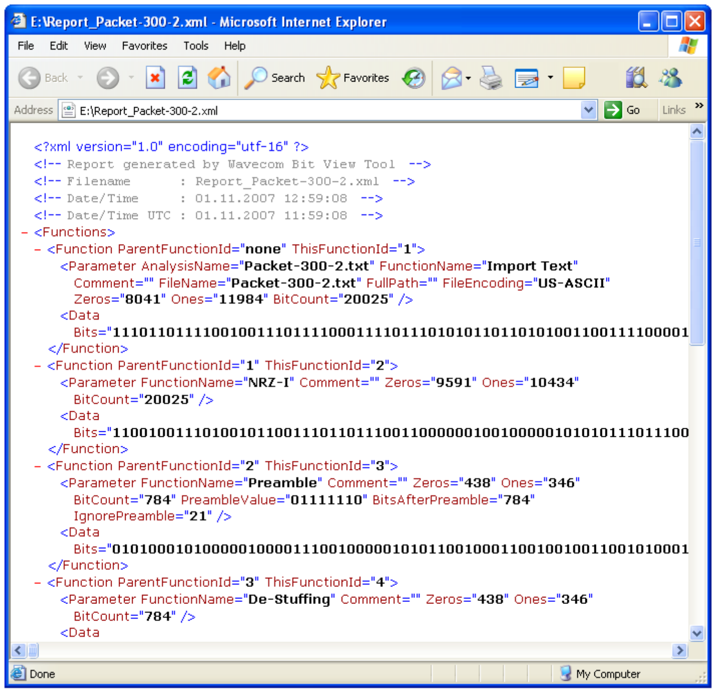
```
Report is beeing generated
Report was saved in C:\Documents and Settings\All Users\Documents\WAVECOM\BitViewTool\Examples\Report_Packet-300-2.txt
```

Example of a report stored in a **text** file.



```
--------------------------------------------------------------------------
Report generated by Wavecom Bit View Tool

Filename      : Report_Packet-300-2.txt
Date/Time     : 01.11.2007 12:56:11
Date/Time UTC : 01.11.2007 11:56:11
--------------------------------------------------------------------------

Parent Function Id : none
This Function Id   : 1

AnalysisName: Packet-300-2.txt
FunctionName: Import Text
Comment:
FileName: Packet-300-2.txt
FullPath:
FileEncoding: US-ASCII
Zeros: 8041
Ones: 11984
BitCount: 20025

Data:
1110110111100100111011110001111011101010110110101001100111100001000110011
1100010000101100001010100101101000110010001011000011110000110010001100100
0100001110101011011010001001000010101000101010001010101101011000011111000
1010100101101110110100010101000101010001001110110111100001111111011111110
1111110111000001101110010011110101111110100111100001111100010110001111101
1110110111110010111100110111011111011111011011111101111110011001110010101
1110111011011111110111010111110100110001111011111010111110111101101101111
1111100001011011111111101011111001111011111100011010011111101011111011111
1110011001001111101011011001111111011000111101111111100110110111101111000
1011101111111110000000111000111111001001101111000111110011100101011011110110
```

Example of a report stored in a **XML** file.


E:\Report_Packet-300-2.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address E:\Report_Packet-300-2.xml
Go Links
Done
My Computer

```
  <?xml version="1.0" encoding="utf-16" ?>
  <!-- Report generated by Wavecom Bit View Tool  -->
  <!-- Filename      : Report_Packet-300-2.xml  -->
  <!-- Date/Time     : 01.11.2007 12:59:08  -->
  <!-- Date/Time UTC : 01.11.2007 11:59:08  -->
- <Functions>
  - <Function ParentFunctionId="none" ThisFunctionId="1">
      <Parameter AnalysisName="Packet-300-2.txt" FunctionName="Import Text"
        Comment="" FileName="Packet-300-2.txt" FullPath="" FileEncoding="US-ASCII"
        Zeros="8041" Ones="11984" BitCount="20025" />
      <Data
        Bits="11101101111001001110111100011110111010101101101010011001111000001
    </Function>
  - <Function ParentFunctionId="1" ThisFunctionId="2">
      <Parameter FunctionName="NRZ-I" Comment="" Zeros="9591" Ones="10434"
        BitCount="20025" />
      <Data
        Bits="11001001110100101100111011011100110000001001000001010101110111100
    </Function>
  - <Function ParentFunctionId="2" ThisFunctionId="3">
      <Parameter FunctionName="Preamble" Comment="" Zeros="438" Ones="346"
        BitCount="784" PreambleValue="01111110" BitsAfterPreamble="784"
        IgnorePreamble="21" />
      <Data
        Bits="01010001010000010000111001000001010110010001100100100110010100001
    </Function>
  - <Function ParentFunctionId="3" ThisFunctionId="4">
      <Parameter FunctionName="De-Stuffing" Comment="" Zeros="438" Ones="346"
        BitCount="784" />
      <Data
```

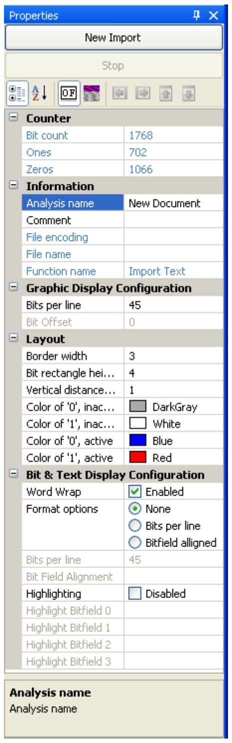
Properties Window

The parameters in the **Properties** window are grouped into different categories providing the operator with information about actual parameter settings and - more important - allowing the operator to configure each function and to add comments.

Detailed information about the selected parameter is displayed in the **Comments** area of the **Properties** window.

In the **Counter** category information on the recorded bit stream, i.e., **Bit count**, number of logical **Ones** and number of logical **Zeros** is found.

The **Information** category contains information on the selected source or analysis set, i.e., **Analysis name**, **Comments** (user input possible), **File name** (information only) and **Function name** (information only). Clicking the function name displays a brief description of the function.


Properties

New Import

Stop

Counter	
Bit count	1768
Ones	702
Zeros	1066
Information	
Analysis name	New Document
Comment	
File encoding	
File name	
Function name	Import Text
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Border width	3
Bit rectangle hei…	4
Vertical distance…	1
Color of '0', inac…	▩ DarkGray
Color of '1', inac…	☐ White
Color of '0', active	🟦 Blue
Color of '1', active	🟥 Red
Bit & Text Display Configuration	
Word Wrap	✓ Enabled
Format options	[✓ None] Bits per line Bitfield alligned
Bits per line	45
Bit Field Alignment	
Highlighting	☐ Disabled
Highlight Bitfield 0	
Highlight Bitfield 1	
Highlight Bitfield 2	
Highlight Bitfield 3	

Analysis name
Analysis name

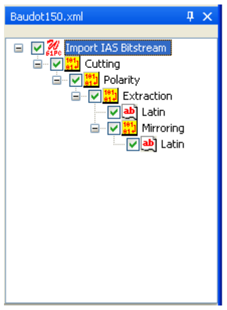
History Explorer Window

The **History Explorer** window provides a quick overview of the current analyzing process. It allows the operator to try out different function paths with different parameter settings and enables instant comparison of the results of these trials.

Functions may be re-arranged and deleted using the mouse pointer (drag-and-drop with left mouse button pressed and held) in combination with the modifier buttons (CTRL, ALT, SHIFT) or the mouse right-click menu.

Keyboard button	Function
No keyboard button pressed	Move and add dragged function
CTRL pressed	Copy and add dragged function
ALT pressed	Move and add dragged function plus all sub-functions
SHIFT pressed	Move and insert dragged function
CTRL and ALT pressed	Copy and add dragged function plus all sub-functions
CTRL and SHIFT pressed	Copy and insert dragged function
SHIFT and ALT pressed	Move and insert dragged function plus all sub-functions
SHIFT/CTRL/ALT pressed	Copy and insert dragged function plus all sub-functions

When one of the modifier buttons is pressed while dragging, detailed information is displayed on top of the window,


Baudot150.xml

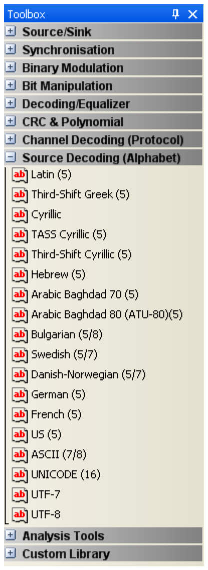
- ✓ Import IAS Bitstream
  - ✓ Cutting
    - ✓ Polarity
      - ✓ Extraction
        - ✓ Latin
        - ✓ Mirroring
          - ✓ Latin

If the **History Explorer** window has been hidden a list of functions applied is still available by clicking the **arrow** button in the top right corner of a document window.

Toolbox Window

The **Toolbox** menu is divided into libraries, and each library contains one or more functions.

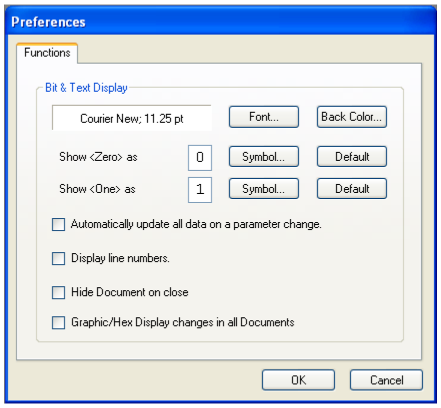
Note: The **Custom Library** is not visible unless a custom function has been added.
The **Analysis Tools** are not added to the **History Explore**r tree and are not persistently stored.



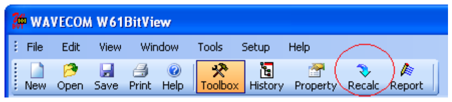
Preferences

The **Preferences** dialog box may be selected from the **Setup** menu.

Substitution symbols for logical zero and logical one may be directly edited or selected using the appropriate **Symbol** button.



If **Automatically update all data on a parameter change** is ticked, all functions are automatically recalculated when the operator changes a parameter. Uncheck the tick box if this behavior is not desirable, and use the **Recalc** button in case a recalculation is necessary.



If **Hide document on close** is ticked, a document is hidden when closed, but remains in the **History Explorer**. Clicking the function associated with the document in the **History Explorer** will make the document visible again.

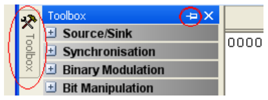
To remove the function completely, select the appropriate function in the **History Explorer** and press the **Delete** key on your keyboard.

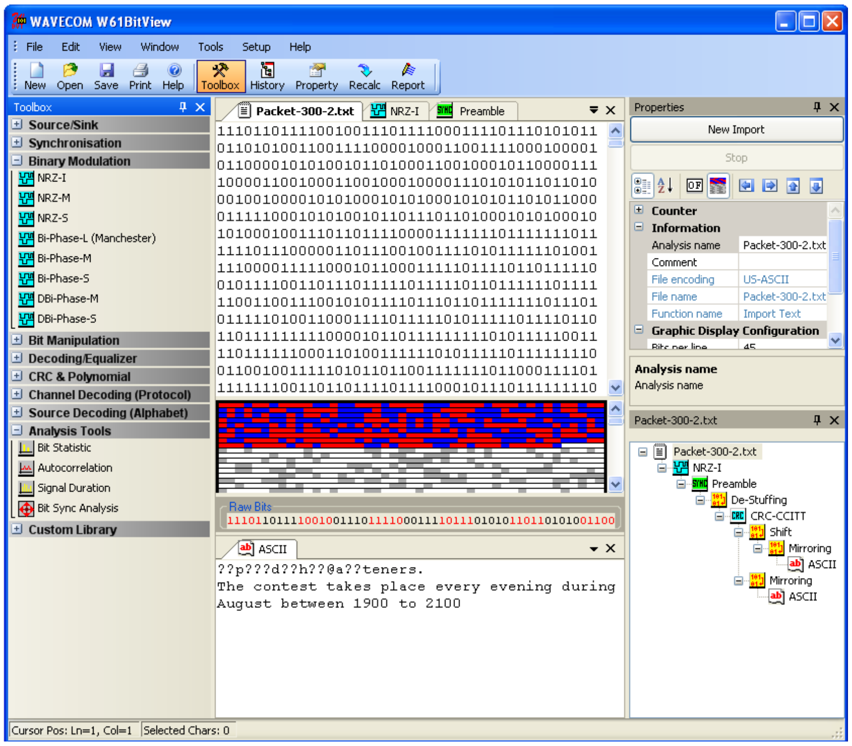
If this option is not checked, documents are completely removed when closed. Closing the root document will close and remove all other functions and their associated documents.

If **Graphic/Hex display changes in all documents** option is selected, all the documents will have the same display settings, i.e., if the display is changed from graphic to hex in one document, and all other documents will change their display type as well.

Layout Settings

At run time, the user can drag and drop all windows to re-arrange them according to the preferred layout when not in **auto-hide** mode (map pin icon on window top line must be in vertical position). In the selected window press and hold the left mouse button and drag the window to the position you want it be in or just double-click the window and it will detach itself.





```
1110110111100100111011110001111011101010 11
0110101001100111100001000110011110001000 01
0110000101010010110100011001000101100001 11
1000011001000110010001000011101010110110 10
0010010000101010001010100010101011010110 00
0111110001010100101101110110100010101000 10
1010001001110110111100001111110111111110 11
1111011100000110111001001111010111111010 01
1110000111110001011000111110111101101111 10
0101111001101110111110111110110111111011 11
1100110011100101011110111011011111111011 01
0111110100110001111011111010111110111101 10
1101111111110000101101111111101011110011
1101111110001101001111110101111101111111 10
0110010011111010110110011111110110001111 01
1111111001101101111011110001011101111111 10
```

Raw Bits: 11101101111001001110111100011110111010101101101010 01100

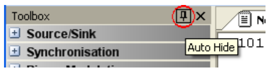
```
??p???d??h??@a??teners.
The contest takes place every evening during
August between 1900 to 2100
```

Counter	
Information	
Analysis name	Packet-300-2.txt
Comment	
File encoding	US-ASCII
File name	Packet-300-2.txt
Function name	Import Text
Graphic Display Configuration	
Bits per line	45

Analysis name
Analysis name

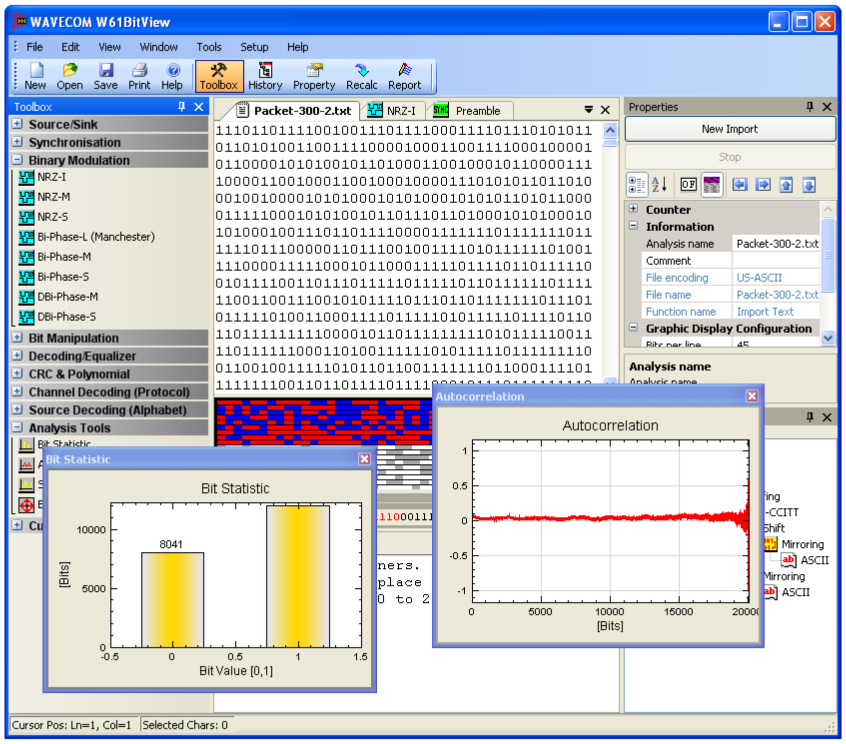
Cursor Pos: Ln=1, Col=1 | Selected Chars: 0

In addition the **Parameters**, **History Explorer** and **Toolbox** windows use **auto-hide** functionality. To activate auto-hide for a window, click the map pin icon on the upper window line. The icon will change from vertical to horizontal position.



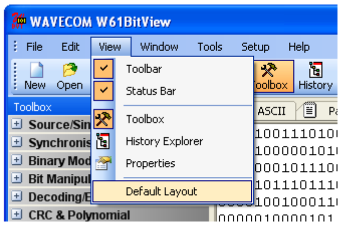
The window is now hidden and a tab with its name appears on the side of the application window. To make the window appear, move the cursor to the tab. To hide the window again, double-click on the window.

To restore the application default layout, in the menu bar click **View > Default Layout**.

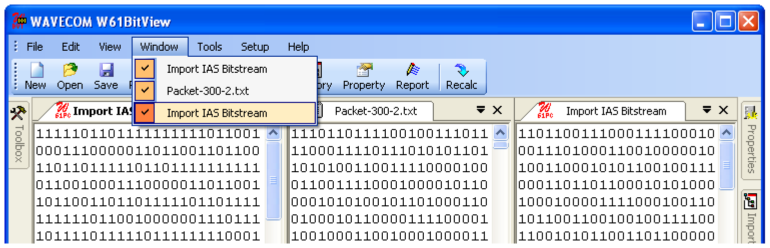


Analysis name	Packet-300-2.txt
Comment	
File encoding	US-ASCII
File name	Packet-300-2.txt
Function name	Import Text

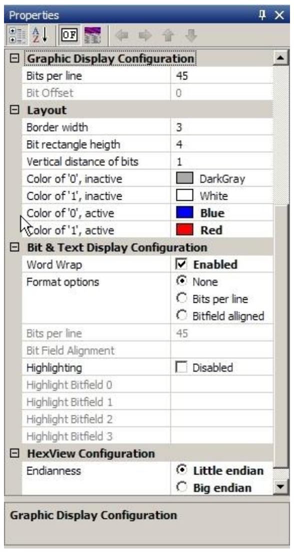
Use **View > Default Layout** to use or restore the default layout of the windows.



More than one **Analysis Set** may be active at a time. Using the **Window** menu allows the user to show or hide these **Analysis Sets**.



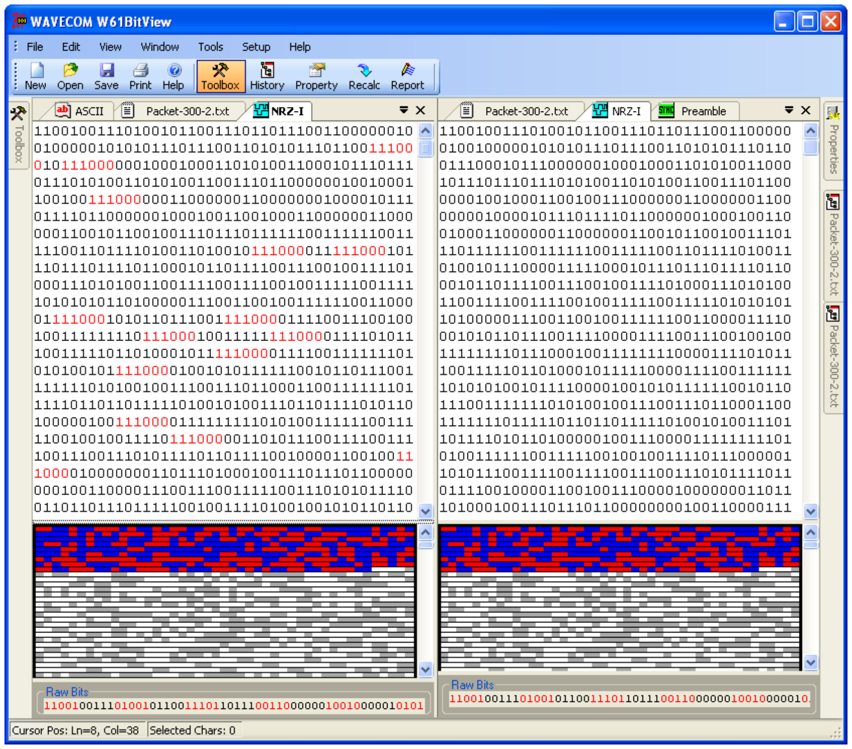
The layout and display settings can be configured from the upper part of the **Properties** window.



Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Border width	3
Bit rectangle heigth	4
Vertical distance of bits	1
Color of '0', inactive	▨ DarkGray
Color of '1', inactive	☐ White
Color of '0', active	🟦 Blue
Color of '1', active	🟥 Red
Bit & Text Display Configuration	
Word Wrap	☑ Enabled
Format options	● None ○ Bits per line ○ Bitfield alligned
Bits per line	45
Bit Field Alignment	
Highlighting	☐ Disabled
Highlight Bitfield 0	
Highlight Bitfield 1	
Highlight Bitfield 2	
Highlight Bitfield 3	
HexView Configuration	
Endianness	● Little endian ○ Big endian

Graphic Display Configuration

Using the **Bit & Text Display Configuration** category in the **Properties** window enables the operator to use different display format options listed below.


WAVECOM W61BitView
File Edit View Window Tools Setup Help
New Open Save Print Help Toolbox History Property Recalc Report
Toolbox
ASCII
Packet-300-2.txt
NRZ-I
Preamble
Properties
Packet-300-2.txt
Raw Bits

```
110010011101001011001110110111001100000010
0100000010101011101110011010101110110011100
010111000000100010001101010011000101110111
011101010011010100110011101100000010010001
1001001110000001100000001100000001000010111
011110110000001000100110010001100000011000
000110010110010011101110111111001111110011
111001101111010011010010111000011111000101
110111011110110001011011110011100100111101
000111010100110011110011110010011111001111
101010101101000001110011001001111110011000
011110001010110111001111000011110011100100
100111111110111000100111111110000111101011
100111110110100010111110000111100111111101
010100101111000010010101111110010110111001
111111010100100111001110110001100111111101
111101101101111101001010011101101111010110
100000100111000011111111101010011111100111
110010010011110111000001101011100111100111
100111001110101111011011110010000110010011
100001000000011011101000100111011101100000
000100110000111001110011111001110101011110
011011011101111100100111101001001010110110
```

```
110010011101001011001110110111001100000
010010000001010101110111001101010101110110
011100010111000000100010001101010011000
101110111011101010011010100110011101100
000010010001100100111000000110000001100
000001000010111011110110000001000100110
010001100000011000000110010110010011101
110111111001111110011111001101111010011
010010111000011111000101110111011110110
001011011110011100100111101000111010100
110011110011110010011111001111101010101
101000000111001100100111111001100001111 0
001010110111001111000011110011100100100
111111101110001001111111100001111 01011
100111110110100010111110000111100111111
101010100101110000100101011111100101 10
111001111111010100100111001110110001100
111111101111101101101111101001010011101
101111010110100000100111000011111111101
010011111100111110010010011110111000001
101011100111001110011100111010111101 1
011110010000110010011100001000000011011
101000100111011101100000000100110000111
```

```
11001001110100101100111011011100110000001001000001 0101
110010011101001011001110110111001100000010010000010.
```

Cursor Pos: Ln=8, Col=38 Selected Chars: 0

Word Wrap

Checking the **Word Wrap Enabled** tick box enables word wrapping in the bit and text document.

Bits per line

This parameter allows displaying a specific number of bits per line. Choose the **Bits per line** radio button under format options to enable this feature.

Bitfield alignment

Whenever the specified bit pattern is found in the bit stream, a new line is started, i.e., a line break is inserted. Choose the **Bitfield aligned** radio button under format options to enable this feature.

Highlighting

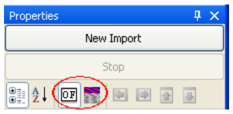
The bit stream is searched for a specific bit pattern and when found the pattern is marked. A maximum of four different search patterns are possible. Check the **Highlighting** tick box to enable this feature and enter the search patterns in the appropriate text fields.

Display options

In addition to the document window(s), two additional views of the bit stream being analyzed are available at the bottom of the document window(s): one is a graphical display and the other one a hexadecimal display.

Graphic Display

A graphic display is associated with the bit stream and may be selected from the top of the **Properties** window clicking the **Show Graphics bit view** icon.



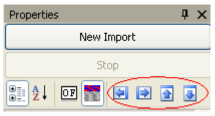
The vertical size of the graphic display can be changed by dragging its top border.

The **Layout** category let you change the appearance of the graphic display.

In the **Graphic Display Configuration** category the number of bits per line can be set. This feature can be used to find periodic bit patterns in the bit stream by changing the number of bits per line until a repeating bit pattern is visible. It is much easier to find those patterns using the graphic display than to use the hex display.

The arrow buttons on top of the **Properties** window are designed to move the active selection in the graphic display. The selected bits are displayed as **Raw Bits** in the bottom of the graphic display.

When the cursor is placed in the graphic display, the cursor changes into a cross hair and a context window indicates the position of the cursor (offset, row and column). Holding the right mouse button allows the user to zoom in on the selected area. Right-clicking will make an **Unzoom** button appear. Use this button to un-zoom the selected area.

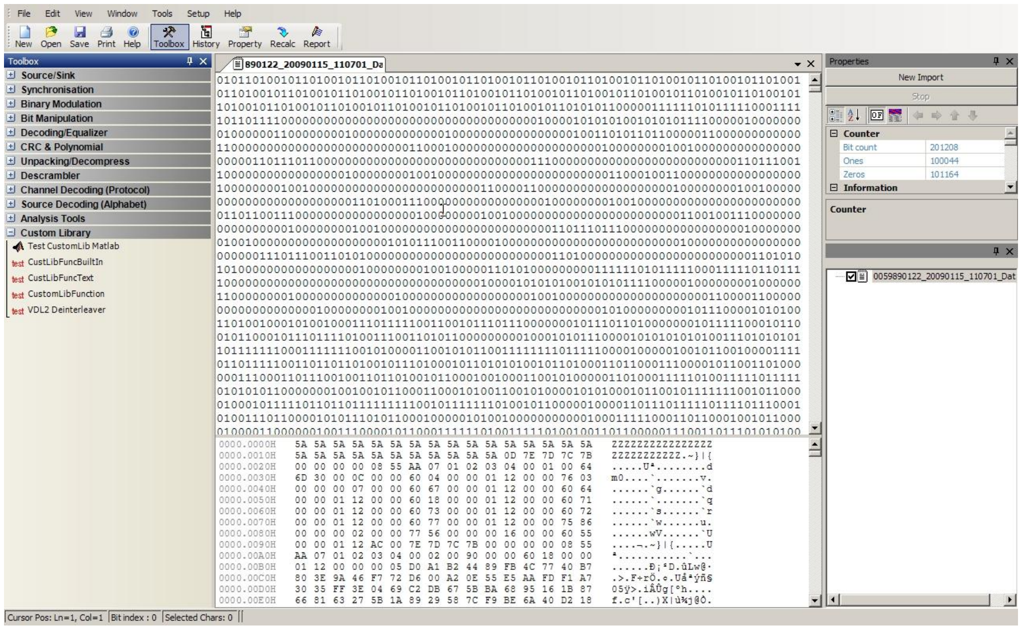


It is possible to zoom into the graphic display. Hold the left mouse button down and select the area that is to be expanded. A right click on the graphic display shows the context menu for un-zooming the view.

Hex Display

A hex display may be selected by pressing the **Show Hex view** button on top of the **Properties** window. This display option offers a standard hex dump layout consisting of an offset in hexadecimal notation, hexadecimal characters separated by a space and finally the data in ASCII characters.

The interpretation of 8-bit frames can be toggled between **Little** and **Big endian** in **Hex View Configuration, Endianness**, i.e., "00101100" is displayed as 0x34 when interpreted as little endian and as 0x2c when interpreted as big endian.



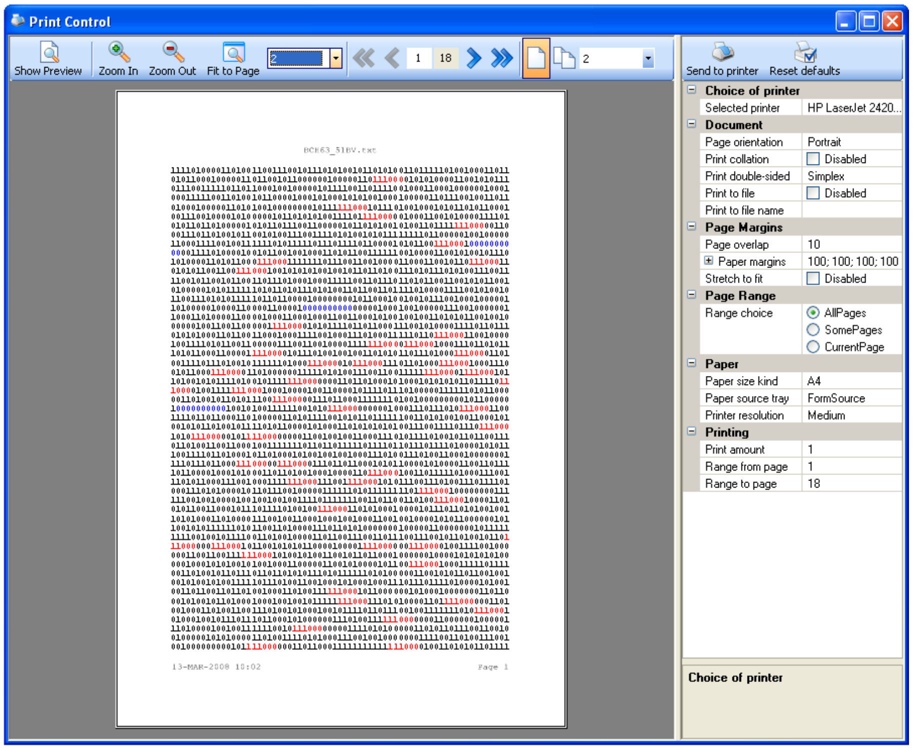
```
0101101001011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101101001
0110100101101001011010010110100101101001011010010110100101101001011010010110100101101001011010010110100101
1010010110100101101001011010010110100101101001011010010110101011000001111110101111110001111
1011011100000000000000000000000000000000010000101010100101010111100000100000000
01000000110000000010000000000000010000000000000000000000100110101101100000110000000000000
1100000000000000000000000011000100000000000000000000000010000000010010000000000000000
0000011011101100000000000000000000000000000001110000000000000000000000000000110111001
1000000000000000010000000010010000000000000000000000000000011000100110000000000000000000
1000000001001000000000000000000000000000110000110000000000000000000000100000000100100000
0000000000000000000110100011100000000000000000010000000010010000000000000000000000000000
0110110011100000000000000000000010000000001001000000000000000000000000000001100100111000000
0000000000100000000010010000000000000000000000000001101110111000000000000000000001000000
01001000000000000000000000101011100110000100000000000000000000000000010000000000000000
0000001110111001101010000000000000000000000000000011010000000000000000000000000001101010
1010000000000000000100000000100100000110101000000000011111101011111000111110110111
10000000000000000000000000000000000100001010101001010101111000001000000001000000
1100000000100000000000000001000000000000000000000100100000000000000000000000011000011000000
0000000000000010000000010010000000000000000000000000000001010000000000101110000101010 0
1101001000101001000111011111001100101110111000000001011101101000000001011111100010110
0101100010111011110100111001101011000000000100010101110000101010101010011101010101
10111111100011111110010100001100101011001111111101111110000100000100101100100001111
0110111110011011011010010111010001011010101001011010001101100011100001011001101000
0001110001101110010011011010010110001001000110010100000110100011111010011111011111
0101010110000000100100101100011000101001100101000010101000101100101111111001011000
0100010111110110110111111111001011111110100101100000100001101110111110111101110001
0100111011000010101110101100010000010100100000000000010001111100011011000100101100 0
0100001100000010011100001011000111111010011111010010011011000001110011011101010100
```

```
0000.0000H    5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A    ZZZZZZZZZZZZZZZZ
0000.0010H    5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 0D 7E 7D 7C 7B    ZZZZZZZZZZZ.~}|{
0000.0020H    00 00 00 00 08 55 AA 07 01 02 03 04 00 01 00 64    .....Uª........d
0000.0030H    6D 30 00 0C 00 00 60 04 00 00 01 12 00 00 76 03    m0....`.......v.
0000.0040H    00 00 00 07 00 00 60 67 00 00 01 12 00 00 60 64    ......`g......`d
0000.0050H    00 00 01 12 00 00 60 18 00 00 01 12 00 00 60 71    ......`.......`q
0000.0060H    00 00 01 12 00 00 60 73 00 00 01 12 00 00 60 72    ......`s......`r
0000.0070H    00 00 01 12 00 00 60 77 00 00 01 12 00 00 75 86    ......`w......u.
0000.0080H    00 00 00 02 00 00 77 56 00 00 00 16 00 00 60 55    ......wV......`U
0000.0090H    00 00 01 12 AC 00 7E 7D 7C 7B 00 00 00 00 08 55    ....¬.~}|{.....U
0000.00A0H    AA 07 01 02 03 04 00 02 00 90 00 00 60 18 00 00    ª...........`...
0000.00B0H    01 12 00 00 00 05 D0 A1 B2 44 89 FB 4C 77 40 B7    ......Ð¡²D.ûLw@·
0000.00C0H    80 3E 9A 46 F7 72 D6 00 A2 0E 55 E5 AA FD F1 A7    .>.F÷rÖ.¢.UåªýñS
0000.00D0H    30 35 FF 3E 04 69 C2 DB 67 5B BA 68 95 16 1B 87    05ÿ>.iÂÛg[ºh....
0000.00E0H    66 81 63 27 5B 1A 89 29 58 7C F9 BE 6A 40 D2 18    f.c'[..)X|ù¾j@Ò.
```

Bit count	201208
Ones	100044
Zeros	101164

Printer Dialog

The printer dialog is used for print preview and the printer settings. Use it by clicking the **Print** button or by selecting the menu entry **File > Print**. All layout settings, i.e., highlighting, alignment or bits per line, are supported.



Choice of printer	
Selected printer	HP LaserJet 2420...
Document	
Page orientation	Portrait
Print collation	☐ Disabled
Print double-sided	Simplex
Print to file	☐ Disabled
Print to file name	
Page Margins	
Page overlap	10
Paper margins	100; 100; 100; 100
Stretch to fit	☐ Disabled
Page Range	
Range choice	[✓ AllPages] SomePages CurrentPage
Paper	
Paper size kind	A4
Paper source tray	FormSource
Printer resolution	Medium
Printing	
Print amount	1
Range from page	1
Range to page	18

Choice of printer

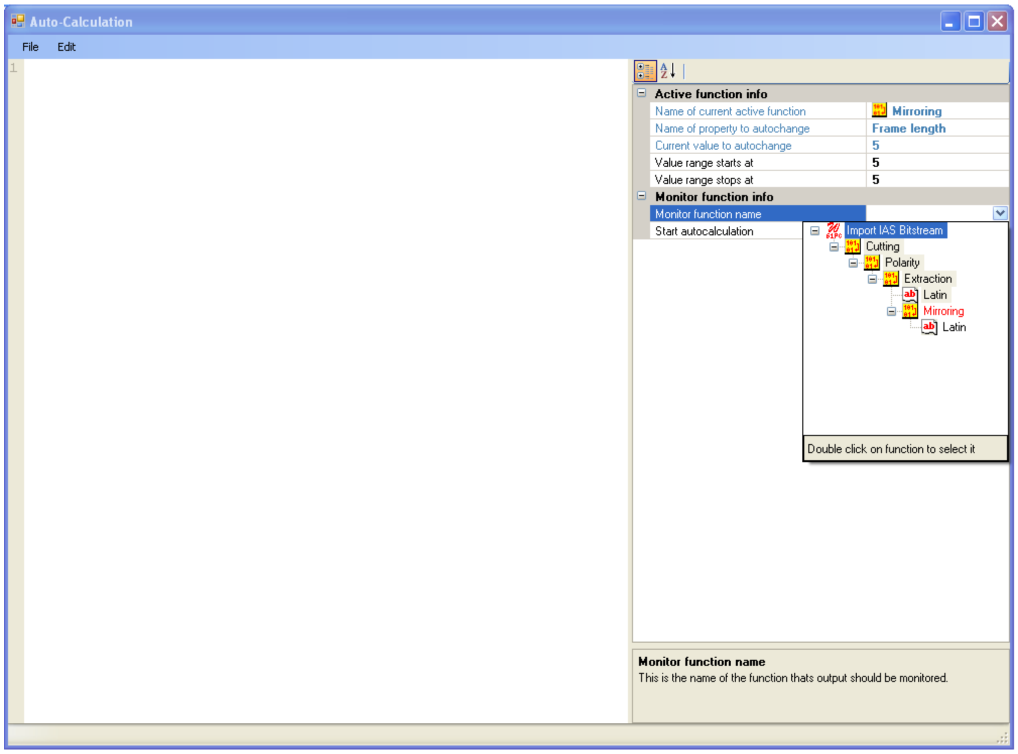
Function Library

Common Functions

Auto-Calculation

To test a hypothesis regarding the properties of a bit stream, the operator may wish to test a property within a specific range of values. This task can either be conducted by changing the parameter value manually and then checking the outputs one by one or it can be performed automatically. That is where Auto-Calculation comes into play.

A right click on the parameter name (not the value) in the parameters window opens a context menu. Click **Reset** to change parameters to their default values or click **AutoCalc Setup** to open the **Auto-Calculation** window:



Active function info

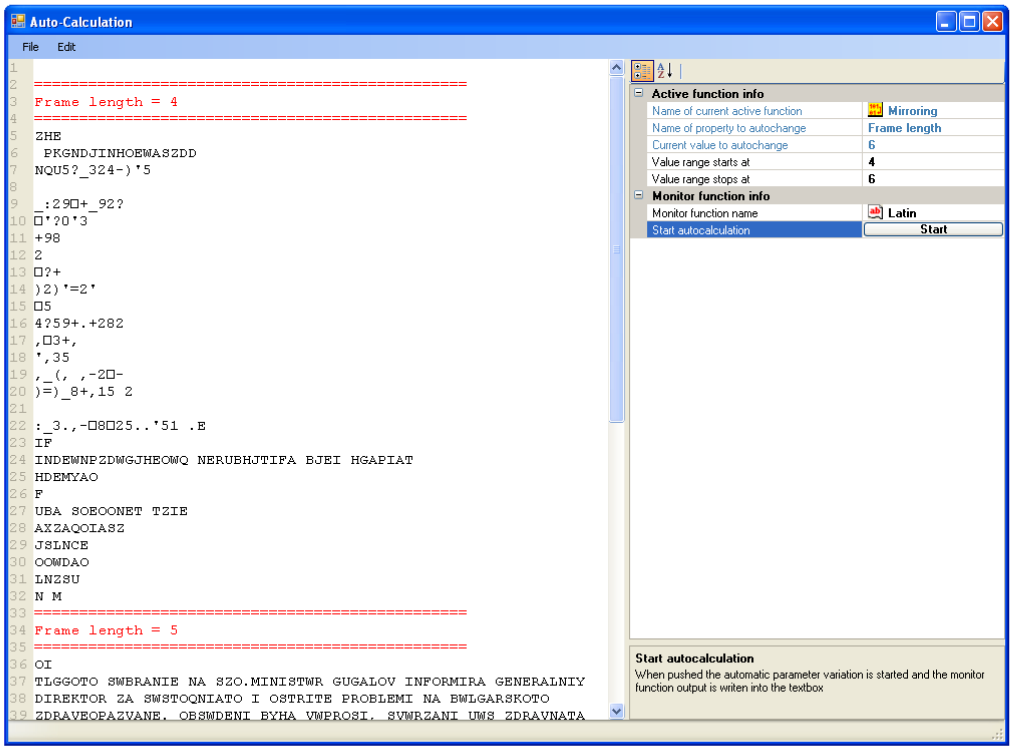
Name of current active function	Mirroring
Name of property to autochange	Frame length
Current value to autochange	5
Value range starts at	5
Value range stops at	5

Monitor function info

Monitor function name	
Start autocalculation	

Monitor function name
This is the name of the function thats output should be monitored.

Start and stop values for the selected parameter must be chosen using the **Value range starts at** and **Value range stops at** parameters. The auto-calculation function then needs to be told which function will monitor the calculation output by use the function drop-down menu of the **Monitor function name** parameter – the function MUST be lower in the function tree as displayed in the **History** window. Automatic calculation can now be started by clicking the **Start** button.



```

==================================================
Frame length = 4
==================================================
ZHE
 PKGNDJINHOEWASZDD
NQU5?_324-)'5

_:29□+_92?
□'?0'3
+98
2
□?+
)2)'=2'
□5
4?59+.+282
,□3+,
',35
,_(, ,-2□-
)=)_8+,15 2

:_3.,-□8□25..'51 .E
IF
INDEWNPZDWGJHEOWQ NERUBHJTIFA BJEI HGAPIAT
HDEMYAO
F
UBA SOEOONET TZIE
AXZAQOIASZ
JSLNCE
OOWDAO
LNZSU
N M
==================================================
Frame length = 5
==================================================
OI
TLGGOTO SWBRANIE NA SZO.MINISTWR GUGALOV INFORMIRA GENERALNIY
DIREKTOR ZA SWSTOQNIATO I OSTRITE PROBLEMI NA BWLGARSKOTO
ZDRAVEOPAZVANE. OBSWDENI BYHA VWPROSI. SVWRZANI UWS ZDRAVNATA
```

Active function info

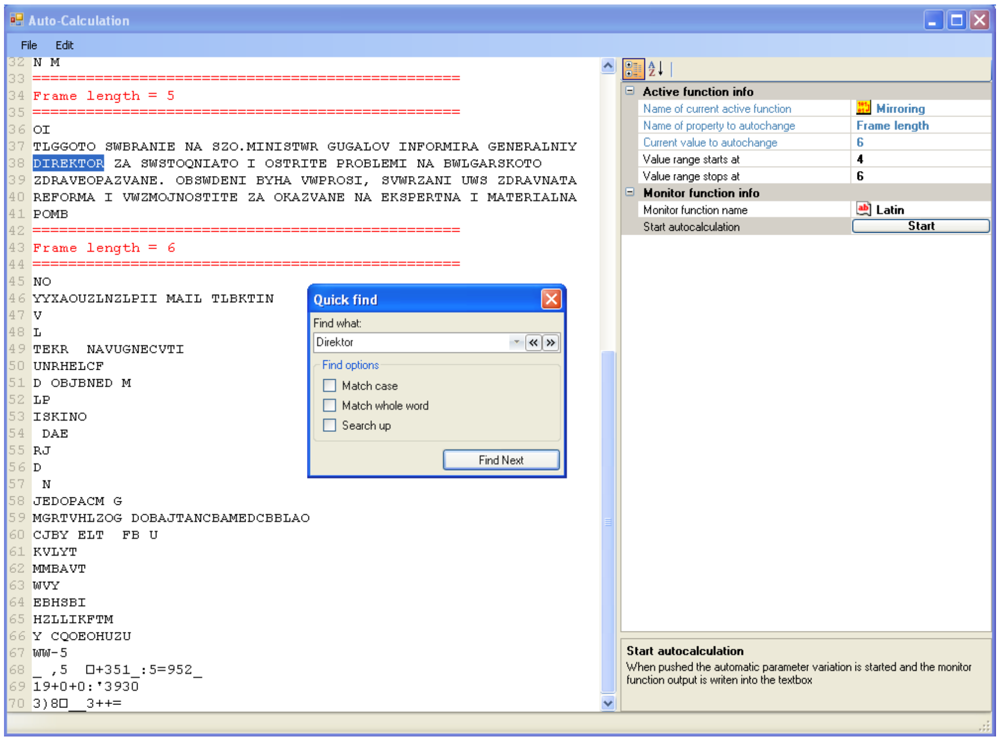
Name of current active function	Mirroring
Name of property to autochange	Frame length
Current value to autochange	6
Value range starts at	4
Value range stops at	6

Monitor function info

Monitor function name	Latin
Start autocalculation	Start

Start autocalculation
When pushed the automatic parameter variation is started and the monitor function output is writen into the textbox

The result of the auto-calculation is displayed in the left side of the **Auto-Calculation** window. **Edit > Quick find** can be used for searching the auto-calculated output.



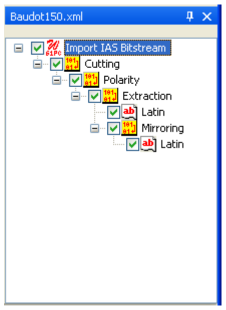
```
N M
================================================
Frame length = 5
================================================
OI
TLGGOTO SWBRANIE NA SZO.MINISTWR GUGALOV INFORMIRA GENERALNIY
DIREKTOR ZA SWSTOQNIATO I OSTRITE PROBLEMI NA BWLGARSKOTO
ZDRAVEOPAZVANE. OBSWDENI BYHA VWPROSI, SVWRZANI UWS ZDRAVNATA
REFORMA I VWZMOJNOSTITE ZA OKAZVANE NA EKSPERTNA I MATERIALNA
POMB
================================================
Frame length = 6
================================================
NO
YYXAOUZLNZLPII MAIL TLBKTIN
V
L
TEKR  NAVUGNECVTI
UNRHELCF
D OBJBNED M
LP
ISKINO
 DAE
RJ
D
 N
JEDOPACM G
MGRTVHLZOG DOBAJTANCBAMEDCBBLAO
CJBY ELT  FB U
KVLYT
MMBAVT
WVY
EBHSBI
HZLLIKFTM
Y CQOEOHUZU
WW-5
_ ,5  □+351_:5=952_
19+0+0:'3930
3)8□__3++=
```

Active function info	
Name of current active function	Mirroring
Name of property to autochange	Frame length
Current value to autochange	6
Value range starts at	4
Value range stops at	6
Monitor function info	
Monitor function name	Latin
Start autocalculation	Start

Start autocalculation
When pushed the automatic parameter variation is started and the monitor function output is writen into the textbox

Enable or Disable Functions

Functions in the analysis tree can easily be enabled or disabled by checking or un-checking the checkbox next to the function in the **History Explorer**.



Source/Sink

Bit streams stored in text files may be imported and differently interpreted according to the three file conversion formats available. A real-time bit stream will always be interpreted as binary ones and zeros.

Import Text Data

Input: Off-line bit stream

Function:

Imports a bit stream from a text file. Only ASCII ones (“1”, 0x31) and zeros (“0”, 0x30) are considered as valid characters, others values are ignored.

Example: “0110w700” is imported as “011000”.

Import Hex Data

Input: Off-line bit stream

Function:

Imports a bit stream from a text file. Only ASCII figures from “0” (0x30) to “9” (0x39) and letters from “A” (0x41) or ”a” (0x61) to “F” (0x47) or ”f” (0x67) are considered as valid characters, others values are ignored.

Example: ‟a1bg0c1kd0” is imported as ‟10100001101100001100000111010000”

Import Binary Data

Input: Off-line bit stream

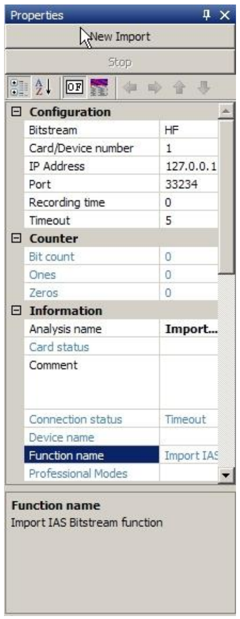
Function:

Imports a bit stream form a text file. All 8 bit ASCII characters”(0x00…0xFF) are considered as valid characters.

Example: ‟Ka□1Z<” is imported as ‟010010110110000110011100001100010101101 0”

Import IAS Bitstream

Input: Real-time bit stream



Configuration	
Bitstream	HF
Card/Device number	1
IP Address	127.0.0.1
Port	33234
Recording time	0
Timeout	5
Counter	
Bit count	0
Ones	0
Zeros	0
Information	
Analysis name	**Import...**
Card status	
Comment	
Connection status	Timeout
Device name	
Function name	Import IAS
Professional Modes	

Function name
Import IAS Bitstream function

Function:

Directly transfers a real-time bit stream from **WAVECOM Server**, the application that manages WAVECOM decoder cards.

Before starting a bit stream import into BitView, the desired mode must be started in the WAVECOM decoder.

In order to be able to connect to the server, the following settings are required:

Parameter	Value
Bitstream	This command will configure WAVECOM Server to run code appropriate to the selected decoder mode. The choices are: **HF** (default setting), **VHF- direct** or **VHF-indirect**
Card/Device number	For W51/W61 enter the decoder card number. For W-CODE enter the device number. These numbers are found in the decoder **Setup** menu
IP address	Enter the IP address or MS computer name of the PC that hosts WAVECOM Server. Default is 127.0.0.1, which must be used for a WAVECOM decoder and BitView installed on the same host
Port	Enter the port number of the XML Remote Control Interface (RCI) of WAVECOM Server. Default is port 33244 (see **WAVECOM Server Control | Networking Information**)
Recording duration	Enter the desired duration of the recording in seconds. ‘0’ means infinite
Timeout	Maximum time to establish a connection to WAVECOM Server. If a connection to the server could not be established within this period of time, the application cancels the connection procedure

To start importing press the **New Import** button on top of the **Properties** window. The document window will display **Recording....** and the same text will be displayed blinking in the display bottom line. The WAVECOM decoder screen will display **Sending bitstream to external application...**

To stop importing press the **Stop** button on top of the **Properties** window. After importing of an IAS bit stream has been stopped, the **Bit Sync Analysis/Import IAS Bitstream** function automatically opens.

Stopping the BitView import will not stop the decoder bit stream**,** nor will a decoder detect that the BitView Tool has been closed. Similarly BitView will not detect that the selected decoder mode has been closed.

If BitView cannot establish a connection with WAVECOM Server the **Recording...** status indication will disappear.

Export Text Data

Input: Current document window contents

Output: Text file

Function:

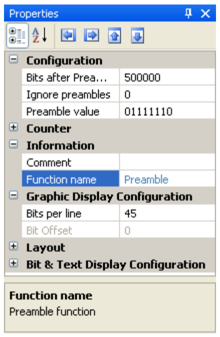
Write the contents of the current document to a text file. A **Save As** dialog will appear from which to select a filename and a folder for the exported file.

Synchronization

Preamble

In: Bit stream

Out: Bit stream



Properties

Configuration	
Bits after Prea...	500000
Ignore preambles	0
Preamble value	01111110
Counter	
Information	
Comment	
Function name	Preamble
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
Preamble function

Function:

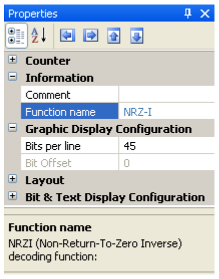
Searches for the **Preamble value** in the incoming bit stream and then writes the number of **Bits after Preamble** to the output. If the bit stream contains more than one preamble, the parameter **Ignore preambles** can be set for the function to skip a certain number of preambles.

Binary Modulation

NRZ-I

In: Bit stream

Out: Bit stream



Properties

Counter	
Information	
Comment	
Function name	NRZ-I
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
NRZI (Non-Return-To-Zero Inverse) decoding function:

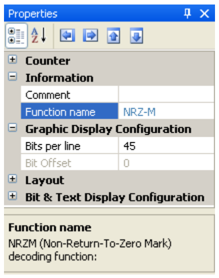
Function:

Changes the bit stream according to the "Non Return to Zero Inverse" (NRZ-I) decoding scheme, where no bit change represents a '1' and a bit change represents a '0'.

NRZ-M

In: Bit stream

Out: Bit stream



Properties

Counter

Information

Comment	
Function name	NRZ-M

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
NRZM (Non-Return-To-Zero Mark) decoding function:

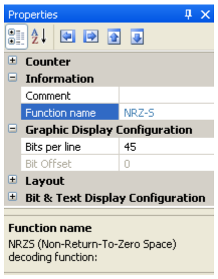
Function:

Changes the bit stream according to the "Non Return to Zero Mark" (NRZ-M) decoding scheme, where a bit change represents a '1' and no bit change represents a '0'.

NRZ-S

In: Bit stream

Out: Bit stream



Properties

Counter

Information

Comment	
Function name	NRZ-S

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
NRZS (Non-Return-To-Zero Space) decoding function:

Function:

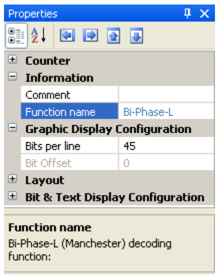
Changes the bit stream according to the "Non Return To Zero Space" (NRZ-S) decoding scheme, where no bit change represents a '1' and a bit change represents a '0'.

Note: This function is identical to NRZ-I.

Bi-Phase-L (Manchester)

In: Bit stream

Out: Bit stream



Properties

Counter

Information

Comment	
Function name	Bi-Phase-L

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
Bi-Phase-L (Manchester) decoding function:

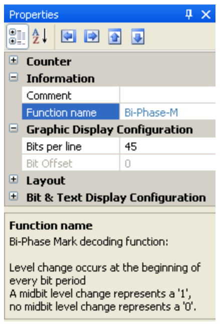
Function:

Analyze the bit changes of the bit stream. A change from '1' to '0' represents a '1' and a change from '0' to '1' represents a '0'. The bits are analyzed in pairs, i.e., the number of output bits is half the number of input bits.

Bi-Phase-M

In: Bit stream

Out: Bit stream


Properties

Counter

Information

Comment	
Function name	Bi-Phase-M

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
Bi-Phase Mark decoding function:

Level change occurs at the beginning of every bit period
A midbit level change represents a '1',
no midbit level change represents a '0'.

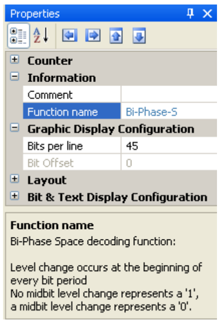
Function:

In Bi-Phase-M encoding, a logical ‘1’ is represented by a pair of bits of opposite values (‘10’ or ‘01’). A logical ‘0’ is represented by a pair of bits of the same values (‘00’ or ‘11’). The decoding procedure halves the number of output bits.

Bi-Phase-S

In: Bit stream

Out: Bit stream


Properties

Counter

Information

Comment	
Function name	Bi-Phase-S

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
Bi-Phase Space decoding function:

Level change occurs at the beginning of every bit period
No midbit level change represents a '1',
a midbit level change represents a '0'.

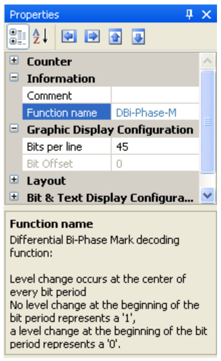
Function:

In Bi-Phase-S encoding, a logical ‘0’ is represented by a pair of two bits of opposite values (‘10’ or ‘01’). A logical ‘1’ is represented by a pair of bits of the same value (‘00’ or ‘11’). The decoding procedure halves the number of output bits.

DBi-Phase-M

In: Bit stream

Out: Bit stream


Properties

Counter

Information

Comment	
Function name	DBi-Phase-M

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configura...

Function name
Differential Bi-Phase Mark decoding function:

Level change occurs at the center of every bit period
No level change at the beginning of the bit period represents a '1',
a level change at the beginning of the bit period represents a '0'.

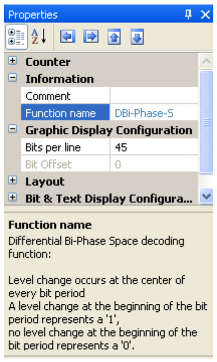
Function:

Two bits form a bit period. A bit change at the beginning of a bit period represents a ‘0’, while no bit change at the beginning of a bit period represents a ‘1’.

DBi-Phase-S

In: Bit stream

Out: Bit stream


Properties

Counter

Information

Comment	
Function name	DBi-Phase-S

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configura...

Function name
Differential Bi-Phase Space decoding function:

Level change occurs at the center of every bit period
A level change at the beginning of the bit period represents a '1',
no level change at the beginning of the bit period represents a '0'.

Function:

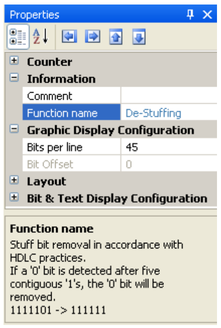
Two bits form a bit period. A bit change at the beginning of a bit period represents a ‘1’, while no bit change at the beginning of a bit period represents a ‘0’.

Bit Manipulation

De-Stuffing (HDLC)

In: Bit stream

Out: Bit stream


Properties

Counter

Information

Comment	
Function name	De-Stuffing

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
Stuff bit removal in accordance with HDLC practices.
If a '0' bit is detected after five contiguous '1's, the '0' bit will be removed.
1111101 -> 111111

Function:

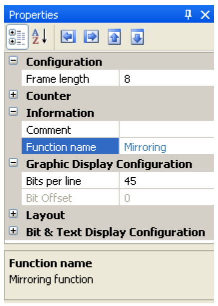
Removes stuff bits inserted in the input bit stream. If a zero bit is detected after five contiguous Ones, the Zero bit will be removed.

Example: “111101” is changed to “111111”.

Mirroring

In: Bit stream

Out: Bit stream


Properties

Configuration

Frame length	8

Counter

Information

Comment	
Function name	Mirroring

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
Mirroring function

Function:

The mirroring function modifies the incoming bit stream frame by frame. The **Frame length** is user defined. The function changes the bit order within each frame.

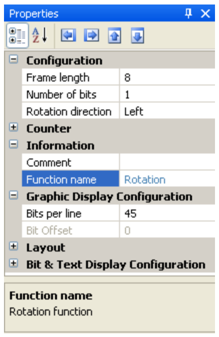
Example with a frame size of 5 bits:

“10111 01101” is changed to “11101 10110”.

Rotation

In: Bit stream

Out: Bit stream


Properties

Configuration	
Frame length	8
Number of bits	1
Rotation direction	Left
Counter	
Information	
Comment	
Function name	Rotation
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
Rotation function

Function:

The rotation function modifies the incoming bit stream frame by frame. The **Frame length** is user defined, as well as the **Rotation direction** and the **Number of bits** to be rotated.

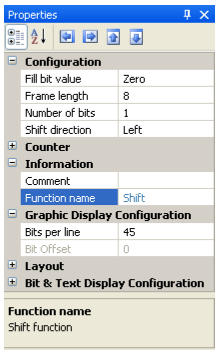
Example with a frame size of 5 bits, left rotation direction and one rotation step:

“10111 01101” is changed to “01111 11010”.

Shift

In: Bit stream

Out: Bit stream


Properties

Configuration	
Fill bit value	Zero
Frame length	8
Number of bits	1
Shift direction	Left
Counter	
Information	
Comment	
Function name	Shift
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
Shift function

Function:

The shift function modifies the incoming bit stream frame by frame. The **Frame length** is user defined, as well as the **Shift direction**, the **Number of bits** to be shifted and the **Fill bit value**.

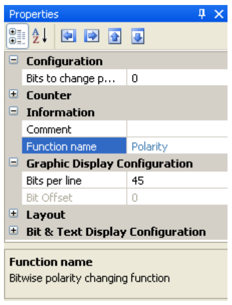
Example with a frame size of 5 bits, left shift direction, two bits shift and a fill value of ‘1’:

“10111 01101” is changed to “11111 10111”.

Polarity

In: Bit stream

Out: Bit stream


Properties

Configuration

Bits to change p...	0

Counter

Information

Comment	
Function name	Polarity

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
Bitwise polarity changing function

Function:

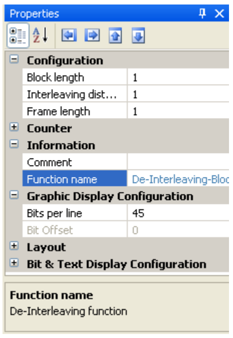
Invert the bit stream according to the **Bits to change polarity** pattern. At positions marked with a One, the bit is inverted, at positions with a Zero, the bit remains unchanged.

Example:	#1	#2	#3
Bits to change polarity	111000111	1	0
Input	111111111000000000	111111111000000000	111111111000000000
Output	000111000111000111	000000000111111111	111111111000000000

De-Interleave Bit Block

In: Bit stream

Out: Bit stream


Properties

Configuration

Block length	1
Interleaving dist...	1
Frame length	1

Counter

Information

Comment	
Function name	De-Interleaving-Bloc

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
De-Interleaving function

Function:

Change the bit order according to the settings of **Block length**, **Frame length** and **Interleaving distance**. The easiest way to understand the de-interleaving function is a closer look at the examples below (imagine that the bit stream is written horizontally into the buffer and read out vertically):

Example:	#1	#2
Interleaving distance	3	2
Block length	12	16
Frame length	1	2
Input	000111000111	0011001111100010

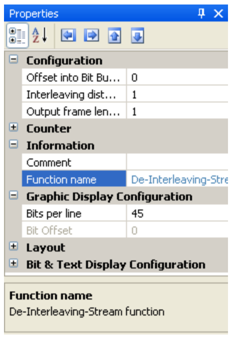
Matrix	000
	111
	000
	111
		00 11
		00 11
		11 10
		00 10

Output	010101010101	0000110011111010

De-Interleaving Stream

In: Bit stream

Out: Bit stream



Configuration	
Offset into Bit Bu…	0
Interleaving dist…	1
Output frame len…	1
Counter	
Information	
Comment	
Function name	De-Interleaving-Stre
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
De-Interleaving-Stream function

Function:

Change the bit order according to the settings of **Offset into Bit Buffer**, **Output frame length** and **Interleaving distance**.

The **Offset into Bit Buffer** tells the function where to start the de-interleaving function. Is **Offset into Bit Buffer** for example set to 3, then the first 3 bits will not be used for calculation of the output data. According to the **Output frame length** setting, the output data will be less than the input data. The easiest way to understand the de-interleaving stream function is a closer look at the example below (imagine that the bit stream is written horizontally into the buffer and read out vertically):

Example 1:

Offset into Bit Buffer = 0

Interleaving distance = 15

Output frame length = 4

```
                b0  distance b14
                |             |
          / -> |000000000000000|      |000000000000000|
output   /  -> |111111111111111| ==\  |111111111111111|
frame    \  -> |000000000000000| ==/  |000000000000000|
length    \ -> |111111111111111|      |111111111111111|
                |             |        |   |    |   |
               b45           b59       v   v    v   v

b0                                                         b59
 |                                                          |
 000000000000000111111111111111000000000000000111111111111111....... input stream
 |              |              |              |              ----->  0101  output frame 0
     |              |              |              |          ----->  0101  output frame 1
         |              |              |              |      ----->  0101  output frame 2
             |              |              |              |  ----->  0101  output frame 3

"000000000000000111111111111111000000000000000111111111111111" is changed to "0101010101010101"
```

Example 2:

Offset into Bit Buffer = 3

Interleaving distance = 8

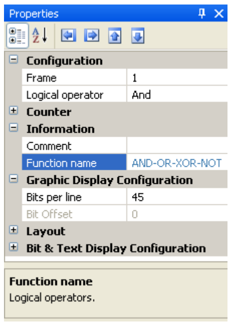
Output frame length = 4

```
011101100111000111100001111100000111111000000111111100000001111
   |       |       |       |                                   ----->  1101  output frame 0
       |       |       |       |                               ----->  0110  output frame 1
           |       |       |       |                           ----->  1011  output frame 2
               |       |       |       |                       ----->  1100  output frame 3
                   |       |       |       |                   ----->  0110  output frame 4
                       |       |       |       |               ----->  1001  output frame 5
                           |       |       |       |           ----->  1101  output frame 6
                               |       |       |       |       ----->  0010  output frame 7
                                   |       |       |       |   ----->  1011  output frame 8

"101100111000111100001111100000111111000000111111100000001111" is changed to "110101101011110001101001110100101011"
```

AND / OR / XOR / NOT

In: Bit stream

Out: Bit stream



Configuration	
Frame	1
Logical operator	And
Counter	
Information	
Comment	
Function name	AND-OR-XOR-NOT
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
Logical operators.

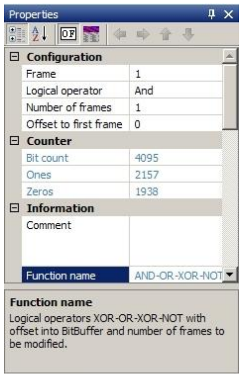
Function:

The output bit values depend on the selected logical operation (**Logical operator**) performed on the input bits. The first operand is the input bit stream, while the second operand (**Frame**) is constant and can be either ‘0’ or ‘1’.

AND / OR / XOR / NOT Range

In: Bit stream

Out: Bit stream



Properties

Configuration	
Frame	1
Logical operator	And
Number of frames	1
Offset to first frame	0
Counter	
Bit count	4095
Ones	2157
Zeros	1938
Information	
Comment	
Function name	AND-OR-XOR-NOT

Function name
Logical operators XOR-OR-XOR-NOT with offset into BitBuffer and number of frames to be modified.

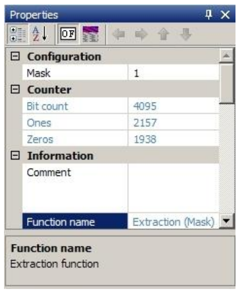
Function:

The output bit values depend on the selected logical operation (**Logical operator**) performed on the input bits from **Offset to first frame** and over **Number of frames**. The first operand is the input bit stream, while the second operand (**Frame**) is constant and can be ‘0’ or ‘1’.

Extraction (Mask)

In: Bit stream

Out: Bit stream



Properties

Configuration	
Mask	1
Counter	
Bit count	4095
Ones	2157
Zeros	1938
Information	
Comment	
Function name	Extraction (Mask)

Function name
Extraction function

Function:

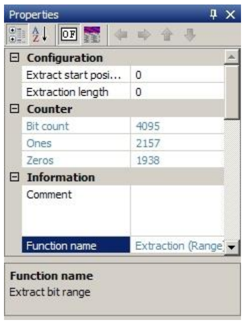
Extracts bits from the incoming bit stream using the user defined **Mask**. Only positions marked with a ‘1’ are extracted. The output bit stream is calculated frame by frame.

Example: “111110” with mask “110” changes to “1111”.

Extraction (Range)

In: Bit stream

Out: Bit stream



Properties

Configuration	
Extract start posi…	0
Extraction length	0
Counter	
Bit count	4095
Ones	2157
Zeros	1938
Information	
Comment	
Function name	Extraction (Range

Function name
Extract bit range

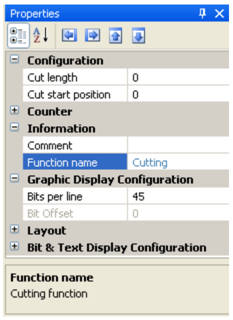
Function:

Extract **Extraction length** bits from the incoming bit stream starting at **Extract start position.**

Cutting

In: Bit stream

Out: Bit stream



Properties

Configuration	
Cut length	0
Cut start position	0
Counter	
Information	
Comment	
Function name	Cutting
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
Cutting function

Function:

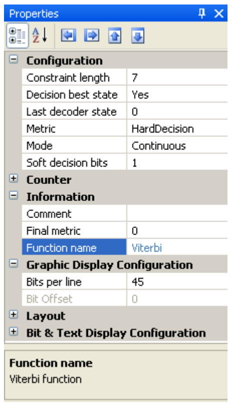
Cuts **Cut length** bits, beginning at **Cut start position**. Note that counting starts at zero, i.e., the first element in the bit stream is number 0.

Decoding/Equalizer

Viterbi-Decoding

In: Bit stream

Out: Bit stream



Properties

Configuration	
Constraint length	7
Decision best state	Yes
Last decoder state	0
Metric	HardDecision
Mode	Continuous
Soft decision bits	1
Counter	
Information	
Comment	
Final metric	0
Function name	Viterbi
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
Viterbi function

Function:

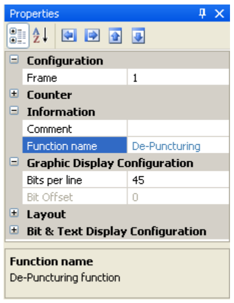
Decode the incoming bit stream using the Viterbi algorithm - a maximum-likelihood decoding procedure for convolutional codes.

Parameter	
Constraint length	Equals n+1, where n is the length of the shift register in the encoder
Decision best state	Use best state or not
Last decoder state	Initial state of the decoder
Metric	Select hard or soft decision
Mode	Select whether the input data should be treated as a continuous stream or a stream of bursts.
Soft decision bits	If soft decision is used, enter the number of soft decision bits

De-Puncturing

In: Bit stream

Out: Bit stream



Configuration
Frame 1
Counter
Information
Comment
Function name De-Puncturing
Graphic Display Configuration
Bits per line 45
Bit Offset 0
Layout
Bit & Text Display Configuration

Function name
De-Puncturing function

Function:

This function adds de-puncturing bits and probability bits to the input bit stream.

Parameter	Value
Frame	Bits are inserted according to the entered bit pattern. The length of the bit pattern corresponds to the frame length. At positions marked with a '0', a bit is inserted. Additionally, for every input bit, a probability bit is added. For received bits (marked with a '1' in the frame pattern), a '1' probability bit is added - for inserted bits, a '0' probability bit is added (equals a probability of 0.5)

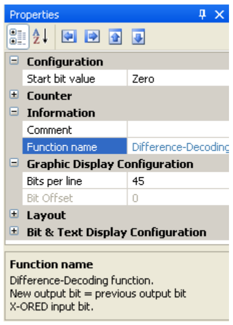
Example with a frame pattern of "110":

The frame pattern "110" means that after two input bits, a de-puncturing bit must be inserted so "1111" becomes "111100111100".

Difference-Decoding

In: Bit stream

Out: Bit stream



Configuration
Start bit value Zero
Counter
Information
Comment
Function name Difference-Decoding
Graphic Display Configuration
Bits per line 45
Bit Offset 0
Layout
Bit & Text Display Configuration

Function name
Difference-Decoding function.
New output bit = previous output bit X-ORED input bit.

Function:

This function performs difference decoding, which is a logical XOR operation on the previous output bit and the current input bit. The **Start bit value** is user defined.

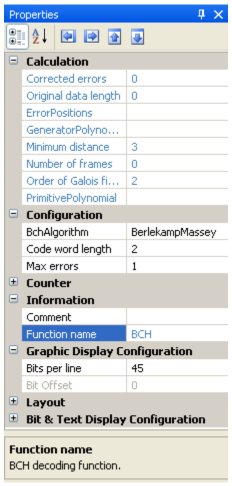
Example with start bit value of '1':

"01101110" is changed to "10110100".

BCH-Decoding

In: Bit stream

Out: Bit stream


Properties

Calculation

Corrected errors	0
Original data length	0
ErrorPositions	
GeneratorPolyno…	
Minimum distance	3
Number of frames	0
Order of Galois fi…	2
PrimitivePolynomial	

Configuration

BchAlgorithm	BerlekampMassey
Code word length	2
Max errors	1

Counter

Information

Comment	
Function name	BCH

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
BCH decoding function.

Function:

Decodes BCH encoded bit streams.

The function returns the parameters listed in the table below.

Parameter	Description
Corrected errors	Number of corrected errors
Error Positions	Positions of detected errors
Generator Polynomial	Encoding generator polynomial on the form g(x) = 111…
Minimum distance	This property determines the number of detectable errors and the number of correctable errors
Number of frames	Number of frames in the decoded bit stream
Order of Galois field	Property of the field of numbers to which a given BCH code belongs
Original data length	Length of original code word
Primitive polynomial	A polynomial describing a given BCH code

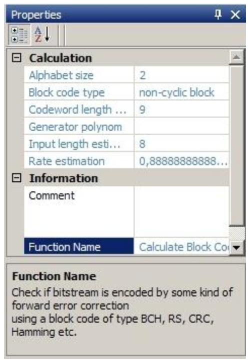
Parameter	Description	Value
BCH algorithm	Algorithm used to find BCH polynomial	Berlekamp-Massey, Euclid
Code word length	Length of code word including redundancy bits.	
Max errors	Error correction capability	

Example with code length 15 and error correction capability 3:
"011001010000111" is changed to "00111".

Block Code Analysis

In: Bit stream

Out: Bit stream


Properties

Calculation

Alphabet size	2
Block code type	non-cyclic block
Codeword length …	9
Generator polynom	
Input length esti…	8
Rate estimation	0,88888888888…

Information

Comment	
Function Name	Calculate Block Co…

Function Name
Check if bitstream is encoded by some kind of forward error correction
using a block code of type BCH, RS, CRC, Hamming etc.

Function:

Identify forward error correction block codes like BCH, RS, CRC or Hamming.

The function returns alphabet size (normally equal to 2 because input is a binary bit stream), block code type (BCH, RS, etc.), code word length estimation (n), generator polynomial, input length estimation (k) and code rate estimation (R = k/n).

The function will identify the following **Block Code Types,**

Unidentified code

BCH

CRC or perfect cyclic code

Binary repetition (reversals)

Binary Golay

Binary Hamming

CRC block code

Non-cyclic block code

The generator polynomial is returned as a string of ones and zeros, starting at the lowest order of 2^x, e.g., 1001110010101 means 1 + x^3 + x^4 + x^5 + x^8 + x^10 + x^12.

The results of the analysis are displayed in a new document window and in the **Calculation** category in the **Properties** window. The table below lists the calculated parameters.

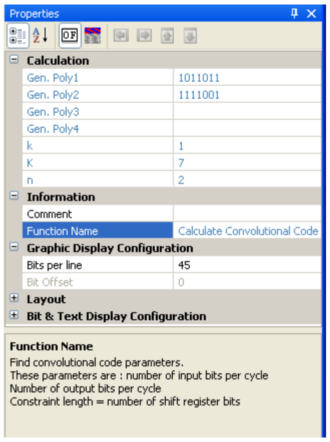
Parameter	Description
Alphabet size	Size of input alphabet
Block code type	Type of block code used
Codeword length estimation	Length of input code word (n) , i.e., number of data bits (k) plus number of checksum bits (n-k)
Generator polynomial	Generator polynomial (G(x)), i.e., the polynomial used to generate the checksum
Input length	Estimated number of data bits per code word
Rate estimation	Code rate (R = k/n), i.e., ratio between number of data bits (k) and total number of bits per frame (n+k)

Constraints: Frames must have equal length and if the bit stream is delimited by flags they must be removed with the **Extract** or **Cu**t functions.

Convolutional Code Analysis

In: Bit stream

Out: Bit stream



Calculation	
Gen. Poly1	1011011
Gen. Poly2	1111001
Gen. Poly3	
Gen. Poly4	
k	1
K	7
n	2
Information	
Comment	
Function Name	Calculate Convolutional Code
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function Name
Find convolutional code parameters.
These parameters are : number of input bits per cycle
Number of output bits per cycle
Constraint length = number of shift register bits

Function:

Find the parameters of convolutional encoded bit streams.

The function returns constraint length (K), number of input bits per shift cycle (k), number of output bits per shift cycle (n) and generator polynomials. The function will search for K = 2..14, and n = 2..4. The numbers of returned generator polynomials depend on the number of output bits per shift cycle (n).

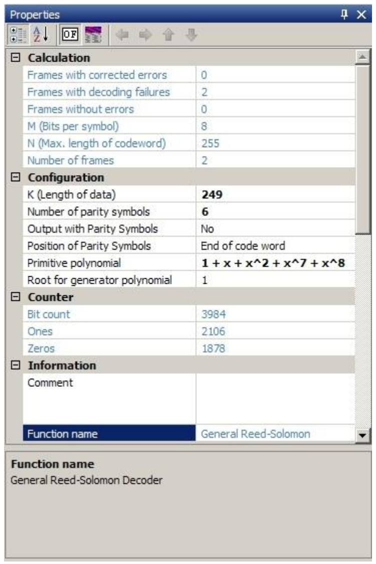
The results of the analysis are displayed in a new document window and in the **Calculation** category in the **Properties** window. The table below lists the calculated parameters.

Parameter	Description
Gen. Poly1	Generator polynomial 1 found
Gen. Poly2	Generator polynomial 2 found
Gen. Poly3	Generator polynomial 3 found
Gen. Poly4	Generator polynomial 4 found
Input bits (k)	Number of input bits to convolutional encoder
Constraint length (K)	Constraint length, i.e., the number of bit in encoder memory affecting the generation of n output bits
Output bits (n)	Number of output bits from convolutional encoder

General Reed Solomon Decoding

In: Bit stream

Out: Bit stream


Properties

Calculation

Frames with corrected errors	0
Frames with decoding failures	2
Frames without errors	0
M (Bits per symbol)	8
N (Max. length of codeword)	255
Number of frames	2

Configuration

K (Length of data)	**249**
Number of parity symbols	**6**
Output with Parity Symbols	No
Position of Parity Symbols	End of code word
Primitive polynomial	**1 + x + x^2 + x^7 + x^8**
Root for generator polynomial	1

Counter

Bit count	3984
Ones	2106
Zeros	1878

Information

Comment	
Function name	General Reed-Solomon

Function name
General Reed-Solomon Decoder

Function:

Decodes bit streams which are encoded with a Reed-Solomon code.

The decoder must be configured by the parameters listed below.

Parameter	Description	Value
Length of data (k)	Number of data symbols in a code word, where the number of data bits (k) is less than the total number of bits (k), i.e., k < n	
Number of parity symbols	Number of parity symbols in a code word (n-k)	
Output with Parity Symbols	Specifies if parity symbols should be included in the output	Yes, No
Position of Parity Symbols	Specifies if parity symbols are leading or trailing the code word	End of code word, Beginning of code word
Primitive poly-nomial	Encoding polynomial	Select from drop-down list
Root for generator polynomial	Root for generator polynomial. See parameter help desk for further explanation	

The results of the analysis are displayed in the **Calculation** category in the **Properties** window. The table below lists the calculated parameters.

Parameter	Description
Frames with corrected errors	Number of frames with correctable errors
Frames with decoding failures	Number of frames with decoding errors

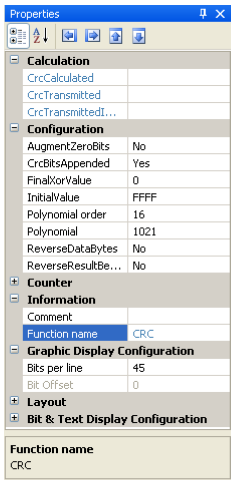
Frames without errors	Number of frames without errors
m (bits per symbol)	Number of bits per symbol (m)
n (max. length of code word)	Maximum number of symbols per code word. N is derived from the selected primitive polynomial
Number of decoded frames	Number of decoded frames

CRC & Polynomial

CRC (1..32)

In: Bit stream

Out: Bit stream


Properties

Calculation

CrcCalculated	
CrcTransmitted	
CrcTransmittedI…	

Configuration

AugmentZeroBits	No
CrcBitsAppended	Yes
FinalXorValue	0
InitialValue	FFFF
Polynomial order	16
Polynomial	1021
ReverseDataBytes	No
ReverseResultBe…	No

Counter

Information

Comment	
Function name	CRC

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name
CRC

Function:

Calculate the cyclic redundancy checksum (CRC) value of the input bit stream according to the settings described below.

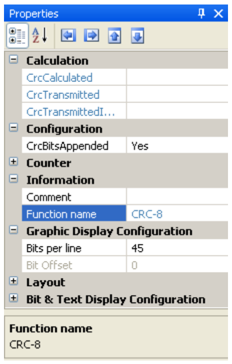
Parameter	Value
Augment Zero Bits	Calculate CRC with or without augmented (added) zero bits
Crc Bits Appended	Are CRC bits appended to the bit stream (Yes or No)
Final Xor Value	Final XOR value in hex
Initial Value	Initial value in hex
Order	Polynomial order (1..32)
Polynomial	Polynomial in hex
Reverse Data Bytes	Reflect data byte before processing (Yes or No)
Reverse Result Before Xor	Reflect final result before XOR (Yes or No)

The calculated CRC value is displayed in the **Crc Calculated** field. If **Crc Bits Appended** is set to ‘Yes’, then the **Crc Transmitted** field contains the transmitted CRC value in hex format. If there are no appended CRC bits, then the **Crc Transmitted** field has no meaning. The transmitted CRC value is also displayed in inverted format (**Crc Transmitted Inverse**).

CRC-8

In: Bit stream

Out: Bit stream



Calculation	
CrcCalculated	
CrcTransmitted	
CrcTransmittedI…	
Configuration	
CrcBitsAppended	Yes
Counter	
Information	
Comment	
Function name	CRC-8
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
CRC-8

Function:

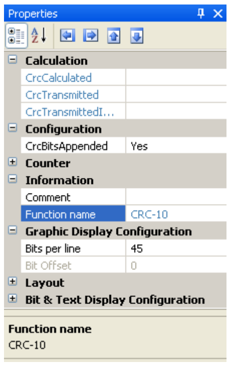
Calculate the standard CRC-8 values of the incoming bit stream.

The calculated CRC value is displayed in the **Crc Calculated** field. If **Crc Bits Appended** is set to **Yes**, then the **Crc Transmitted** field contains the transmitted CRC value in hex format. If there are no appended CRC bits, then the **Crc Transmitted** field has no meaning. The transmitted CRC value is also displayed in inverted format (**Crc Transmitted Inverse**).

CRC-10

In: Bit stream

Out: Bit stream



Calculation	
CrcCalculated	
CrcTransmitted	
CrcTransmittedI…	
Configuration	
CrcBitsAppended	Yes
Counter	
Information	
Comment	
Function name	CRC-10
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
CRC-10

Function:

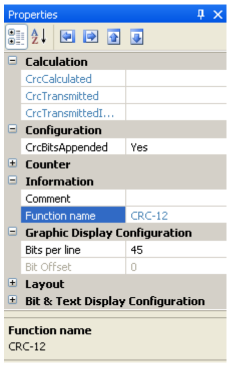
Calculate the standard CRC-10 values of the incoming bit stream.

The calculated CRC value is displayed in the **Crc Calculated** field. If **Crc Bits Appended** is set to **Yes**, then the **Crc Transmitted** field contains the transmitted CRC value in hex format. If there are no appended CRC bits, then the **Crc Transmitted** field has no meaning. The transmitted CRC value is also displayed in inverted format (**Crc Transmitted Inverse**).

CRC-12

In: Bit stream

Out: Bit stream



Calculation	
CrcCalculated	
CrcTransmitted	
CrcTransmittedI…	
Configuration	
CrcBitsAppended	Yes
Counter	
Information	
Comment	
Function name	CRC-12
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
CRC-12

Function:

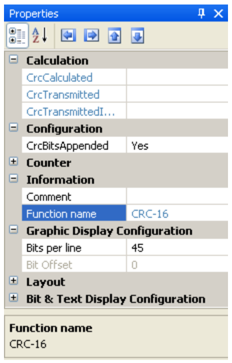
Calculate the standard CRC-12 values of the incoming bit stream.

The calculated CRC value is displayed in the **Crc Calculated** field. If **Crc Bits Appended** is set to **Yes**, then the **Crc Transmitted** field contains the transmitted CRC value in hex format. If there are no appended CRC bits, then the **Crc Transmitted** field has no meaning. The transmitted CRC value is also displayed in inverted format (**Crc Transmitted Inverse**).

CRC-16

In: Bit stream

Out: Bit stream



Calculation	
CrcCalculated	
CrcTransmitted	
CrcTransmittedI…	
Configuration	
CrcBitsAppended	Yes
Counter	
Information	
Comment	
Function name	CRC-16
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
CRC-16

Function:

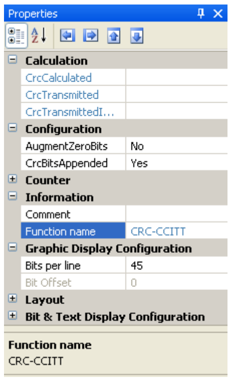
Calculate the standard CRC-16 values of the incoming bit stream.

The calculated CRC value is displayed in the **Crc Calculated** field. If **Crc Bits Appended** is set to **Yes**, then the **Crc Transmitted** field contains the transmitted CRC value in hex format. If there are no appended CRC bits, then the **Crc Transmitted** field has no meaning. The transmitted CRC value is also displayed in inverted format (**Crc Transmitted Inverse**).

CRC-CCITT

In: Bit stream

Out: Bit stream



Calculation	
CrcCalculated	
CrcTransmitted	
CrcTransmittedI…	
Configuration	
AugmentZeroBits	No
CrcBitsAppended	Yes
Counter	
Information	
Comment	
Function name	CRC-CCITT
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
CRC-CCITT

Function:

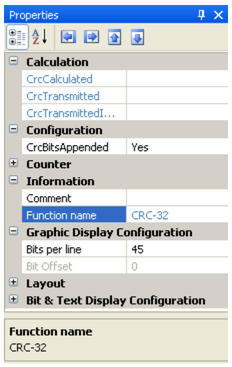
Calculate the standard CRC-CCITT values of the incoming bit stream.

The calculated CRC value is displayed in the **Crc Calculated** field. If **Crc Bits Appended** is set to **Yes**, then the **Crc Transmitted** field contains the transmitted CRC value in hex format. If there are no appended CRC bits, then the **Crc Transmitted** field has no meaning. The transmitted CRC value is also displayed in inverted format (**Crc Transmitted Inverse**).

CRC-32

In: Bit stream

Out: Bit stream



Properties

Calculation	
CrcCalculated	
CrcTransmitted	
CrcTransmittedI…	
Configuration	
CrcBitsAppended	Yes
Counter	
Information	
Comment	
Function name	CRC-32
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Function name
CRC-32

Function:

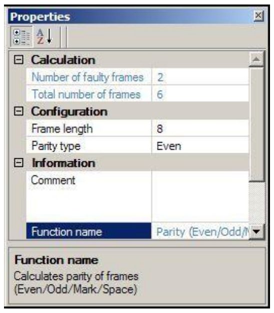
Calculate the standard CRC-32 values of the incoming bit stream.

The calculated CRC value is displayed in the **Crc Calculated** field. If **Crc Bits Appended** is set to **Yes**, then the **Crc Transmitted** field contains the transmitted CRC value in hex format. If there are no appended CRC bits, then the **Crc Transmitted** field has no meaning. The transmitted CRC value is also displayed in inverted format (**Crc Transmitted Inverse**).

Parity (Even/Odd/Mark/Space)

In: Bit stream

Out: Bit stream



Properties

Calculation	
Number of faulty frames	2
Total number of frames	6
Configuration	
Frame length	8
Parity type	Even
Information	
Comment	
Function name	Parity (Even/Odd/M

Function name
Calculates parity of frames (Even/Odd/Mark/Space)

Function:

Calculate the parity of frames.

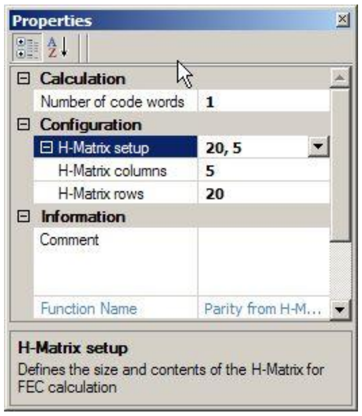
Enter **Frame length** and **Parity type** from the drop-down list.

The function returns **Number of faulty frames** and **Total number of frames** in the **Calculation** category and in a document window the selected matrix and a list of the processed frames in bit notation and parity marked as **Parity o.k.** or **Parity error**.

Parity from H-Matrix

In: Bit stream

Out: Bit stream



Properties

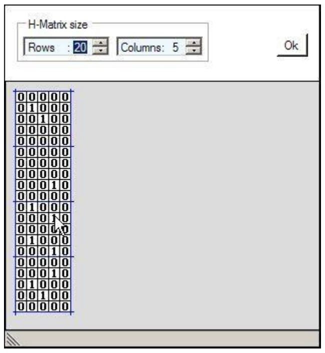
Calculation	
Number of code words	1
Configuration	
H-Matrix setup	20, 5
H-Matrix columns	5
H-Matrix rows	20
Information	
Comment	
Function Name	Parity from H-M…

H-Matrix setup
Defines the size and contents of the H-Matrix for FEC calculation

Function:

Calculate the parity of frames using H-matrix multiplication.

By clicking the **H-matrix setup** field a drop-down list appears. Enter the matrix size in the **Rows and Columns** fields, then proceed to fill in logical '1' in the appropriate positions of the matrix by left-clicking the position. A context label indicates the cursor position.



Rows : 20
Columns: 5

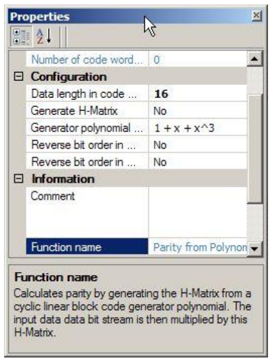
```
0 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 1 0
0 0 0 0 0
0 1 0 0 0
0 0 0 1 0
0 0 0 0 0
0 1 0 0 0
0 0 0 1 0
0 0 0 0 0
0 0 0 1 0
0 1 0 0 0
0 0 1 0 0
0 0 0 0 0
```

The function returns **Number of code words** processed in the **Calculation** category and in a document window a list of the analyzed frames and the calculated parity bits in bit notation and parity marked as **Parity o.k.** or **Parity error**.

Parity from polynomial

In: Bit stream

Out: Bit stream



Number of code word...	0
Configuration	
Data length in code ...	**16**
Generate H-Matrix	No
Generator polynomial ...	1 + x + x^3
Reverse bit order in ...	No
Reverse bit order in ...	No
Information	
Comment	
Function name	Parity from Polynon

Function name
Calculates parity by generating the H-Matrix from a cyclic linear block code generator polynomial. The input data data bit stream is then multiplied by this H-Matrix.

Function:

Calculate the parity of frames by generating an H-matrix from the corresponding cyclic linear block code generator polynomial. The bit stream is then multiplied by the generated H-matrix.

The parameters listed below configure the function:

Parameter	Description	Value
Data length in code word	Number of data bits in code word	
Generate H-matrix	Generate H-matrix	Yes, No
Generator polynomial (standard format)	Use the drop-list to open a window for entering the generator polynomial	Example: 1+x+x^2…
Reverse bit order in data field	Reverse bit order in the data part of code word	Yes, No
Reverse bit order in gen. poly.	Reverse bit order in generator polynomial	Yes, No

The function returns these parameters:

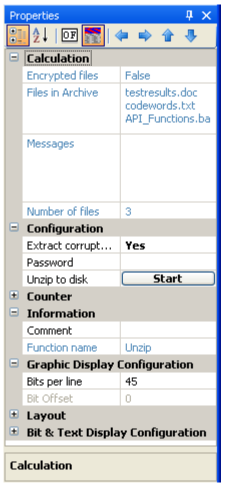
Parameter	Description
Code word length	Total length of code word (data and parity bits)
Number of code words	Number of complete code words
Number of code words with error	Number of code words with bad parity

Unpacking/Decompress

Unzip

In: bit stream

Out: bit stream



Properties

Calculation	
Encrypted files	False
Files in Archive	testresults.doc codewords.txt API_Functions.ba
Messages	
Number of files	3
Configuration	
Extract corrupt…	**Yes**
Password	
Unzip to disk	Start
Counter	
Information	
Comment	
Function name	Unzip
Graphic Display Configuration	
Bits per line	45
Bit Offset	0
Layout	
Bit & Text Display Configuration	

Calculation

Function:

The unzip function decompresses a ZIP archive from the input bit stream using the standard DEFLATE algorithm.

If the function is able to detect compressed files in the bit stream, the names of these files is shown in the **Files in Archive** field.

The output of this function can be processed further by using other functions or can be saved into files. The **Unzip to disk** function is started by pressing the **Start** button. If a password is required enter it in **Password.** A file dialog will appear to select the folder for the decompressed files.

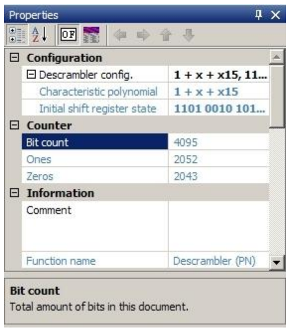
If a file is corrupted or data is missing and **Extract corrupted files** is set to **No**, an error message will appear when the decompression of the corrupted file has completed. If **Extract corrupted files** is set to **Yes**, damaged archives will be processed without an error message. It should be noted however, that if essential parts of the file header are in error the function may be unable to decompress the file.

Descrambling

Descrambler (PN)

In: Bit stream

Out: Bit stream



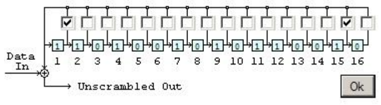
Properties

Configuration	
Descrambler config.	**1 + x + x15, 11...**
Characteristic polynomial	**1 + x + x15**
Initial shift register state	**1101 0010 101...**
Counter	
Bit count	4095
Ones	2052
Zeros	2043
Information	
Comment	
Function name	Descrambler (PN)

Bit count
Total amount of bits in this document.

Function:

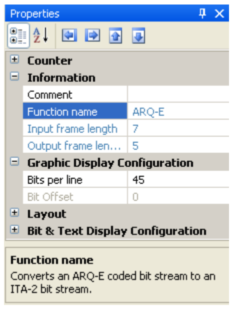
Descramble a bit stream scrambled with a scrambling polynomial (pseudo random sequence). The polynomial is entered using the **Descrambler config** field. Clicking the drop-down list will open a shift register schematic with 16 stages each representing in increasing order the elements of the scrambling polynomial.



A click on the tick box for each stage enables or disables the individual stage. Clicking a stage icon will toggle its initial state between 1 and 0. When configuring the descrambler is complete click **Ok** to save the configuration, which is displayed in the **Characteristic polynomial** and **Initial shift register state** fields.

Channel Decoding (Protocol)

The channel decoding functions convert a number of channel coded input bit streams into ITA-2 or ITA-5 bit streams. The **Properties** windows for these functions are identical to the figure below, and thus they will not be reproduced for every single function.


Properties

Counter

Information

Comment	
Function name	ARQ-E
Input frame length	7
Output frame len…	5

Graphic Display Configuration

Bits per line	45
Bit Offset	0

Layout

Bit & Text Display Configuration

Function name

Converts an ARQ-E coded bit stream to an ITA-2 bit stream.

ARQ-E

In: Bit stream

Out: Bit stream

Function:

Convert an ARQ-E coded bit stream into an ITA-2 bit stream.

SITOR

In: Bit stream

Out: Bit stream

Function:

Convert a SITOR coded bit stream into an ITA-2 bit stream.

FEC-A

In: Bit stream

Out: Bit stream

Function:

Convert an FEC-A coded bit stream into an ITA-2 bit stream.

BAUER

In: Bit stream

Out: Bit stream

Function:

Convert a BAUER coded bit stream into an ITA-2 bit stream.

HNG-FEC

In: Bit stream

Out: Bit stream

Function:

Convert a HNG-FEC coded bit stream into an ITA-2 bit stream.

RUM-FEC

In: Bit stream

Out: Bit stream

Function:

Convert a RUM-FEC coded bit stream into an ITA-2 bit stream.

ITA-3 (M.342)

In: Bit stream

Out: Bit stream

Function:

Convert an ITA-3 coded bit stream into an ITA-2 bit stream.

ITA-5

In: Bit stream

Out: Bit stream

Function:

Remove the parity bits of an ITA-5 coded bit stream.

PSK-31 (Varicode)

In: Bit stream

Out: Bit stream

Function:

Convert a PSK-31 (Varicode) code into an ITA-2 bit stream.

Source Decoding (Alphabet)

Latin

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Latin). If **Transparent** display mode is set to **No,** then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

The source code format can be set to ITA-2 or ITA-1.

Third-shift Greek

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Third-shift Greek). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

Cyrillic

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Cyrillic). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

Tass-Cyrillic

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (TASS Cyrillic). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

Third-shift Cyrillic

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Third-shift Cyrillic). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

Hebrew

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Hebrew). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

Arabic Baghdad-70

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Arabic Baghdad-70). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

Arabic Baghdad-80 (ATU-80)

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Arabic Baghdad-80). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

Bulgarian

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Bulgarian). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

Swedish

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Swedish). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If the **Transparent** display mode is set to **Yes**, then the special characters are displayed, using a corresponding, descriptive Symbol.

Danish-Norwegian

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (Danish/Norwegian). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

German

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (German). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

French

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text (French). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

US

In: Bit stream

Out: Character stream

Function:

Converts a bit stream to Unicode text (US). If **Transparent** display mode is set to **No**, then special characters like carriage return or line feed are treated as control characters and applied to the output window. If **Transparent** display mode is set to **Yes**, then the special characters are displayed using a corresponding, descriptive symbol.

ASCII

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to ASCII text. The user can select between 7 bit ASCII and 8 bit ASCII.

UNICODE

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to Unicode text. The user can select between Little- and Big-Endian.

UTF-7

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to UTF-7 text.

UTF-8

In: Bit stream

Out: Character stream

Function:

Convert a bit stream to UTF-8 text.

Pager Numeric

In: Bit stream

Out: Character stream

Function:

Convert a pager (POCSAG) bit stream to numeric characters.

Analysis Tools

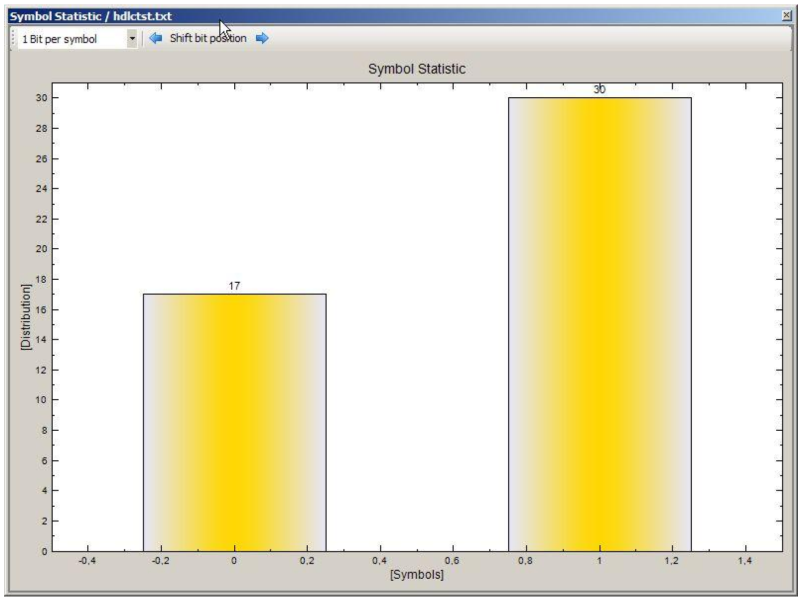
Symbol Statistics

In: Bit stream

Out: Chart

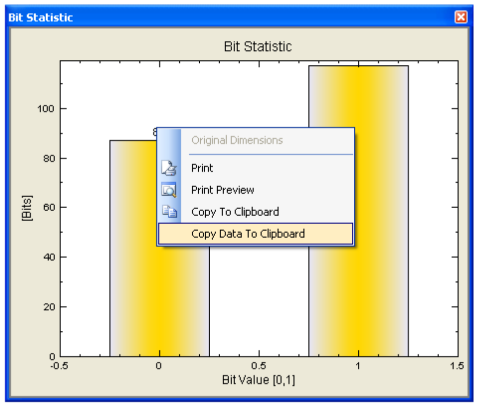
Function:

Generate a histogram displaying the statistical distribution of symbols. The number of bits per symbol can be adjusted in a field in the upper left corner of the chart. Additionally bits may be shifted right or left using the arrow buttons.



This chart is only calculated once, i.e., the content does not change, even if the analysis set is recalculated. To update histogram values, close the window and then reopen it.

A right click on the histogram makes additional functionality available.



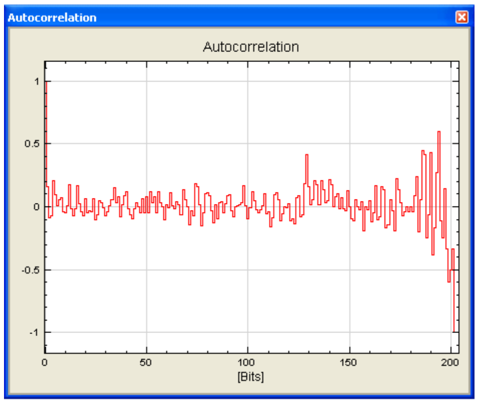
Autocorrelation

In: Bit stream

Out: Chart

Function:

Generate a graphical display of an autocorrelation operation on the input bit stream.

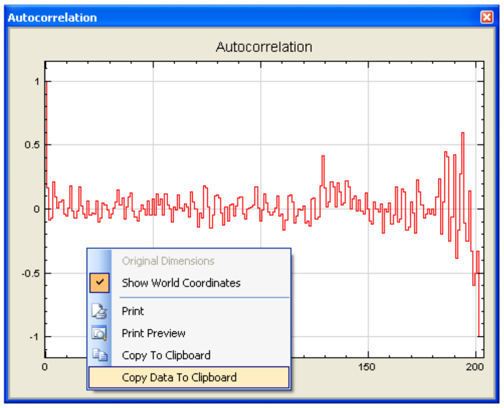


This chart is only calculated once, i.e., the content does not change, even if the analysis set is recalculated. To update graph values, close the window and then reopen it.

Built-in zoom functions that are available by using mouse clicks.

A drag-and-drop operation will select an area for zooming.

A right click on the display makes additional functionality available.



If the view has been changed by drag and drop, the original dimensions can be restored by clicking **Original Dimensions**.

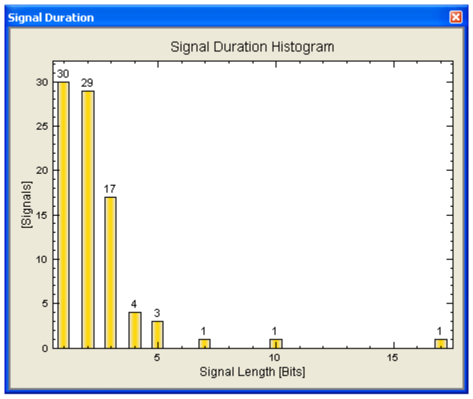
Signal Duration

In: Bit stream

Out: Histogram

Function:

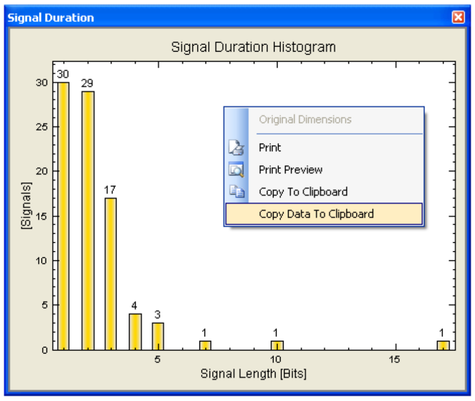
Generates a statistical histogram of signal duration.



This chart is only calculated once, i.e., the content does not change, even if the analysis set is recalculated. To update histogram values, close the window and then reopen it.

A drag-and-drop operation will select an area for zooming.

A right click on the display makes additional functionality available.



If the view has been changed by drag and drop, the original dimensions can be restored by clicking **Original Dimensions**.

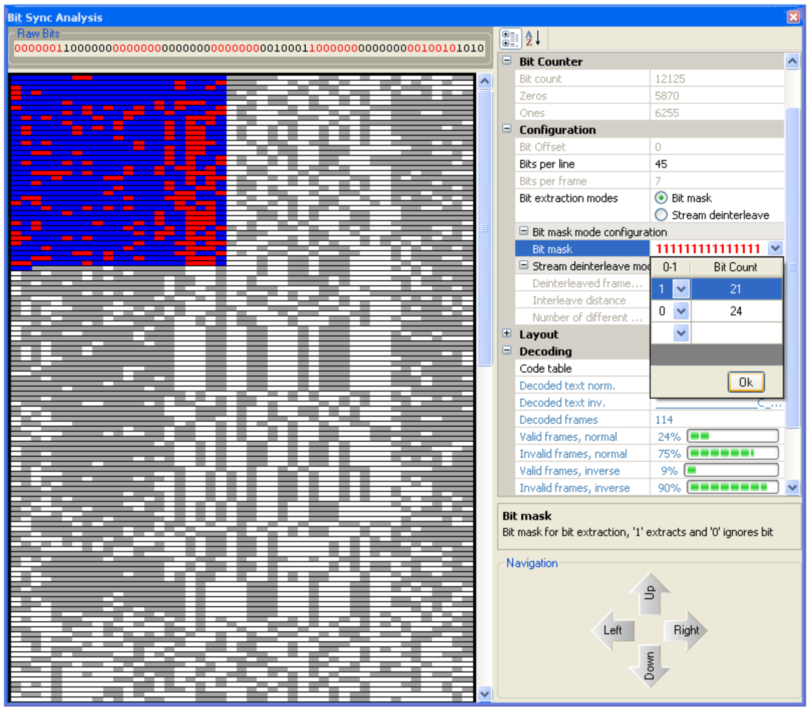
Bit Sync Analysis

In: Bit stream

Out: Analysis window

Function:

Opens a bit synchronization analysis window.



```
0000001100000000000000000000000000000000010001100000000000000000100101010
```

Bit Counter	
Bit count	12125
Zeros	5870
Ones	6255
Configuration	
Bit Offset	0
Bits per line	45
Bits per frame	7
Bit extraction modes	● Bit mask ○ Stream deinterleave
Bit mask mode configuration	
Bit mask	111111111111111
Stream deinterleave mod	
Deinterleaved frame…	
Interleave distance	
Number of different …	
Layout	
Decoding	
Code table	
Decoded text norm.	
Decoded text inv.	C_…
Decoded frames	114
Valid frames, normal	24%
Invalid frames, normal	75%
Valid frames, inverse	9%
Invalid frames, inverse	90%

0-1	Bit Count
1	21
0	24

Bit mask
Bit mask for bit extraction, '1' extracts and '0' ignores bit

Navigation

Bit Sync Analysis is designed to find the starting position of a frame. For this reason, all bits are displayed in a graphical view with an adjustable number of bits per line; this makes it easier to find periodic sequences.

The settings for **Bit Sync Analysis** are defined in the **Properties** window:

Bits per line define the number of bits per line.

Bit extraction mode offers a choice between **Bit mask** and **Stream de-interleave** modes. Depending on the mode selected, either the bit mask or the de-interleaving can be configured. For details of these functions, please refer to the descriptions of the **Extraction** and **De-Interleaving Bit Stream** functions in the **Bit Manipulation** section of this manual.

The **Layout** category controls the appearance of the graphical bit display.

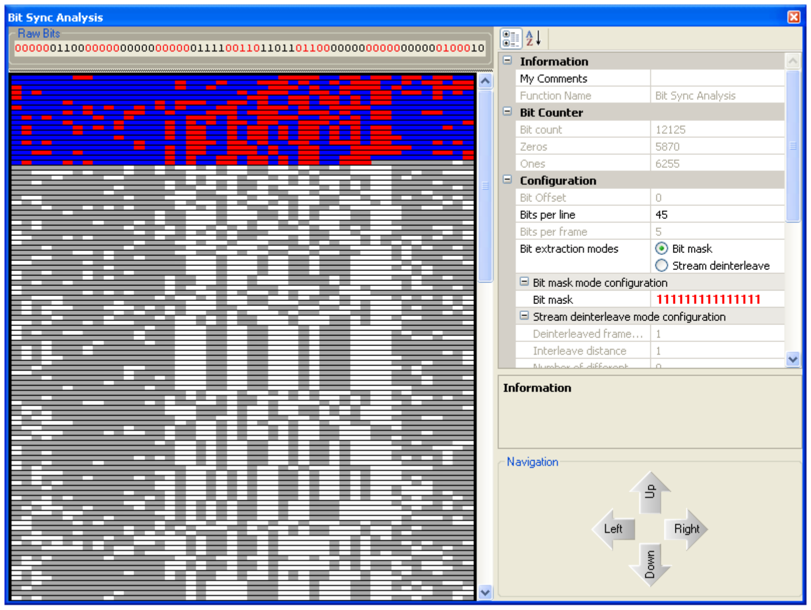
In the **Decoding** category, the decoding alphabet is selected. Supported alphabets are ITA2, ITA3, CCIR476_5, ASCII (7 Bit) and ASCII (8 Bit). Decoding is only possible if **Bit mask** is activated.

The CCIR476_5 and ITA3 alphabets are redundant alphabets allowing a calculation of the number of valid frames, which is displayed in the **Decoding** category. This is an additional help to find the start of a frame inside a bit stream. For the other alphabets, a validation is not possible.

The **Navigation** category defines the behavior of the four arrow buttons in the bottom area of the dialog window.

Example:

Open the SITOR example analysis, located in the **BitViewTool\Examples** folder and select the **Import IAS Bit stream** function in the **History Explorer**. Open the **Analysis Tools** in the **Toolbox** and select **Bit Sync Analysis**. The dialog below will appear:



Bit Sync Analysis

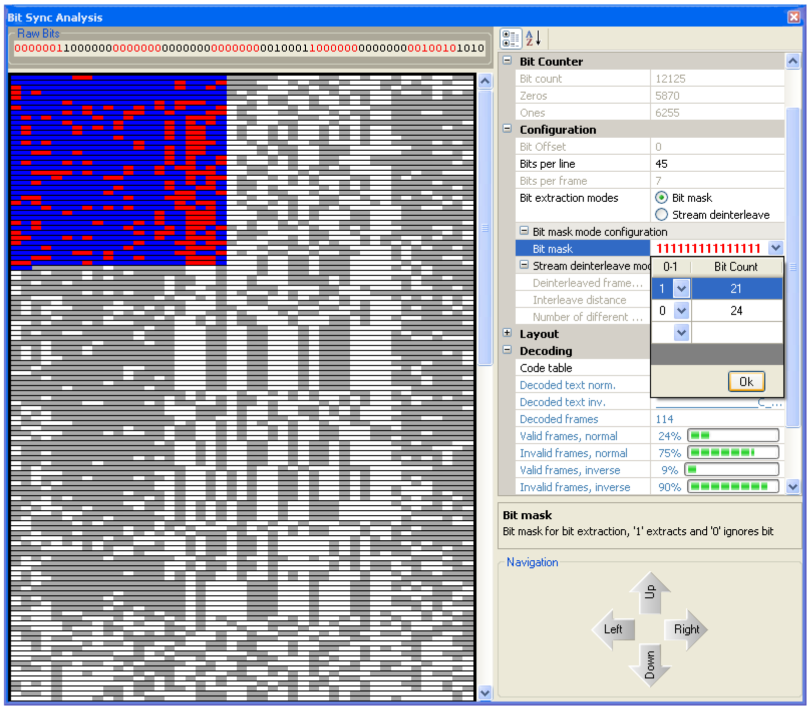
Raw Bits

```
000000110000000000000000000111100110110110110000000000000000000100010
```

Information	
My Comments	
Function Name	Bit Sync Analysis
Bit Counter	
Bit count	12125
Zeros	5870
Ones	6255
Configuration	
Bit Offset	0
Bits per line	45
Bits per frame	5
Bit extraction modes	[✓ Bit mask] Stream deinterleave
Bit mask mode configuration	
Bit mask	111111111111111
Stream deinterleave mode configuration	
Deinterleaved frame…	1
Interleave distance	1

Information

SITOR-A has a block length of 45, so adjust **Bits per line** to 45 bits. As the alphabet is known, select **CCIR476_5**. The next step is to configure the **Bit mask** with 21 ones and 24 zeros. After configuration of the bit mask, the dialog looks like this:



Bit Sync Analysis

Raw Bits

```
000000110000000000000000000000000000000000100011000000000000000000100101010
```

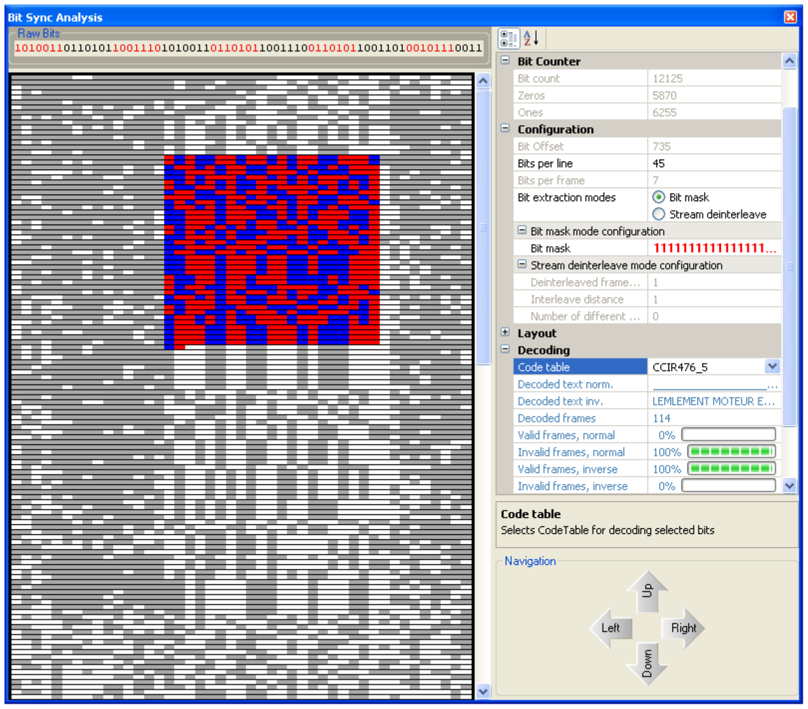
Bit Counter	
Bit count	12125
Zeros	5870
Ones	6255
Configuration	
Bit Offset	0
Bits per line	45
Bits per frame	7
Bit extraction modes	[✓ Bit mask] Stream deinterleave
Bit mask mode configuration	
Bit mask	111111111111111
Stream deinterleave mod…	
Deinterleaved frame…	
Interleave distance	
Number of different …	
Layout	
Decoding	
Code table	
Decoded text norm.	
Decoded text inv.	C_…
Decoded frames	114
Valid frames, normal	24%
Invalid frames, normal	75%
Valid frames, inverse	9%
Invalid frames, inverse	90%

0-1	Bit Count
1	21
0	24

Ok

Bit mask
Bit mask for bit extraction, '1' extracts and '0' ignores bit

As the SITOR alphabet allows validation, the remaining task is to move the selection across the window with the navigation arrow buttons and check the percentage of valid frames. The figure below shows a valid bit block:



```
1010011011010110011101010011011010110011100110101100110100101110011
```

Bit Counter	
Bit count	12125
Zeros	5870
Ones	6255
Configuration	
Bit Offset	735
Bits per line	45
Bits per frame	7
Bit extraction modes	[✓ Bit mask] Stream deinterleave
Bit mask mode configuration	
Bit mask	1111111111111111...
Stream deinterleave mode configuration	
Deinterleaved frame...	1
Interleave distance	1
Number of different ...	0
Layout	
Decoding	
Code table	CCIR476_5
Decoded text norm.	...
Decoded text inv.	LEMLEMENT MOTEUR E...
Decoded frames	114
Valid frames, normal	0%
Invalid frames, normal	100%
Valid frames, inverse	100%
Invalid frames, inverse	0%

Code table
Selects CodeTable for decoding selected bits

The decoded text is displayed in the **Decoding** category.

Analysis Set Library

Familiarize yourself with Bit Analysis

To familiarize you with bit analysis using the rich function and tool set of BitView, it is strongly recommended to study the working of ready-made Analysis Sets. The library of analysis sets is found here,

Windows XP and earlier:

- Documents and Settings\All Users\Documents \WAVECOM\BitViewTool\Examples
- Documents and Settings\All Users\Shared Documents \WAVECOM\BitViewTool\Examples

Windows Vista:

- Users\Public\Public Documents\WAVECOM\BitViewTool\Examples

Custom Library

Want to Roll Your Own functions?

The ability to expand BitViewTool with custom developed functions is one of the most powerful features of this application. However, before you start rolling your own functions you must have a good grasp of C#, object oriented programming and the use of Microsoft Visual Studio and .NET. In addition a solid knowledge of MatLab and the mathematical aspects of communication systems is a prerequisite to benefit from these powerful tools. The books listed below may help the programmer to get acquainted with the development tools,

- Bernhard Sklar, “Digital Communications: Fundamentals and Applications”, 2nd Ed., Prentice-Hall, 2001
- John Sharp, “Microsoft Visual C# 2005 – Step by Step”, Microsoft Press, 2006
- Ruda Pratap, “Getting Started with MATLAB 7”, Oxford

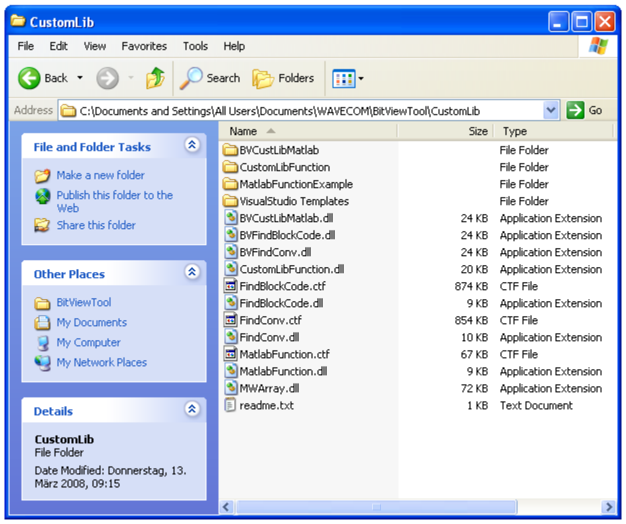
The Custom Library Interface supports the integration of third party functions into BitViewTool. Functions may be implemented using any .NET language, i.e., C#.NET, VB.NET, J#.NET and C++.NET, using the .NET Framework 2.0. Custom functions are compiled into individual 32-bit .NET DLLs and executed on operating systems supported by BitViewTool, including Windows 2000, XP, 2003 Server and Vista.

As from release 2.1 it is also possible to write mathematical functions in MatLab and integrate them with BitViewTool.

Examples using custom functions are distributed with BitViewTool. The custom library DLL as well as the source code can be found in the **CustomLib** folder of the BitViewTool. The projects (solutions) provided were created using MS Visual Studio .NET 2005 with .NET Framework 2.0.

The **CustomLib** folder is created during installation.

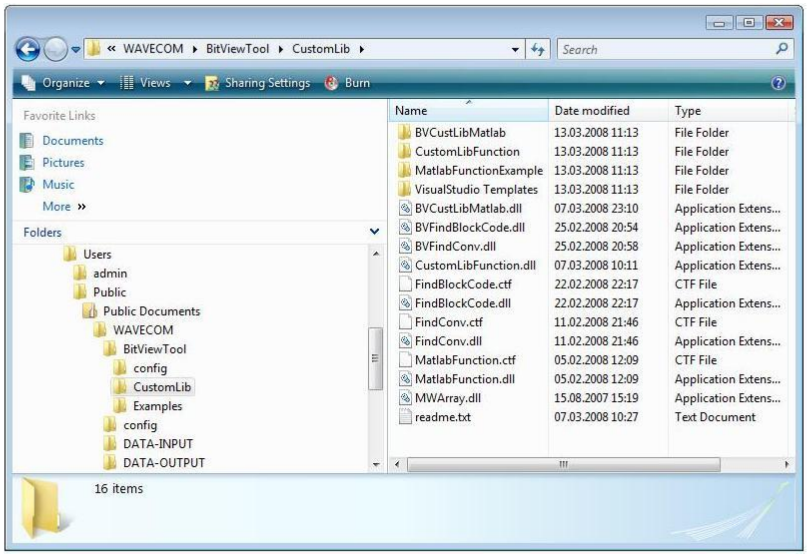
On Windows XP and older:



CustomLib
File Folder
Date Modified: Donnerstag, 13. März 2008, 09:15

Name	Size	Type
BVCustLibMatlab		File Folder
CustomLibFunction		File Folder
MatlabFunctionExample		File Folder
VisualStudio Templates		File Folder
BVCustLibMatlab.dll	24 KB	Application Extension
BVFindBlockCode.dll	24 KB	Application Extension
BVFindConv.dll	24 KB	Application Extension
CustomLibFunction.dll	20 KB	Application Extension
FindBlockCode.ctf	874 KB	CTF File
FindBlockCode.dll	9 KB	Application Extension
FindConv.ctf	854 KB	CTF File
FindConv.dll	10 KB	Application Extension
MatlabFunction.ctf	67 KB	CTF File
MatlabFunction.dll	9 KB	Application Extension
MWArray.dll	72 KB	Application Extension
readme.txt	1 KB	Text Document

On Windows Vista:



Name	Date modified	Type
BVCustLibMatlab	13.03.2008 11:13	File Folder
CustomLibFunction	13.03.2008 11:13	File Folder
MatlabFunctionExample	13.03.2008 11:13	File Folder
VisualStudio Templates	13.03.2008 11:13	File Folder
BVCustLibMatlab.dll	07.03.2008 23:10	Application Extens...
BVFindBlockCode.dll	25.02.2008 20:54	Application Extens...
BVFindConv.dll	25.02.2008 20:58	Application Extens...
CustomLibFunction.dll	07.03.2008 10:11	Application Extens...
FindBlockCode.ctf	22.02.2008 22:17	CTF File
FindBlockCode.dll	22.02.2008 22:17	Application Extens...
FindConv.ctf	11.02.2008 21:46	CTF File
FindConv.dll	11.02.2008 21:46	Application Extens...
MatlabFunction.ctf	05.02.2008 12:09	CTF File
MatlabFunction.dll	05.02.2008 12:09	Application Extens...
MWArray.dll	15.08.2007 15:19	Application Extens...
readme.txt	07.03.2008 10:27	Text Document

Four subfolders are also created:

CustomLibFunction

A .NET custom library example. The output of this project is **CustomLibFunction.dll**, which can be used with BitViewTool.

BVCustLibMatlab

Example project for a wrapper for a MatLab custom function. The output of this project is **BVCustLibMatlab.dll.**

MatlabFunctionExample

Example project for MatLab. This is an example that shows how to compile MatLab code into .NET code. The output of this project is **MatlabFunction.dll** and **MatlabFunction.ctf**. Together with **BVCustLibMatlab.dll**, these files build a MatLab custom function that can be used with BitViewTool.

VisualStudio Templates

Templates for Microsoft Visual Studio are provided to create new custom function projects. Please read the **Readme.txt** file that is included in this subfolder for more information.

Adding a Custom Function

Adding a new custom function is simply done by adding its DLL to the **CustomLib** folder.

Windows XP and older:

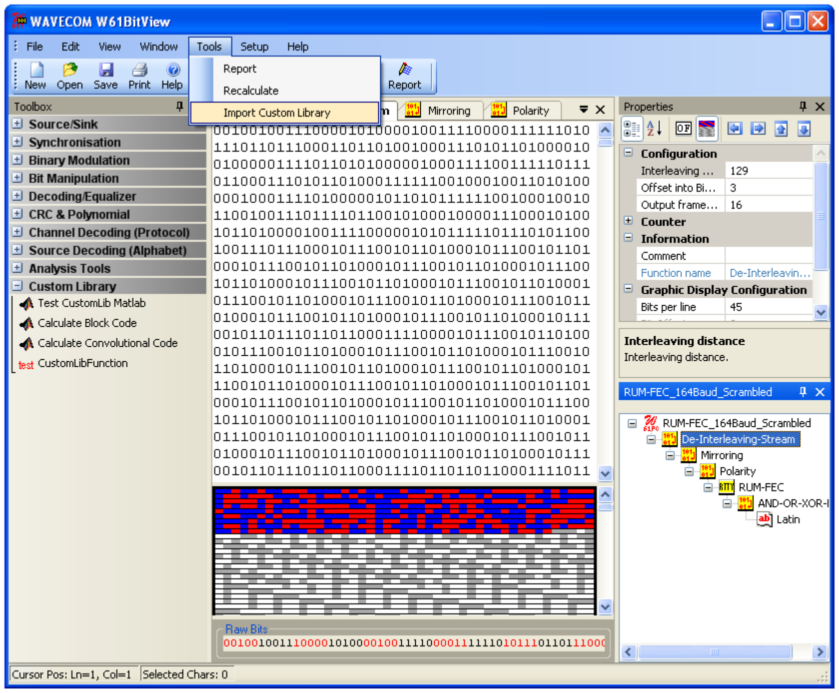
- Documents and Settings\All Users\Documents\WAVECOM\BitViewTool\CustomLib
- Documents and Settings\All Users\Shared Documents\WAVECOM\BitViewTool\CustomLib

Windows Vista:

- Users\Public\Public Documents\WAVECOM\BitViewTool\ CustomLib

BitViewTool automatically adds the function to the Custom Library list in the Toolbox after launch. Alternatively, if BitViewTool is already running, the custom function may be added to the list using the **Import Custom Library** item in the **Tools** menu.

Please be aware that the Custom Library button will not appear in the Toolbox if a custom function is not provided.



```
0010010011100001010000100111100001111111010
1110110111000110110100100011101011010000010
0100000111101101010000010001111001111110111
011000111010110100011111100100010011010100
000100011110100000101101011111100100010010
110010011101111011001010001000011100010100
101101000010011110000010101111101110101100
100111011100010111001011010001011100101101
000101110010110100010111001011010001011100
101101000101110010110100010111001011010001
011100101101000101110010110100010111001011
010001011100101101000101110010110100010111
001011011101101100011110000101110010110100
010111001011010001011100101101000101110010
110100010111001011010001011100101101000101
110010110100010111001011010001011100101101
000101110010110100010111001011010001011100
101101000101110010110100010111001011010001
011100101101000101110010110100010111001011
010001011100101101000101110010110100010111
001011011101101100011110110110110001111011
```

Raw Bits: 00100100111000010100001001111000011111110101110110111000

Configuration	
Interleaving ...	129
Offset into Bi...	3
Output frame...	16
Counter	
Information	
Comment	
Function name	De-Interleavin...
Graphic Display Configuration	
Bits per line	45

Interleaving distance
Interleaving distance.

Cursor Pos: Ln=1, Col=1 | Selected Chars: 0

The icon file for the custom function is named **CustomLibFunction.bmp** and can be customized using the built-in editor in Visual Studio. By compiling the custom function, the icon is integrated into the custom lib DLL.

Constraints

All BitViewTool functions belong to one of three groups:

- Source functions, which have no input.
- Sink functions, which have no output.
- General functions, which have both input and output, and where the input type is BitArray and the output type may be BitArray or string.

Presently custom functions are limited to be general functions with the input and output type is **BitArray**.

Important Notes

When creating custom functions, certain rules must be observed. Please refer to the source code templates and examples for more details.

The following source code elements (names) are mandatory and may not be modified:

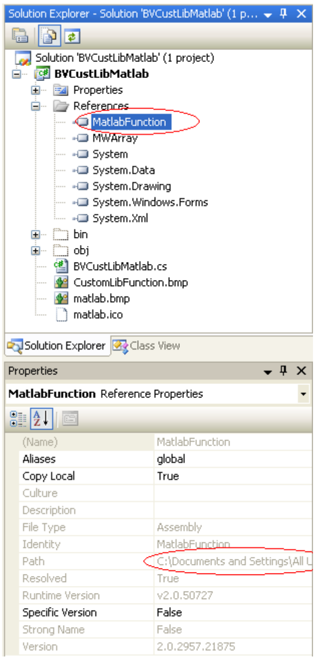
- namespace CustomLib
- public string FunctionName
- public string Comment
- public BitArray Calculate(BitArray buf)

Property categories and additional properties may be defined without limitations.

Steps to Write a Custom Function in C# .NET

- Open the CustomLibFunction example or create a new project based on the BVCustLibFunc template
- Change assembly and class name according to your needs
- Modify the properties and expand the calculate function with your code
- Compile the project and copy the output DLL to the CustomLib folder (see above)

Steps to Write a Custom Function with MatLab



(Name)	MatlabFunction
Aliases	global
Copy Local	True
Culture	
Description	
File Type	Assembly
Identity	MatlabFunction
Path	C:\Documents and Settings\All U
Resolved	True
Runtime Version	v2.0.50727
Specific Version	False
Strong Name	False
Version	2.0.2957.21875

- Open the MatlabFunctionExample project (MatlabFunction.prj) in MatLab
- Change the .M-File according to your needs
- Compile the MatLab code. If the compilation process was successful, there will be two new files in the .\MatlabFunction\distrib subfolder, MatlabFuntion.dll and MatlabFunction.ctf
- Open the BVCustLibMatlab example or create a new project based on the BVCustLibMatlab template
- Change assembly and class name according to your needs
- Set a reference to the MatlabFunction.dll in the \distrib subfolder. Thus, the BVCustLibMatlab class will know where to find the MatLab DLL
- Compile the BVCustLibMatlab project and copy all the files in the output directory to the CustomLib folder

Source Code Template / Example (C# .NET)

CustomLibFunction.cs

```
//---------------------------------------------------------------------------
// File        : CustomLibFunction.cs
// Author      : WAVECOM Elektronik AG
// Date        : February 2008
// Description : Template/example for a custom defined library function.
//
//               Important note:
//               In the project settings the "Assembly Name" must be
//               equal to the "Default Namespace".
//
//---------------------------------------------------------------------------

#region using references

using System;
using System.Collections;
using System.Text;
using System.ComponentModel;
using CustomLibFunction.Properties;

#endregion

namespace CustomLib  // Mandatory! Do not change this name.
{
    /// <summary>
    /// </summary>
    [Serializable]
    [DefaultProperty("FunctionName")] // adjust string if necessary, it points to a property
below
    public class CustomLibFunction
    {
        #region Constant fields

        private class PropertyCategory
        {
            // these are the categories in the BitViewTool property grid (Parameter Window)
            public const string Information = "Information";
            public const string Counter = "Counter";
            public const string Configuration = "Configuration";
            public const string Calculation = "Calculation";

            // do not change these names, they are used to distinguish the different catego-
ries
            // when any change in the parameters happens
        }

        private const int MAX_SIZE = 500000; // do not modify this value

        #endregion

        #region Fields

        // generate a new BitArray for calculation, it will be returned to the calling func-
tion
        private BitArray outbox = new BitArray(MAX_SIZE);

        #endregion

        #region Mandatory Properties

        private string functionName = "CustomLibFunction";
        /// <summary>
        /// Mandatory! Do not delete this property! The string "CustomLibFunction" may be
modified.
        /// This will be the name of the Function in the BitViewTool History explorer and
ToolBox
        /// </summary>
        [Category(PropertyCategory.Information)]
        [Description("Description of CustomLibFunction goes here.\n" +
                      "Here: example of a custom defined library function. Inverts all the
input bits.\n")]
        [DisplayName("Function Name")]
        public string FunctionName
        {
            get
            {
                return (functionName);
            }
        }


        private string comment = string.Empty;
        /// <summary>
        /// Mandatory! Do not modify this property!
        /// </summary>
        [Category(PropertyCategory.Information)]
        [Description("My comments.")]
        [DefaultValue("")]
        public string Comment
        {
            get
            {
                return comment;
            }
            set
            {
                comment = value;
            }
        }

        private static System.Drawing.Image iconImage = Resources.CustomLibFunction;
        /// <summary>
        /// If available, you can specify a custom image displayed on the function button
        /// in the BitView toolbox. The image must be 16x16 pixel. The bitmap file must be
        /// imported into the CustomLibFunction resources, so that it can be refered to it as
shown above!
        /// Currently the image is the .bmp file 'CustomFunction.bmp'. It is part of this so-
lution, it can be
        /// seen in the Solution Explorer Window and it can be modified.
        /// If there is no image available, remove all private and public fields, i.e. iconI-
mage and IconImage,
        /// iconTransparentColor and IconTransparentColor
        /// below. A default icon image is then added by the BitViewTool main application.
        /// </summary>
        [Category(PropertyCategory.Information)]
        [Description("Icon bitmap for Toolbox Button and History Explorer")]
        [Browsable(false)]
        public static System.Drawing.Image IconImage
        {
            get
            {
                return iconImage;
            }
        }

        private     static     System.Drawing.Color     iconTransparentColor     =     Sys-
tem.Drawing.Color.White;
        /// <summary>
        /// If available, you can specify a custom image display on the function button
        /// in the toolbox. The image must be 16x16 pixel, the transparent color must
        /// be specified here
        /// </summary>
        [Category(PropertyCategory.Information)]
        [Description("Icon bitmap transparent color")]
        [Browsable(false)]
        public static System.Drawing.Color IconTransparentColor
        {
            get
            {
                return iconTransparentColor;
            }
        }

        #endregion

        #region Optional Properties
        private string dummyCalculation = string.Empty;
        /// <summary>
        /// Optional result form this function! DummyCalculation. Put all results into this
category
        /// </summary>
        [Category(PropertyCategory.Calculation)]
        [Description("Description of DummyCalculation goes here. Here: number of output
bits.")]
        [DisplayName("Dummy Calc.")]
        public string DummyCalculation
        {
            get
            {
                return (dummyCalculation);
            }
        }


        // add more Calculation properties here
        // ...


        private int dummyParameter = 0;
        /// <summary>
        /// Optional! Dummy Parameter. Put all input parameters for the calculation into this
category
        /// </summary>
        [Category(PropertyCategory.Configuration)]
        [Description("Description of DummyParameter goes here. Here: not used.")]
        [DefaultValue(0)]
        [DisplayName("Dummy Parameter")]
        public int DummyParameter
        {
            get
            {
                return (dummyParameter);
            }
            set
            {
                dummyParameter = value;
            }
        }


        // add more Configuration properties here
        // ...


        #endregion

        #region Constructor
        /// <summary>
        /// Constructor
        /// </summary>
        public CustomLibFunction()
        {
            // add initialisation code if necessary
        }
        #endregion

        #region calculation function
        /// <summary>
        /// // THE calculate function. Do not change name or parameter! This function will be
searched for
        /// by reflection and called for execution, when this library is added to an analy-
sis.
        /// </summary>
        public BitArray Calculate(BitArray buf)
        {
            if (buf == null)
                return (null);

            if (buf.Count <= 0)
            {
                outbox.Length = 0;
                return (outbox);
            }

            //---------------------------------------------------------------------
            // any calculation goes here
            //---------------------------------------------------------------------

            outbox = (BitArray)buf.Clone();
            outbox.Not();

            this.dummyCalculation = outbox.Count.ToString();     // any calculation results of
interest will be reflected in the property grid

            return (outbox);    // must return the result

        }
        #endregion
    }
}
```

Source Code Template / Example (C# .NET for MatLab)

BVCustLibMatlab.cs

```
//---------------------------------------------------------------------------
// File        : BVCustLibMatlab.cs
// Author      : WAVECOM Elektronik AG
// Date        : February 2008
// Description : Template/example for a custom defined library function.
//
//               Important note:
//               In the project settings the "Assembly Name" must be
//               equal to the "Default Namespace".
//
//---------------------------------------------------------------------------

#region using references

using System;
using System.Windows.Forms;
using System.Collections;
using System.Text;
using System.ComponentModel;
using System.Diagnostics;
using BVCustLibMatlab.Properties;

// if working with MatLab .net DLLs
using MathWorks.MATLAB.NET.Utility;
using MathWorks.MATLAB.NET.Arrays;

// reference to the MatLab dll
using MatlabFunction;

#endregion

namespace CustomLib  // Mandatory! Do not change this name.
{
    /// <summary>
    /// </summary>
    [Serializable]
    [DefaultProperty("FunctionName")] // adjust string if necessary, it points to a property
below
    public class BVCustLibMatlab
    {
        #region Constant fields

        private class PropertyCategory
        {
            // these are the categories in the BitViewTool property grid (Parameter Window)
            public const string Information = "Information";
            public const string Counter = "Counter";
            public const string Configuration = "Configuration";
            public const string Calculation = "Calculation";
            // do not change these names, they are used to distinguish the different catego-
ries
            // when any change in the parameters happens
        }

        private const int MAX_SIZE = 500000; // do not modify this value, it specifies maxi-
mum number of bits processed by BitViewTool

        #endregion

        #region Fields

        // generate a new BitArray for calculation, it will be returned to the calling func-
tion
        private BitArray outbox = new BitArray(MAX_SIZE);

        #endregion

        #region Mandatory Properties

        private string functionName = "Test CustomLib MatLab";
        /// <summary>
        /// Mandatory! Do not delete this property! The string "BVCustLibMatlab" may be mod-
ified.
        /// This will be the name of the Function in the BitViewTool History explorer and
ToolBox
        /// </summary>
        [Category(PropertyCategory.Information)]
        [Description("Description of Test CustomLib MatLab goes here.\n" +
                     "Here: example of a custom defined library function. Inverts all the
input bits.\n" +
                       "Inversion is performed by a MatLab function encapsulated in a .net
dll assembly.")]
        [DisplayName("Function Name")]
        public string FunctionName
        {
            get
            {
                return (functionName);
            }
        }


        private string comment = string.Empty;
        /// <summary>
        /// Mandatory! Do not modify this property!
        /// </summary>
        [Category(PropertyCategory.Information)]
        [Description("My comments.")]
        [DefaultValue("")]
        public string Comment
        {
            get
            {
                return comment;
            }
            set
            {
                comment = value;
            }
        }

        private static System.Drawing.Image iconImage = Resources.matlab;
        /// <summary>
        /// If available, you can specify a custom image displayed on the function button
        /// in the BitView toolbox. The image must be 16x16 pixel. The bitmap file must be
        /// imported into the CustomLibFunction resources, so that it can be refered to it as
shown above!
        /// Currently the image is the .bmp file 'CustomFunction.bmp'. It is part of this so-
lution, it can be
        /// seen in the Solution Explorer Window and it can be modified.
        /// If there is no image available, remove all private and public fields, i.e. iconI-
mage and IconImage,
        /// iconTransparentColor and IconTransparentColor
        /// below. A default icon image is then added by the BitViewTool main application.
        /// </summary>
        [Category(PropertyCategory.Information)]
        [Description("Icon bitmap for Toolbox Button and History Explorer")]
        [Browsable(false)]
        public static System.Drawing.Image IconImage
        {
            get
            {
                return iconImage;
            }
        }

        private     static     System.Drawing.Color     iconTransparentColor     =     Sys-
tem.Drawing.Color.FromArgb(224, 223, 227);
        /// <summary>
        /// If available, you can specify a custom image display on the function button
        /// in the toolbox. The image must be 16x16 pixel, the transparent color must
        /// be specified here
        /// </summary>
        [Category(PropertyCategory.Information)]
        [Description("Icon bitmap transparent color")]
        [Browsable(false)]
        public static System.Drawing.Color IconTransparentColor
        {
            get
            {
                return iconTransparentColor;
            }
        }

        #endregion

        #region Optional Properties

        //  Input Parameter for calculation function

        private int inpar1 = 13; // set a reasonable default value
        /// <summary>
        /// optional parameter for calculation, put this into the Configuration category
        /// </summary>
        [Category(PropertyCategory.Configuration)]
        [Description("Description of Input 1 goes here")]
        [DefaultValue(13)]   // set a reasonable default value
        [DisplayName("Input 1")]
        public int Inpar1
        {
            get
            {
                return inpar1;
            }
            set
            {
                inpar1 = value;
            }
        }


        private int inpar2 = 5; // set a reasonable default value
        /// <summary>
        /// optional parameter for calculation, put this into the Configuration category
        /// </summary>
        [Category(PropertyCategory.Configuration)]
        [Description("Description of Input 2 goes here")]
        [DefaultValue(5)]   // set a reasonable default value
        [DisplayName("Input 2")]
        public int Inpar2
        {
            get
            {
                return inpar2;
            }
            set
            {
                inpar2 = value;
            }
        }


        // add more Configuration properties here
        // ...



        //  results from calculation function

        private int outpar1 = 0;
        /// <summary>
        /// Optional result form this function! Put all results into the calculation category
        /// </summary>
        [Category(PropertyCategory.Calculation)]
        [Description("Description of Output 1 goes here.")]
        [DisplayName("Output 1")]
        public int Outpar1
        {
            get
            {
                return outpar1;
            }
        }


        private int outpar2 = 0;
        /// <summary>
        /// Optional result form this function! Put all results into the calculation category
        /// </summary>
        [Category(PropertyCategory.Calculation)]
        [Description("Description of Output 2 goes here.")]
        [DisplayName("Output 2")]
        public int Outpar2
        {
            get
            {
                return outpar2;
            }
        }

        private int outpar3 = 0;
        /// <summary>
        /// Optional result form this function! Put all results into the calculation category
        /// </summary>
        [Category(PropertyCategory.Calculation)]
        [Description("Description of Output 3 goes here.")]
        [DisplayName("Output 3")]
        public int Outpar3
        {
            get
            {
                return outpar3;
            }
        }

        private int outpar4 = 0;
        /// <summary>
        /// Optional result form this function! Put all results into the calculation category
        /// </summary>
        [Category(PropertyCategory.Calculation)]
        [Description("Description of Output 4 goes here.")]
        [DisplayName("Output 4")]
        public int Outpar4
        {
            get
            {
                return outpar4;
            }
        }


        private string outpar5 = string.Empty;
        /// <summary>
        /// Optional result form this function! Put all results into this category
        /// </summary>
        [Category(PropertyCategory.Calculation)]
        [Description("Description of Output 5 (string) goes here.")]
        [DisplayName("Output 5")]
        public string Outpar5
        {
            get
            {
                return (outpar5);
            }
        }


        // add more Calculation properties here
        // ...

        #endregion

        #region Constructor
        /// <summary>
        /// Constructor
        /// </summary>
        public BVCustLibMatlab()
        {
            // add initialisation code if necessary
        }
        #endregion

        #region calculation function
        /// <summary>
        /// // THE calculate function. Do not change name or parameter! This function will be
searched for
        /// by reflection and called for execution, when this library is added to an analy-
sis.
        /// </summary>
        public BitArray Calculate(BitArray buf)
        {
            if (buf == null)
                return (null);

            if (buf.Count <= 0)
            {
                outbox.Length = 0;
                return (outbox);
            }

            //---------------------------------------------------------------------
            // any calculation goes here
            //---------------------------------------------------------------------

            try      // catch any exception from matlab dll. If calling the matlab function
fails, we do not crash the app
            {
                // Generate data arrays for MatLab function.
                // Data input for matlab function is derived from buf, which is type of Bi-
tArray
                // MatLab has not a compatible data type, so we have to convert BitArray buf
                // into the MatLab default numeric type double 0.0 and 1.0

                // new instance of a MatLab numeric array with a size of [buf.Length,1]
                MWNumericArray data = new MWNumericArray(MWArrayComplexity.Real, MWNumeric-
Type.Double, new int[] { buf.Length, 1 });

                // Initialize data from BitArray buf
                // MatLab index range is from 1...n, in C# it is from 0 ... (n-1)
                for (int idx = 1; idx <= buf.Length; idx++)
                {
                    data[idx, 1] = buf[idx - 1] ? 1.0 : 0.0;
                }

                // data output from matlab function is an Array of arguments
                MWArray[] argsOut = null;

                // prepare for calling the MatLab function, create an instance of the MatLab
function class
                // generated by the MatLab deploytool, use Intellisense in visual studio,
then you see
                // the list of classes found up to now in this project. Our example is
                // MatLab function called 'MatlabFunction' in the .m File with the same name.
                // Here is the complete listing of the MatLab .m File:

                /*------------------------------------------------------------------------
--------------------
```

```
                function [y, outpar1, outpar2, outpar3, outpar4, outpar5] = MatlabFunction(
x, inpar1, inpar2 )
                %
                %--------------------------------------------------------------------------
                % Example of a function to be called from .Net environment, especially from
                % a BitViewTool CustomLib function. This function declares different input
                % and output parameters, to demonstrate how to access these parameters from
                % a .Net environment.
                %
                % Inputs:
                % -------
                %
                % x      :  should be a [n,1] input array containing the bit stream from the
                %           BitViewTool, the expected values must be 0 and 1
                % inpar1, inpar2:  these are parameters to control the function's behaviour
                %
                % Outputs:
                % --------
                %
                % y      :  should be a [k,1] output array containing the bit stream as an
                %           output from this function, the type of y should be double or logi-
cal.
                % The values must be 0.0 and 1.0 if double.
                % outpar1, outpar2, outpar3, outpat4, outpar5 : these are additional calcula-
tion
                % results from this function of type scalar or string depending on the
                % function's behaviour. outpar5 is of type char array
                %
                %
                %--------------------------------------------------------------------------
                % After this function is debugged and tested, the MatLab deploytool has to be
                % started with 'deploytool' in the Command Window
                % In the deploytool create a new .net project, project type is .NET
                % Component, enter a Component name, in the project settings under .NET set
the
                % the Microsoft Framework to Version 2.0, the Assembly type to private.
                % Then push the Build Project button in the Deployment toolbar.
                % When finished copy the .dll and .ctf files from the function's
                % project\distrib directory to the BitViewTool customlib directory
                %              C:\Documents       and       Settings\All       Us-
ers\Documents\WAVECOM\BitViewTool\CustomLib
                %
                % Next step is to create a new WAVECOM CustomLib function from the template
                % found in the new project wizzard in Visual Studio.
                % Add a the reference in the Solution Explorer to the .dll just copied to to
                % CustomLib directory. Add a reference in the C# CustomLib source under the
                % 'using' region, the namespace to be used is the same used in the solution
                % explorer.
                %
                %--------------------------------------------------------------------------

                outpar1 = nargin;   % for example : outpar1 returns number of input function
arguments
                [m, n] = size( x );   % for example : outpar2 returns length of input array
x, i.e. number of rows
                outpar2 = m;
                outpar3 = n;        % number of columns, should be 1 in our example

                outpar4 = inpar1 * inpar2;  % example of calculation for outpar4

                y = ~x;              % this is the only calculation for the input data, outpt
y is inverse(x)
                                     % and converts y to type 'logical' !!!!
                if (n ~= 1)
                    outpar5 = 'Function error: input data dimension [m,n] with n ~= 1';
                else
                    outpar5 = 'Function called successfully';
                end
                ----------------------------------------------------------------------------
--------
               */


                // we can now see 3 Input parameters and 6 output parameters
                // x is the input data column vector, type is double
                // y is the returned column vector, type can be double or logical,
                // all other parameters are scalars, expected
                // to be of type double, outpar5 is of type string
                // Use Intellisense to see all available classes
                // the matlab deploytool generates a class by using the MatLab function name
                // with 'class' appended:

                MatlabFunctionclass MatlabFunc = new MatlabFunctionclass(); // new instance
of MatLab function

                // now call this function, with 6 output parameters : y and outpar 1...5
                argsOut = MatlabFunc.MatlabFunction(6, data, (double)Inpar1, (double)Inpar2);

                // now get all results from argsOut, its an array of numeric/char arrays
                // the first entry is our bit array result, can be of type numeric(double) or
logical,
                // depending on the matlab function internal code

                if  ((argsOut[0].IsNumericArray)  &&  (argsOut[0].NumberofDimensions  ==  2))
// if numeric and number of dimension exactly 2
                {
                    MWNumericArray numericBits = (MWNumericArray)argsOut[0];      // cast to
MWNumericArray
                    MWNumericType numericType = numericBits.NumericType;          // get the
numeric type, can be double, float, int, etc..., this has to be checked
                    Array   numericBitsArray   =   numericBits.ToVector(MWArrayComponent.Real);
// must be anything but not complex !!!
                    int length;

                    switch (numericType)    // check types, we expect double
                    {
                        case MWNumericType.Double:
                            double[] doubleArray = (double[])numericBitsArray;   // cast to
double array
                            length = doubleArray.Length;
                            if (length > MAX_SIZE)
                            {
                                length = MAX_SIZE;
                            }
                            // convert now result to type of BitArray
                            for (int i = 0; i < length; ++i)
                            {
                                outbox[i] = (doubleArray[i] == 0.0) ? false : true;
                            }
                            outbox.Length = length; // set length explicitely, otherwise we
have MAX_SIZE !!
                            break;

                        case MWNumericType.Int32:
                            Int32[] int32Array = (Int32[])numericBitsArray;  // cast to int32
array
                            length = int32Array.Length;
                            if (length > MAX_SIZE)
                            {
                                length = MAX_SIZE;
                            }
                            // convert now result to type of BitArray
                            for (int i = 0; i < length; ++i)
                            {
                                outbox[i] = (int32Array[i] == 0) ? false : true;
                            }
                            outbox.Length = length; // set length explicitely, otherwise we
have MAX_SIZE !!
                            break;

                        case MWNumericType.Int16:
                            Int16[] int16Array = (Int16[])numericBitsArray;  // cast to int16
array
                            length = int16Array.Length;
                            if (length > MAX_SIZE)
                            {
                                length = MAX_SIZE;
                            }
                            // convert now result to type of BitArray
                            for (int i = 0; i < length; ++i)
                            {
                                outbox[i] = (int16Array[i] == 0) ? false : true;
                            }
                            outbox.Length = length; // set length explicitely, otherwise we
have MAX_SIZE !!
                            break;

                        case MWNumericType.Int8:
                            Byte[] byteArray = (Byte[])numericBitsArray;   // cast to int16
array
                            length = byteArray.Length;
                            if (length > MAX_SIZE)
                            {
                                length = MAX_SIZE;
                            }
                            // convert now result to type of BitArray
                            for (int i = 0; i < length; ++i)
                            {
                                outbox[i] = (byteArray[i] == 0) ? false : true;
                            }
                            outbox.Length = length; // set length explicitely, otherwise we
have MAX_SIZE !!
                            break;


                        default:    // other numeric types not supported
                            throw new ApplicationException("Bad type returned from " + Func-
tionName);
                    }
                }
                else if ((argsOut[0].IsLogicalArray) && (argsOut[0].NumberofDimensions == 2))
// if logical and number of dimension exactly 2
                {
                    MWLogicalArray logicalBits = (MWLogicalArray)argsOut[0];          // cast
to MWLogicalArray

                    bool[] boolOutputArray;
                    boolOutputArray = logicalBits.ToVector();        // convert all to a bool
array in .net
                    outbox = new BitArray(boolOutputArray);          // then initialize a Bi-
tArray for use with BitViewTool
                }
                else
                {
                    throw new ApplicationException("Bad type returned from " + FunctionName);
                }

                // now get other results, we do not check types again, but you should do that
                outpar1 = (int)(MWNumericArray)argsOut[1];
                outpar2 = (int)(MWNumericArray)argsOut[2];
                outpar3 = (int)(MWNumericArray)argsOut[3];
                outpar4 = (int)(MWNumericArray)argsOut[4];
                outpar5 = ((MWCharArray)argsOut[5]).ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Errors in " + FunctionName);
                outbox.Length = 0;
            }

            return (outbox);

        }
        #endregion
    }
}
```

Source Code Template / Example (MatLab)

MatlabFunction.m

```
function [y, outpar1, outpar2, outpar3, outpar4, outpar5] = MatlabFunction( x, inpar1, inpar2
)
%
%-----------------------------------------------------------------------
% Example of a function to be called from .Net environment, especially from
% a BitViewTool CustomLib function. This function declares different input
% and output parameters, to demonstrate how to access these parameters from
% a .Net environment.
%
% Inputs:
% -------
%
% x     :  should be a [n,1] input array containing the bit stream from the
%          BitViewTool, the expected values must be 0 and 1
% inpar1, inpar2:  these are parameters to control the function's behaviour
%
% Outputs:
% --------
%
% y     :  should be a [k,1] output array containing the bit stream as an
%          output from this function, the type of y should be double or logical.
% The values must be 0.0 and 1.0 if double.
% outpar1, outpar2, outpar3, outpat4, outpar5 : these are additional calculation
% results from this function of type scalar or string depending on the
% function's behaviour. outpar5 is of type char array
%
%-----------------------------------------------------------------------
% After this function is debugged and tested, the MatLab deploytool has to be
% started with 'deploytool' in the Command Window
% In the deploytool create a new .net project, project type is .NET
% Component, enter a Component name, in the project settings under .NET set the
% the Microsoft Framework to Version 2.0, the Assembly type to private.
% Then push the Build Project button in the Deployment toolbar.
% When finished copy the .dll and .ctf files from the function's
% project\distrib directory to the BitViewTool customlib directory
%  C:\Documents and Settings\All Users\Documents\WAVECOM\BitViewTool\CustomLib
%
% Next step is to create a new WAVECOM CustomLib function from the template
% found in the new project wizzard in Visual Studio.
% Add a the reference in the Solution Explorer to the .dll just copied to to
% CustomLib directory. Add a reference in the C# CustomLib source under the
% 'using' region, the namespace to be used is the same used in the solution
% explorer.
%
%-----------------------------------------------------------------------

outpar1 = nargin;   % for example : outpar1 returns number of input function arguments
[m, n] = size( x );   % for example : outpar2 returns length of input array x, i.e. number of
rows
outpar2 = m;
outpar3 = n;        % number of columns, should be 1 in our example

outpar4 = inpar1 * inpar2;  % example of calculation for outpar4

y = ~x;             % this is the only calculation for the input data, output y is inverse(x)
                % this converts y to type 'logical' !!!!

if (n ~= 1)
    outpar5 = 'Function error: input data dimension [m,n] with n ~= 1';
else
    outpar5 = 'Function called successfully';
end
```

Appendix

CodeMeter and CmStick

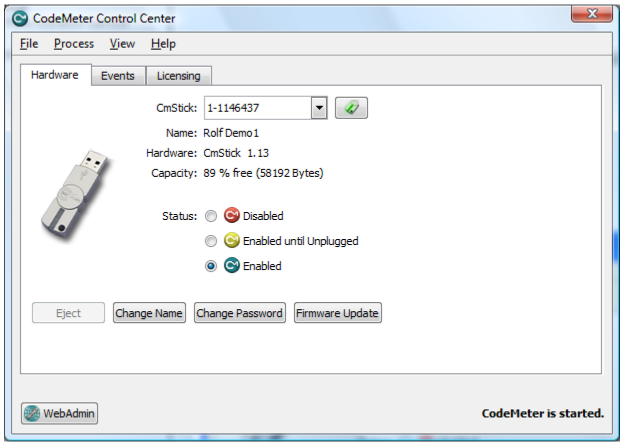
Important: For current and detailed information consult the CodeMeter help files.

CodeMeter Control Center

When the CodeMeter has been installed the CodeMeter Control Center may be opened by clicking the CodeMeter icon in the Windows system tray.
The Control Center offers several configuration options via its menu bar and its three tabs in addition to basic information regarding the installation.
The tabs cover the areas listed below

- **Hardware**
- **Events**
- **Licensing**

CodeMeter Hardware

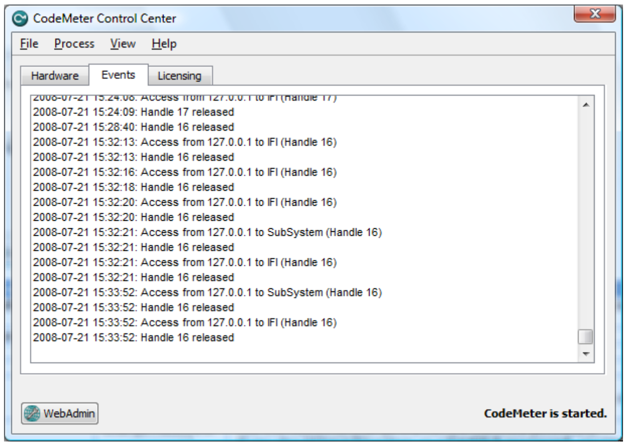


CmStick: 1-1146437
Name: Rolf Demo1
Hardware: CmStick 1.13
Capacity: 89 % free (58192 Bytes)
Status: Disabled / Enabled until Unplugged / [✓ Enabled]

CodeMeter is started.

Hardware Page	Remarks
WebAdmin	Starts the CodeMeter WebAdministrator in your Internet standard browser
Eject	Used to safely remove (a) CmStick(s)
Change Name	Use this button to change the name of the selected CmStick. Just click the button "Change Name" and enter the new name into the field in the “Change Name” window
Change Password	Use this button to change the current CmStick password. If you should have forgotten your CmStick password, you can use your CodeMeter Master-Password to create a new CmStick password
Firmware Update	Use this button to update the firmware of the CmStick to the current version. To execute the firmware update an Internet connection is required. Do not remove your CmStick during the CodeMeter firmware update. This can cause serious damage to the CmStick

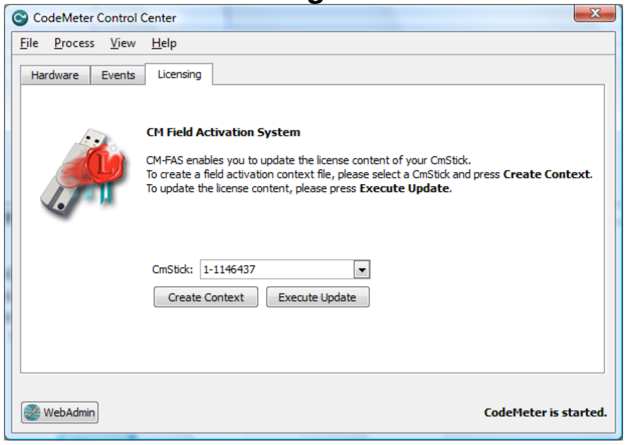
CodeMeter Events



```
2008-07-21 15:24:08: Access from 127.0.0.1 to IFI (Handle 17)
2008-07-21 15:24:09: Handle 17 released
2008-07-21 15:28:40: Handle 16 released
2008-07-21 15:32:13: Access from 127.0.0.1 to IFI (Handle 16)
2008-07-21 15:32:13: Handle 16 released
2008-07-21 15:32:16: Access from 127.0.0.1 to IFI (Handle 16)
2008-07-21 15:32:18: Handle 16 released
2008-07-21 15:32:20: Access from 127.0.0.1 to IFI (Handle 16)
2008-07-21 15:32:20: Handle 16 released
2008-07-21 15:32:21: Access from 127.0.0.1 to SubSystem (Handle 16)
2008-07-21 15:32:21: Handle 16 released
2008-07-21 15:32:21: Access from 127.0.0.1 to IFI (Handle 16)
2008-07-21 15:32:21: Handle 16 released
2008-07-21 15:33:52: Access from 127.0.0.1 to SubSystem (Handle 16)
2008-07-21 15:33:52: Handle 16 released
2008-07-21 15:33:52: Access from 127.0.0.1 to IFI (Handle 16)
2008-07-21 15:33:52: Handle 16 released
```

Events Page	Remarks
WebAdmin	Starts the CodeMeter WebAdministrator in your Internet standard browser
Events	The Event window contains general information on: • The number of the connected CmSticks detected • The number of available CmStick entries • The number of Firm Items • All access to the CodeMeter Runtime Server

CodeMeter Licensing



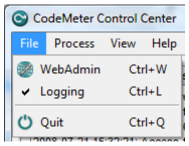
CM Field Activation System

CM-FAS enables you to update the license content of your CmStick.
To create a field activation context file, please select a CmStick and press **Create Context**.
To update the license content, please press **Execute Update**.

CmStick: 1-1146437

Licensing Page	Remarks
WebAdmin	Starts the CodeMeter WebAdministrator in your Internet standard browser
CmStick	Select the desired CmStick by selecting its serial number
Create Context	Use this option to create a Remote Context file for the selected CmStick
Execute Update	Use this option to write the data from a Remote Update file into the selected CmStick. Further information can be found in the Field Activation Services (CmFAS) page of the CodeMeter help file

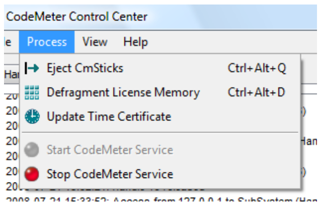
CodeMeter File



File Menu	Remarks
WebAdmin	Starts the CodeMeter WebAdministrator in your Internet standard browser
Logging	Activate/deactivate the logging process. If activated CodeMeter will automatically create a logging file in the CodeMeter installation folder (default[windows]: [..\ProgramFiles\CodeMeter\logs])

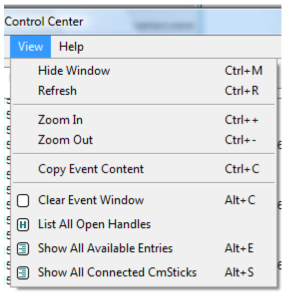
Quit	Closes the CodeMeter Control Center window without stopping the CodeMeter Runtime Server process

CodeMeter Process



Process Menu	Remarks
Eject CmSticks	Used to safely remove (a) CmStick(s)
Defragment License Memory	Defragment the memory part in the CodeMeter Chip (affects selected CmStick)
Update Time Certificate	Performs a Certified Time update for the selected CmStick
Start CodeMeter Service	Starts the CodeMeter Runtime Server service/daemon
Stop CodeMeter Service	Stops the CodeMeter Runtime Server service/daemon

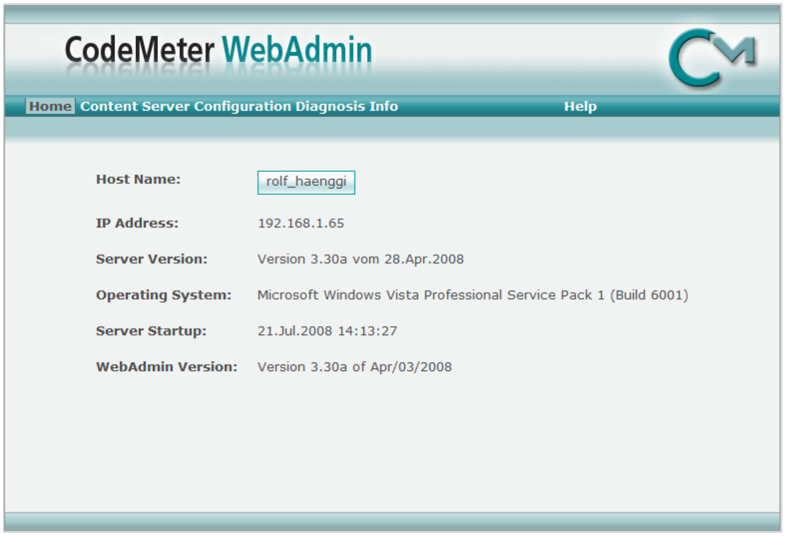
CodeMeter View



View Menu	Remarks
Show all Connected CmSticks	Prints a list of all connected CmSticks
Hide Window	Hides the CodeMeter Control Center
Show All Available Entries	Prints a list of all available CmStick entries
List All Open Handles	Prints a list of all currently opened and used CodeMeter handles. This information is useful for developers
Clear Event Windows	Clear all entries in the event window
Copy Event Contents	Copy the contents of the event window
Zoom In	Increases the font size of the event window content
Zoom Out	Decreases the font size of the event window content

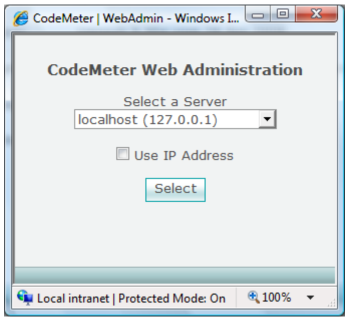
CodeMeter WebAdmin

CodeMeter WebAdmin Home



Host Name:	rolf_haenggi
IP Address:	192.168.1.65
Server Version:	Version 3.30a vom 28.Apr.2008
Operating System:	Microsoft Windows Vista Professional Service Pack 1 (Build 6001)
Server Startup:	21.Jul.2008 14:13:27
WebAdmin Version:	Version 3.30a of Apr/03/2008

The home page displays general information about your computer and the CodeMeter WebAdministrator. By pressing the button with the DNS name of your PC, the server selection pop-up window will appear.

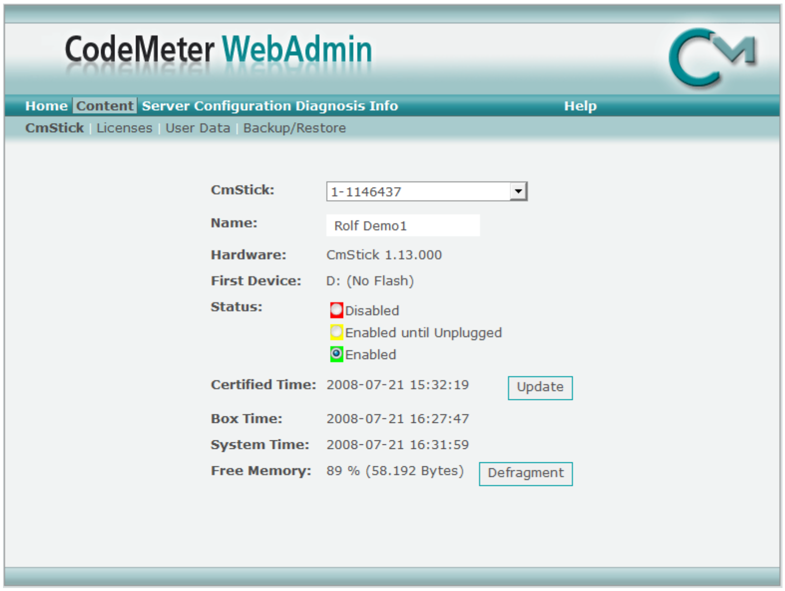

CodeMeter | WebAdmin - Windows I...
CodeMeter Web Administration
Select a Server
localhost (127.0.0.1)
Use IP Address
Select
Local intranet | Protected Mode: On
100%

By using the drop-down menu, it is possible to select a different CodeMeter Server.

CodeMeter WebAdmin Content

The **Content** page is split into four subsections:

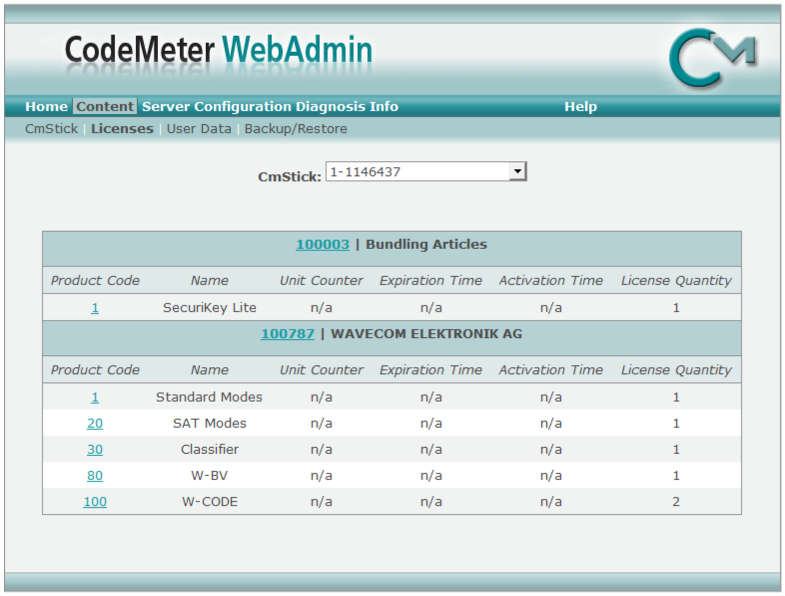
- **CmStick**
- **Licenses**
- **User Data**
- **Backup/Restore**


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
CmStick | Licenses | User Data | Backup/Restore
Update
Defragment

CmStick:	1-1146437
Name:	Rolf Demo1
Hardware:	CmStick 1.13.000
First Device:	D: (No Flash)
Status:	Disabled Enabled until Unplugged [✓ Enabled]
Certified Time:	2008-07-21 15:32:19
Box Time:	2008-07-21 16:27:47
System Time:	2008-07-21 16:31:59
Free Memory:	89 % (58.192 Bytes)

The **CmStick** page offers different CmStick management options.

Content CmStick	Remarks
CmStick	You can select a CmStick via the drop-down menu by choosing its serial number
Name	Shows the name for the selected CmStick
Hardware	Shows the hardware version of the selected CmStick
First Device	Shows the drive letter and the size of the first CodeMeter partition, if a CmStick/M (CmStick with flash memory) is connected
Status	Shows the current status of the selected CmStick. You can change the status of your CmStick by using the CodeMeter Control Center
Certified Time	Displays the current internal Certified Time of the selected CmStick. You can get a Certified Time update from the defined CodeMeter Time Server by clicking on the “Update” button
Box Time	Shows the Box Time (internal time) of your CmStick
System Time	Displays the selected CmStick’s internal clock time
Free Memory	Displays the selected CmStick’s available memory. To defragment the CmStick memory, click the “Defragment” button


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
CmStick | Licenses | User Data | Backup/Restore

CmStick: 1-1146437

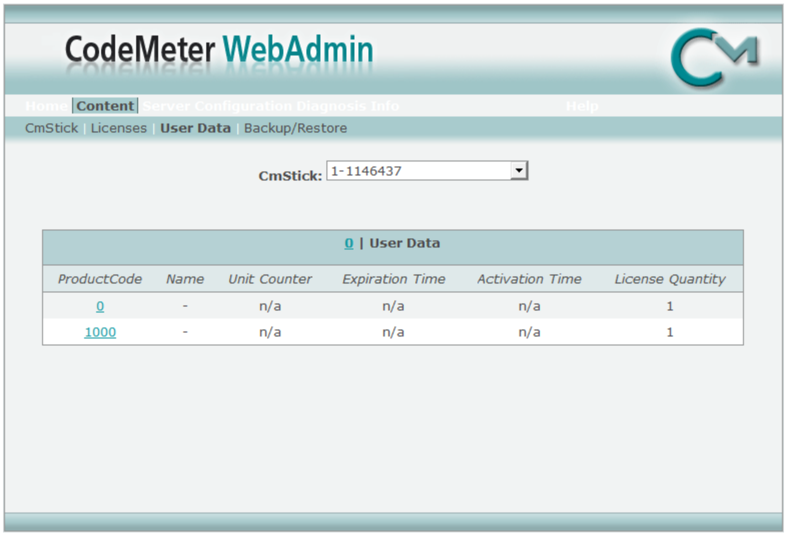
100003 | Bundling Articles					
Product Code	Name	Unit Counter	Expiration Time	Activation Time	License Quantity
1	SecuriKey Lite	n/a	n/a	n/a	1
100787 | WAVECOM ELEKTRONIK AG					
Product Code	Name	Unit Counter	Expiration Time	Activation Time	License Quantity
1	Standard Modes	n/a	n/a	n/a	1
20	SAT Modes	n/a	n/a	n/a	1
30	Classifier	n/a	n/a	n/a	1
80	W-BV	n/a	n/a	n/a	1
100	W-CODE	n/a	n/a	n/a	2

The **Licenses** page shows all licenses (Firm Items) contained in the selected CmStick. Beneath each Firm Item, which is characterized by its Firm Code and "Name", a list of all corresponding Product Codes is displayed.

To get an overview of all stored products for any Firm Item, just click on the Firm Code entry.

For detailed information about a Product Item click on the Product Code entry in the Product Item field

Content License	Remarks
Product Code	Shows the Product Code for the Product Item
Name	Displays the Product Item Text, normally the name of the product
Unit Counter	Displays (if available) the remaining credits for this product
Expiration Time	Displays the Expiration Time of the product, if it is available
Activation Time	Displays the Activation Time of the product, if it is available
License Quantity	Displays the number of licenses that are available for this product


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
CmStick | Licenses | User Data | Backup/Restore

CmStick: 1-1146437

0 | User Data

ProductCode	Name	Unit Counter	Expiration Time	Activation Time	License Quantity
0	-	n/a	n/a	n/a	1
1000	-	n/a	n/a	n/a	1

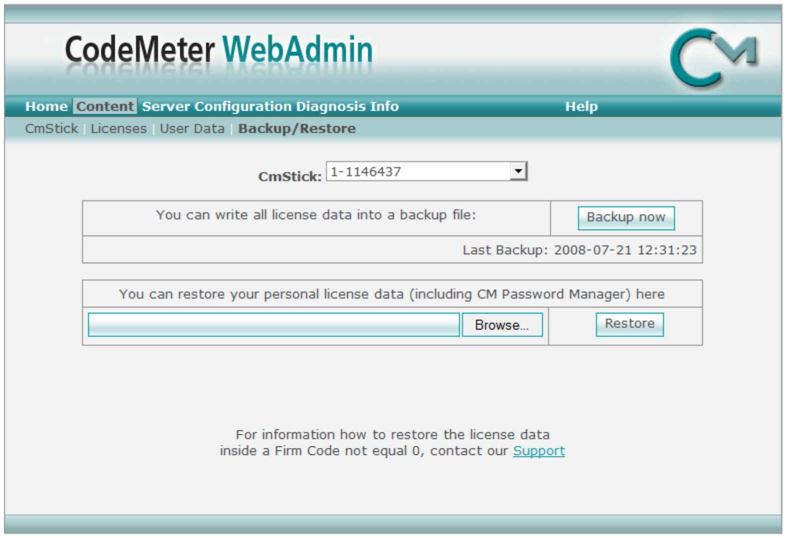
The **User Data** page shows detailed information about the Firm Code 0 (Implicit Firm Item).

The Implicit Firm Item or the Firm Code 0 is "The user's own Firm Code". Every time a program wants to create a new entry into this Firm Code, CodeMeter asks for the CmStick password. CodeMeter will only allow the program to create a new entry or to change data if the password is correctly entered.

The User Data is saved automatically and can be restored if needed.

Further information about the backup and restore functionality can be found on the **Backup/Restore** page.

The table structure is the same as for the "Content License" table.


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
CmStick | Licenses | User Data | Backup/Restore
Backup now
Browse...
Restore

CmStick: 1-1146437

You can write all license data into a backup file:

Last Backup: 2008-07-21 12:31:23

You can restore your personal license data (including CM Password Manager) here

For information how to restore the license data inside a Firm Code not equal 0, contact our Support

On the **Backup/Restore** page, you can create CmStick backups or restore saved user data (licenses).

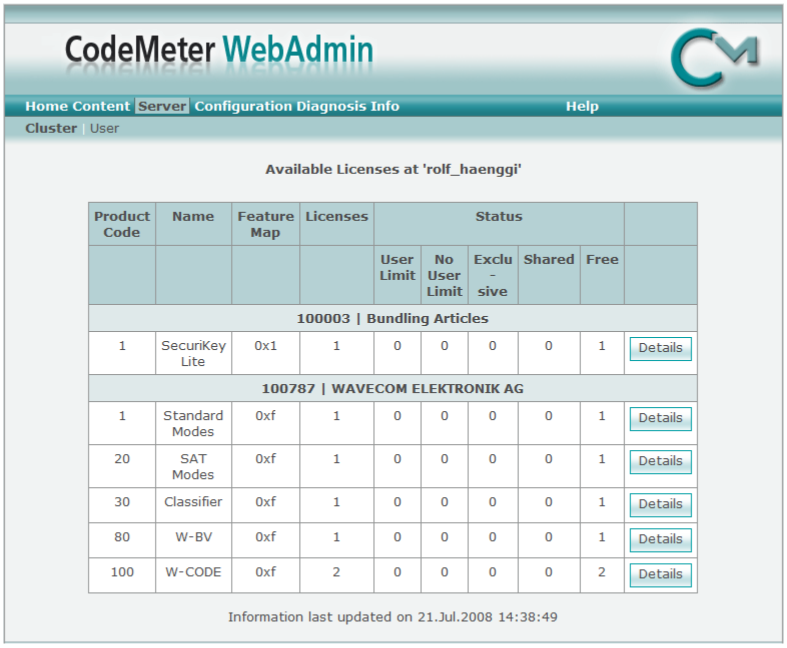
Content License	Remarks
CmStick	With the CmStick drop-down box, you can select the desired CmStick
Section Create Backup	By clicking the "Backup now" button, you can perform a backup of all personal CmStick User Data (User Data page). A backup file will be created in the defined folder (see Configuration Page - Backup) Additionally, the date and the time stamp of your last CmStick backup are displayed

Section Restore Backup	Use this option to restore saved personal license data from a backup file. It is also possible to transfer this data to another CmStick. Therefore the new CmStick must have the same password as the first CmStick

CodeMeter WebAdmin Server

The **Server** page displays detailed information about all available CodeMeter network licenses. The Server page is split into two subsections:

- **Cluster**
- **User**



Available Licenses at 'rolf_haenggi'

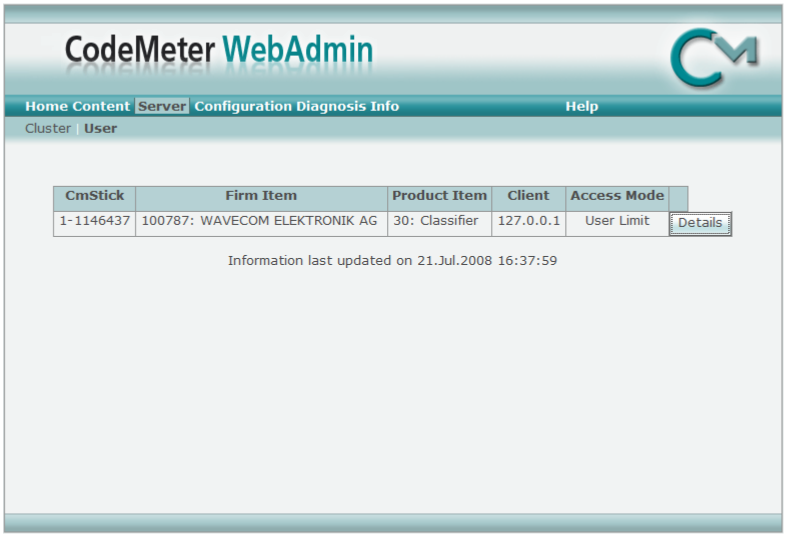
Product Code	Name	Feature Map	Licenses	Status					
				User Limit	No User Limit	Exclu-sive	Shared	Free	
100003 | Bundling Articles									
1	SecuriKey Lite	0x1	1	0	0	0	0	1	Details
100787 | WAVECOM ELEKTRONIK AG									
1	Standard Modes	0xf	1	0	0	0	0	1	Details
20	SAT Modes	0xf	1	0	0	0	0	1	Details
30	Classifier	0xf	1	0	0	0	0	1	Details
80	W-BV	0xf	1	0	0	0	0	1	Details
100	W-CODE	0xf	2	0	0	0	0	2	Details

Information last updated on 21.Jul.2008 14:38:49

The **Cluster** sub-section.

Note: Network licenses can only be used if the CodeMeter Runtime Server is started as a network server.

Item	Remarks
Product Code	Displays the Product Code
Name	Displays the name of the Product Item, normally the name of the product
Feature Map	Displays the Feature Map. The Feature Map is used to control the WAVECOM software upgrade period
Licenses	Displays the total number of network licenses
User Limit	Displays the number of licenses that are currently used in the User Limit mode
No User Limit	Displays the number of licenses that are currently used in the No User Limit mode
Exclusive	Displays the number of licenses that are currently used in the Exclusive mode
Shared	Displays the number of licenses that are currently used in the Shared mode
Free	Displays the number of licenses that are currently unused
Details	Displays detailed information about network licenses in use

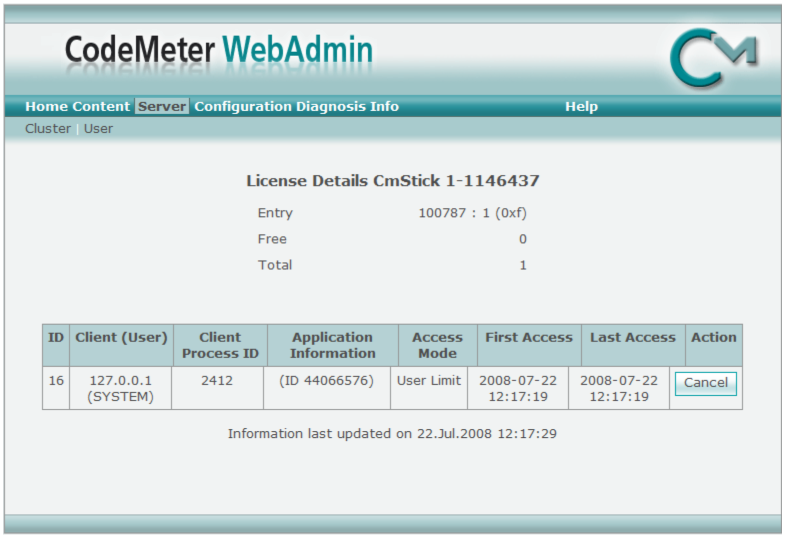


CmStick	Firm Item	Product Item	Client	Access Mode	
1-1146437	100787: WAVECOM ELEKTRONIK AG	30: Classifier	127.0.0.1	User Limit	Details

Information last updated on 21.Jul.2008 16:37:59

The **User** sub-section.

Server User	Remarks
CmStick	Displays the serial number of the connected CmStick
Firm Item	Displays information about the Firm Item (Firm Code and text)
Product Code	Displays the Product Code
Client	Displays the IP address of the current CodeMeter client
Access Mode	Displays the current Access Mode



License Details CmStick 1-1146437

Entry	100787 : 1 (0xf)
Free	0
Total	1

ID	Client (User)	Client Process ID	Application Information	Access Mode	First Access	Last Access	Action
16	127.0.0.1 (SYSTEM)	2412	(ID 44066576)	User Limit	2008-07-22 12:17:19	2008-07-22 12:17:19	Cancel

Information last updated on 22.Jul.2008 12:17:29

This screen displays detailed information about the network licenses in use. The upper area displays general information about the selected network licenses.

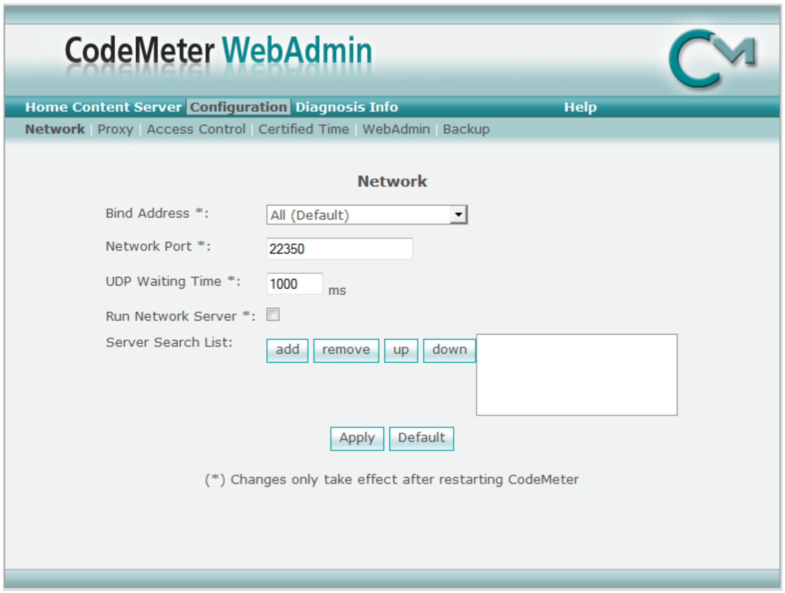
License Details	Remarks
ID	Displays the current CodeMeter process ID on the network server
Client (User)	Displays the IP address of the connected CodeMeter clients
Client Process ID	Displays the current CodeMeter process ID on the client PC
Application Information	Displays application specific information
Access Mode	Displays the current Access Mode
First Access	Displays the date and time stamp of the first access (client to server)
Last Access	Displays the date and time stamp of the last access (client to server)
Action	By clicking the "Cancel" button the selected license will be freed

Important: If a network license that is currently in use is canceled, the associated client application will experience an error. Please, ensure that the selected license is not in use.

CodeMeter WebAdmin Configuration

The **Configuration** page is split into six sub-sections:

- **Network**
- **Proxy**
- **Access Control**
- **Certified Time**
- **WebAdmin**
- **Backup**


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info
Help
Network | Proxy | Access Control | Certified Time | WebAdmin | Backup
add
remove
up
down
Apply
Default

Network

Bind Address *: All (Default)

Network Port *: 22350

UDP Waiting Time *: 1000 ms

Run Network Server *:

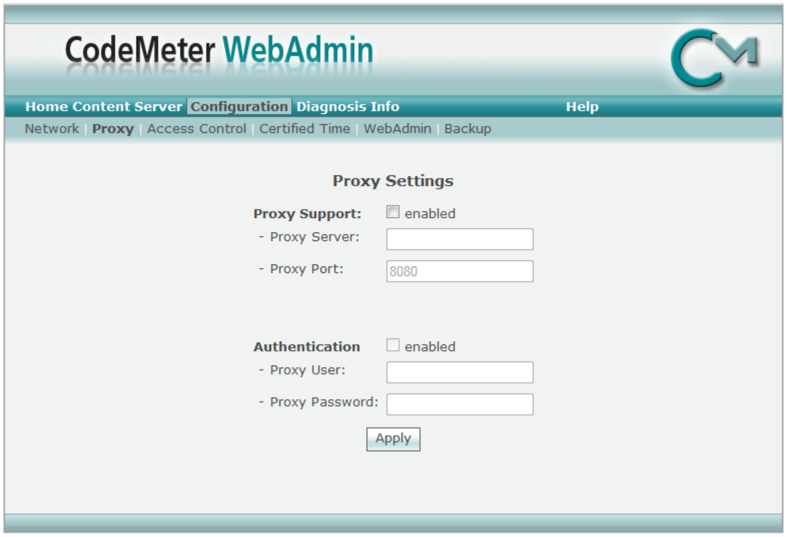
Server Search List:

(*) Changes only take effect after restarting CodeMeter

The **Network** sub-section.

License Details	Remarks
Bind Address	Select which network adapter (virtual adapter) the CodeMeter Runtime Server will be bound to. This is very useful if your PC has several network adapters and it should act as a network license server. By default, the CodeMeter Runtime Server uses the first detected network adapter (NIC)
Network Port	Port 22350 is the standard port for CodeMeter communication. You may edit this value if required. If changing port number make sure that all CodeMeter Runtime Servers use the same port if you want to use an application via the network
UDP Waiting Time	Specifies the maximum UDP wait time. This time specifies the maximum time interval within which UDP requests (requests via the UDP protocol) must be answered
Run Net-	If this option is enabled, the specified PC may be used as a CodeMeter Network Server

work Server	
Server Search List	On the Server Search List you can define the access to specific CodeMeter Servers. In this case the client only searches for these servers. To do so, just set the DNS name or the IP address of the CodeMeter Server in the Server Search List **If the CodeMeter Server is located in a different subnet, it is strongly recommended that you use the IP address of the CodeMeter Server in the Server Search List**


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
Network | Proxy | Access Control | Certified Time | WebAdmin | Backup

Proxy Settings

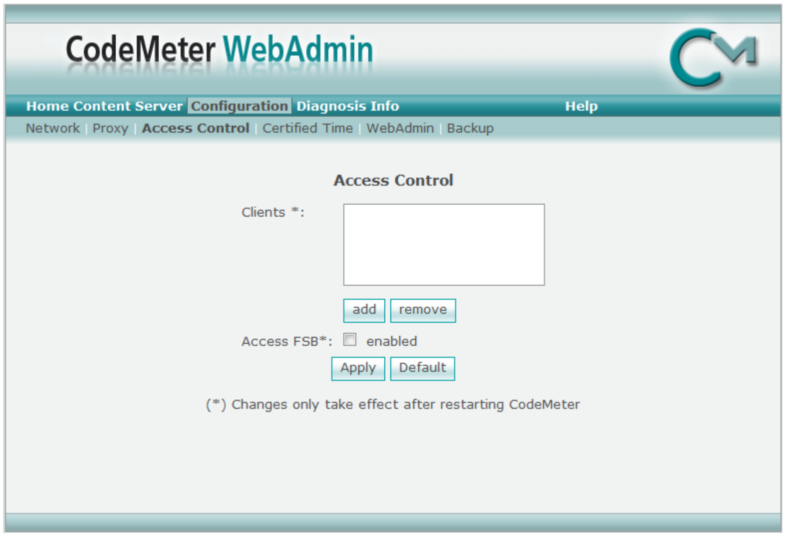
Proxy Support: enabled
- Proxy Server:
- Proxy Port: 8080

Authentication enabled
- Proxy User:
- Proxy Password:

Apply

The **Proxy** sub-section.

Configuration Proxy	**Remarks**
Proxy support	Enable or disable Proxy Server support; this option has to be enabled if you are using a proxy server
Proxy Server	Set the IP address of your proxy server
Proxy Port	Set the port number of your proxy server
Authentication	Enable or disable the proxy server authentication
Proxy User	User ID for the proxy server
Proxy Password	User password for the proxy server


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
Network | Proxy | Access Control | Certified Time | WebAdmin | Backup

Access Control

Clients *:

add remove

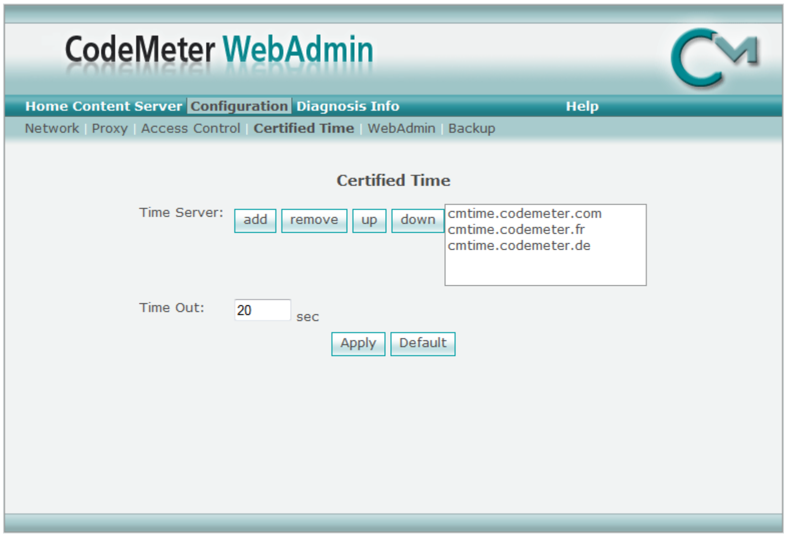
Access FSB*: enabled

Apply Default

(*) Changes only take effect after restarting CodeMeter

The **Access Control** sub-section.

Configuration Access Control	Remarks
Clients	Shows a list of all CodeMeter client computers that have permission to use the CodeMeter Server (check out a license) If this list is empty (default setting), all CodeMeter Clients are allowed to use the CodeMeter Server.
Add, Remove	To allow a special CodeMeter Client to use the CodeMeter Server, just click on the "Add" button and enter the IP address or the DNS name of the client into the input box. To remove a Client, highlight its IP address and click "Remove".
Access FS	If this feature is enabled (box is checked), CodeMeter Clients or developer PCs can access the CodeMeter Firm Security Box (FSB). To disable this feature, un-check the check-box This feature is only useful for CodeMeter Licensors


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
Network | Proxy | Access Control | Certified Time | WebAdmin | Backup
add
remove
up
down
Apply
Default

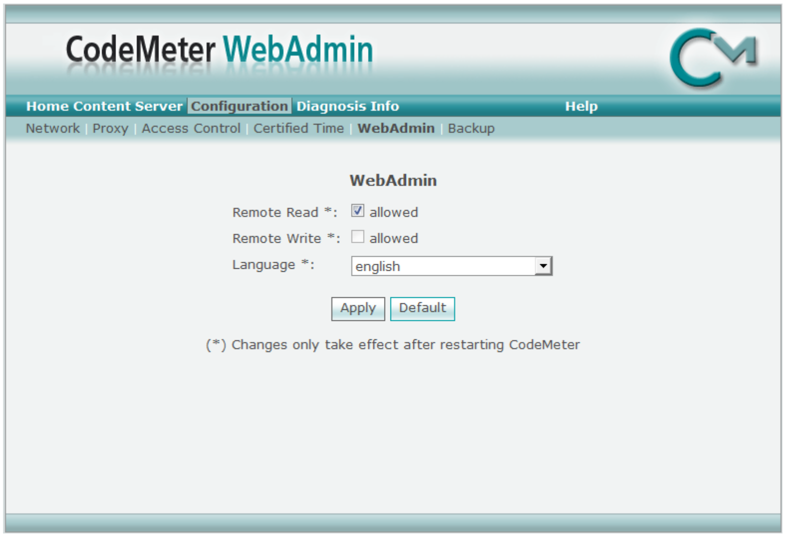
Certified Time

Time Server: cmtime.codemeter.com, cmtime.codemeter.fr, cmtime.codemeter.de

Time Out: 20 sec

The **Certified Time** sub-section.

Configuration Certified Time	Remarks
Time Server	Specify the access order or add/remove the IP address or the URL of a CodeMeter Time Server using the "add", "remove", "up" and "down" buttons
Time Out	Set the time out value of the CodeMeter Time Servers


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
Network | Proxy | Access Control | Certified Time | WebAdmin | Backup
Apply
Default

WebAdmin

Remote Read *: [✓] allowed

Remote Write *: [ ] allowed

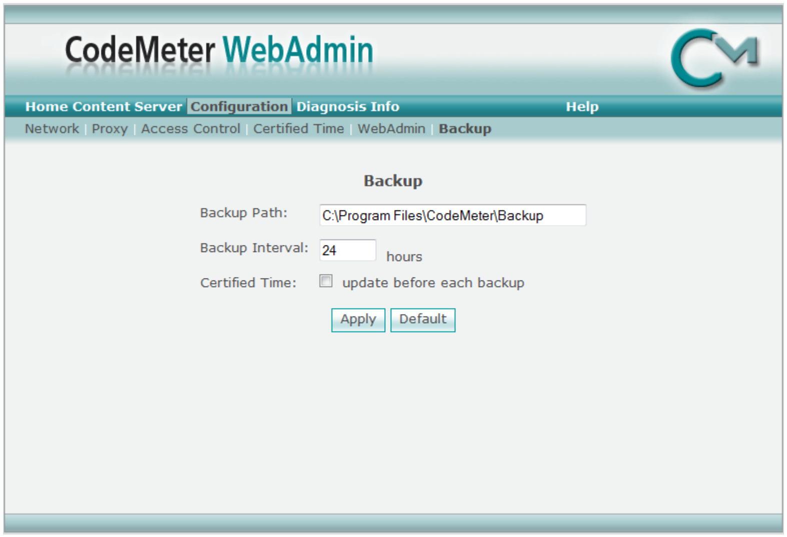
Language *: english

(*) Changes only take effect after restarting CodeMeter

The **WebAdmin** sub-section.

Config. WebAdmin	Remarks
Remote Read	Check this box to allow remote reading of your CodeMeter Runtime Server over the network
Remote Write	Check this box to allow a remote access to change the configuration settings of

	your CodeMeter Runtime Server
Language	Specify the language used in your CodeMeter WebAdministrator


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info Help
Network | Proxy | Access Control | Certified Time | WebAdmin | Backup
Backup
Apply
Default

Backup Path: C:\Program Files\CodeMeter\Backup
Backup Interval: 24 hours
Certified Time: ☐ update before each backup

The **Backup** sub-section.

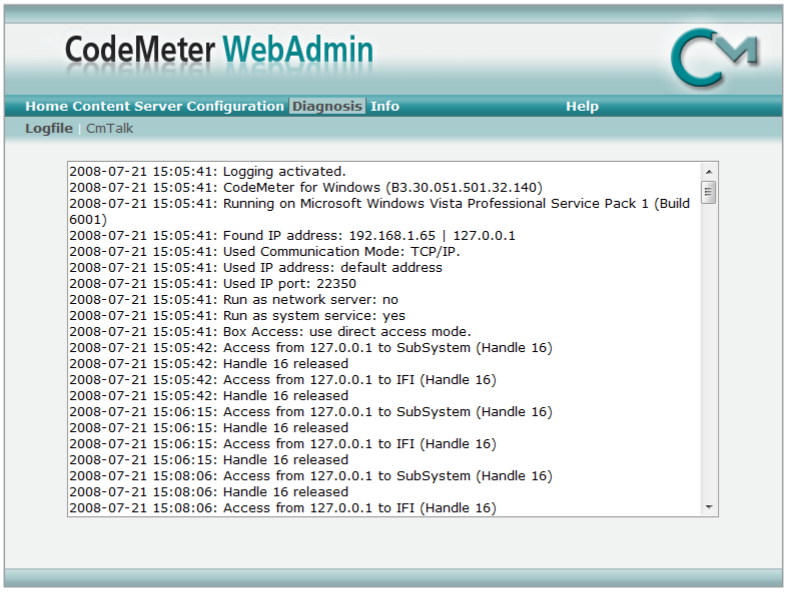
Config. Backup	Remarks
Backup Path	Specify the Backup Path to which the CmStick backup is written The default Backup Path under Windows is [\CodeMeter\Runtime\bin]
Backup Interval	Specify the interval (in hours) between CodeMeter automatic backups. The default value is 24 hours
Certified Time	Check to enable a Certified Time update before each backup

Note: Manual backups can be done immediately - instructions can be found on the **Backup** page.

CodeMeter WebAdmin Diagnosis

The **Contents** page is split into two sub-sections:

- **Logfile**
- **CmTalk**

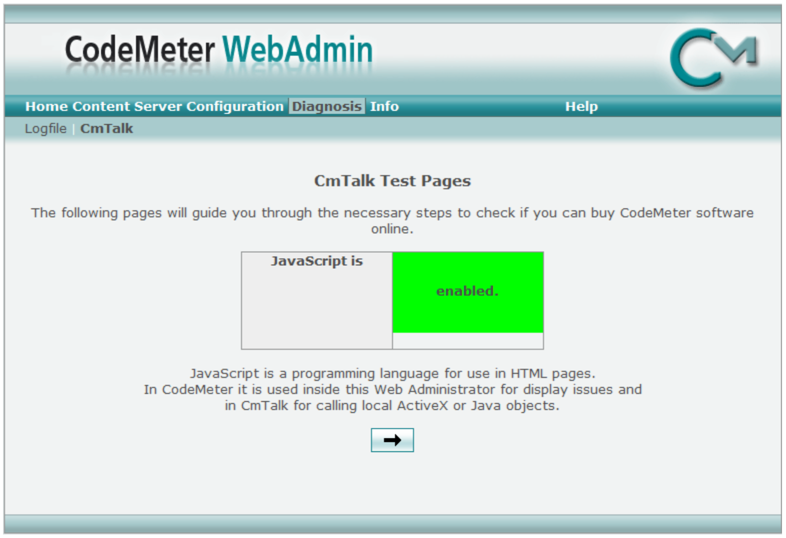


```
2008-07-21 15:05:41: Logging activated.
2008-07-21 15:05:41: CodeMeter for Windows (B3.30.051.501.32.140)
2008-07-21 15:05:41: Running on Microsoft Windows Vista Professional Service Pack 1 (Build
6001)
2008-07-21 15:05:41: Found IP address: 192.168.1.65 | 127.0.0.1
2008-07-21 15:05:41: Used Communication Mode: TCP/IP.
2008-07-21 15:05:41: Used IP address: default address
2008-07-21 15:05:41: Used IP port: 22350
2008-07-21 15:05:41: Run as network server: no
2008-07-21 15:05:41: Run as system service: yes
2008-07-21 15:05:41: Box Access: use direct access mode.
2008-07-21 15:05:42: Access from 127.0.0.1 to SubSystem (Handle 16)
2008-07-21 15:05:42: Handle 16 released
2008-07-21 15:05:42: Access from 127.0.0.1 to IFI (Handle 16)
2008-07-21 15:05:42: Handle 16 released
2008-07-21 15:06:15: Access from 127.0.0.1 to SubSystem (Handle 16)
2008-07-21 15:06:15: Handle 16 released
2008-07-21 15:06:15: Access from 127.0.0.1 to IFI (Handle 16)
2008-07-21 15:06:15: Handle 16 released
2008-07-21 15:08:06: Access from 127.0.0.1 to SubSystem (Handle 16)
2008-07-21 15:08:06: Handle 16 released
2008-07-21 15:08:06: Access from 127.0.0.1 to IFI (Handle 16)
```

The **Logfile** sub-section.

With the enabled option you can activate or deactivate the logging function.

If the logging function is activated, the **Logfile** page displays the contents of the current log file in the CodeMeter installation folder (default: [..\ProgramFiles\CodeMeter\Logs]).



CmTalk Test Pages

The following pages will guide you through the necessary steps to check if you can buy CodeMeter software online.

JavaScript is	enabled.

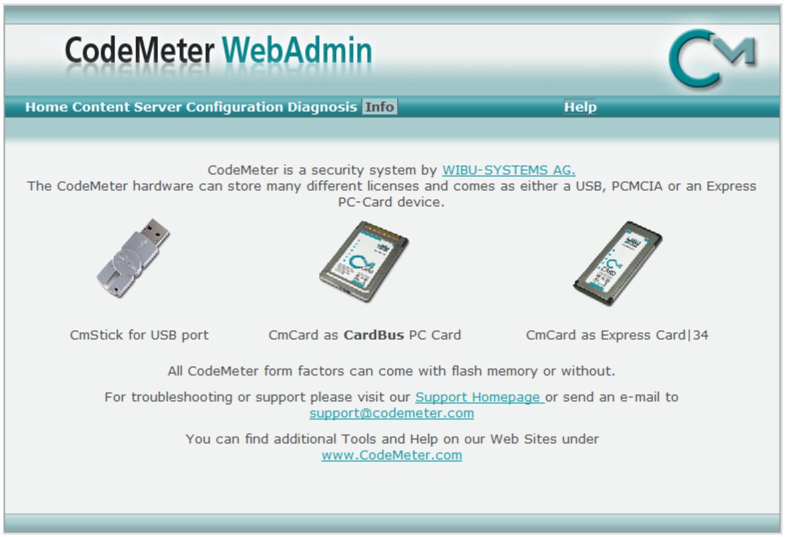
JavaScript is a programming language for use in HTML pages.
In CodeMeter it is used inside this Web Administrator for display issues and in CmTalk for calling local ActiveX or Java objects.

The **CmTalk** sub-section.

This page provides the possibility to test the CmTalk environment.

The CmTalk protocol is required for the automatic transfer or update of CodeMeter licenses via the Internet.

CodeMeter WebAdmin Info


CodeMeter WebAdmin
Home Content Server Configuration Diagnosis Info
Help
CmStick for USB port
CmCard as CardBus PC Card
CmCard as Express Card|34

CodeMeter is a security system by WIBU-SYSTEMS AG.
The CodeMeter hardware can store many different licenses and comes as either a USB, PCMCIA or an Express PC-Card device.

All CodeMeter form factors can come with flash memory or without.

For troubleshooting or support please visit our Support Homepage or send an e-mail to support@codemeter.com

You can find additional Tools and Help on our Web Sites under www.CodeMeter.com

General information regarding the CodeMeter system.

Glossary of Terms

ANSI

An acronym for the American National Standards Institute, an organization that sets standards for a variety of programming languages and systems.

ASCII

An acronym for American Standard Code for Information Interchange, pronounced "ASK-ee." It is a code in which the numbers from 0 to 127 stand for letters, numbers, punctuation marks and other characters. ASCII code is standardized to facilitate transmitting text between computers or between a computer and a peripheral device.

BCH

A BCH (Bose, Ray-Chaudhuri, Hocquenghem) code is an error-correcting code. It is a multilevel, cyclic, error-correcting, variable-length digital code used to correct mutiple random error patterns.

Convolutional code

A type of channel coding that adds patterns of redundancy to the data in order to improve the signal-to-noise ratio (SNR) for more accurate decoding at the receiving end. The Viterbi algorithm is used to decode a particular type of convolutional code

CRC

CRC (Cyclical Redundancy Checking) is an error checking technique used to ensure the accuracy of transmitting digital data. The transmitted messages are divided into predetermined lengths which, used as dividends, are divided by a fixed divisor. The remainder of the calculation is appended onto and sent with the message. At the receiving end, the computer recalculates the remainder. If it does not match the transmitted remainder, an error is detected.

DEFLATE

Deflate is a lossless data compression algorithm that uses a combination of the LZ77 algorithm and Huffman coding.

HDLC

HDLC (High-level Data Link Control) is a group of protocols for transmitting synchronous data packets between point-to-point nodes. In HDLC, data is organized into a frame. HDLC uses zero insertion/deletion process (bit stuffing) to ensure that the bit pattern of the delimiter flag does not occur in the fields between flags.

HEX

In mathematics and computer science, hexadecimal, or simply hex, is a numeral system with a radix or base of 16 usually written using the symbols 0–9 and A–F or a–f.

Huffman coding

Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. It was developed by David A. Huffman

LZ77

LZ77 is the name for a lossless data compression algorithms published in papers by Abraham Lempel and Jacob Ziv. LZ77 algorithms achieve compression by replacing portions of the data with references to matching data that has already passed through both encoder and decoder.

Matlab

MATLAB is a numerical computing environment and programming language. Created by The MathWorks, MATLAB allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages. Although it specializes in numerical computing, an optional toolbox interfaces with the Maple symbolic engine, allowing it to be part of a full computer algebra system.

Index