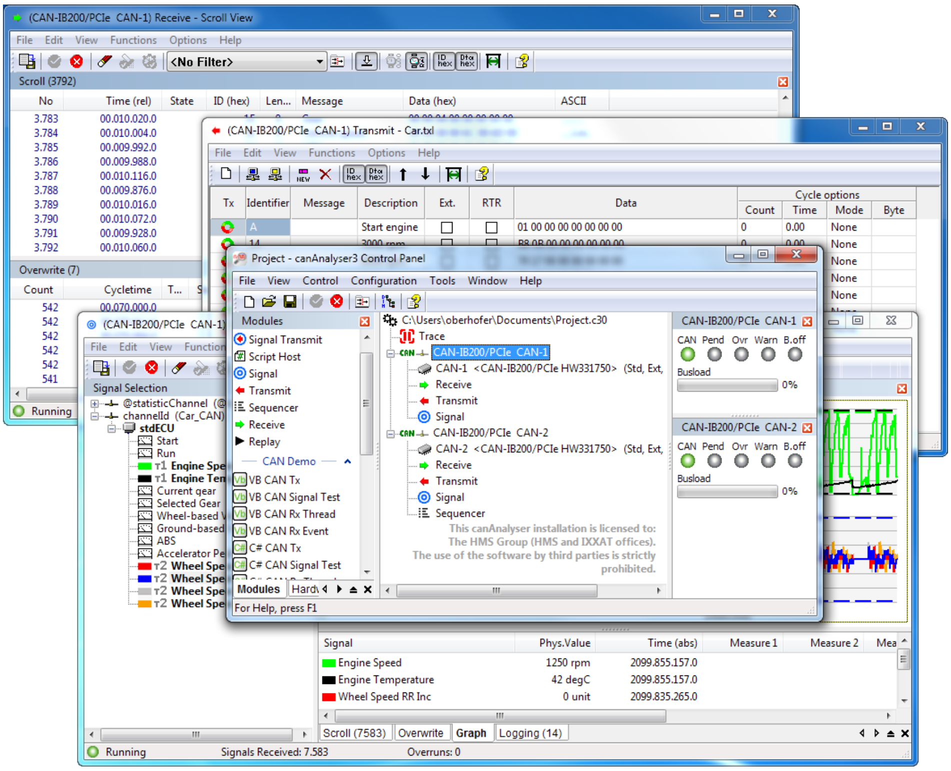
canAnalyser

Version 3



No	Time (rel)	State	ID (hex)	Len...	Message	Data (hex)	ASCII
3.783	00.010.020.0						
3.784	00.010.004.0						
3.785	00.009.992.0						
3.786	00.009.988.0						
3.787	00.010.116.0						
3.788	00.009.876.0						
3.789	00.010.016.0						
3.790	00.010.072.0						
3.791	00.009.928.0						
3.792	00.010.060.0						

Count	Cycletime	T...
542	00.070.000.0	
542		
542		
542		
542		
541		

Tx	Identifier	Message	Description	Ext.	RTR	Data	Cycle options Count	 Time	 Mode	 Byte
	A		Start engine	□	□	01 00 00 00 00 00 00 00	0	0.00	None	
	14		3000 rpm	□	□	B8 0B 00 00 00 00 00 00	0	0.00	None	
									None	
									None	
									None	

This canAnalyser installation is licensed to:
The HMS Group (HMS and IXXAT offices).
The use of the software by third parties is strictly prohibited.

Signal	Phys.Value	Time (abs)	Measure 1	Measure 2	Mea
🟩 Engine Speed	1250 rpm	2099.855.157.0			
■ Engine Temperature	42 degC	2099.855.157.0			
🟥 Wheel Speed RR Inc	0 unit	2099.835.265.0			

Signals Received: 7.583

Overruns: 0

HMS Technology Center Ravensburg GmbH
Helmut-Vetter-Straße 2
88213 Ravensburg
Germany

Tel.: +49 751 56146-0
Fax: +49 751 56146-29
Internet: www.hms-networks.de
E-Mail: info-ravensburg@hms-networks.de

Support

In case of unsolvable problems with this product or other HMS products please contact HMS in written form:

Fax: +49 751 56146-29
E-Mail: support@ixxat.de

Further international support contacts can be found on our webpage www.hms-networks.de

Copyright

Registered trademarks

Document number: 1.02.0133.30000
Version: E-3.01

Contents

Chapter 1

Overview

1.1 Area of application

The canAnalyser is a modern, powerful tool for the development, operation, maintenance and testing of CAN networks.
The canAnalyser is based on the VCI software interface of IXXAT and can be used with all hardware interfaces of IXXAT.

1.2 Functional mode

The canAnalyser is based on a modular concept: communication with the driver and the hardware is handled by a central server application, the control panel, to which several client applications, so-called analysis modules, can be connected. These analysis modules are managed by the control panel and they are supplied with the messages received by the hardware. Time-critical pre-processing, such as buffering and stamping of the telegrams with the time of reception is carried out on the hardware.
The analysis modules provide the actual analysis functionality with pre-processing and editing of the telegrams supplied by the control panel. The network is also stimulated via analysis modules, which transfer the messages to be transmitted to the server, which handles further communication with the hardware.
The advantage of this structure lies in the modularity and easy extendibility. In addition, the same analysis modules can be started more than once. With the aid of different module settings (e.g. filters), a better overview can be obtained.
The following basic functions are provided by the analysis modules:

- Online display of layer-2 messages (Receive module)
- Individual and cyclic transmission of layer-2 messages (Transmit module)
- Tracing and offline analysis of layer-2 messages (Trace module)
- Text and graphic display of interpreted messages (signals) along with statistic signals (Signal module)
- Sending of signals (SignalTransmit module)
- Time-synchronous analysis of several buses
- Display of bus load

- Emulation of nodes and protocol sequences by processing command-controlled message sequences (Sequencer module)
- Data modification and cycle time monitoring

Extended functionality could be added by creating user defined modules in a .NET compatible language. Examples in C# and VB.NET for typical scenarios are installed during setup. For further information on this topic have a look at the .NET API documentation.

1.3 Basic functions

The following section provides an introduction to the most important functions of the control panel and of the analysis modules. A more detailed explanation of the individual program modules is given in section 5 - The modules of the canAnalyser.

Configuration

The control panel is the central control of the canAnalyser and provides the following functions:

- Configuration of the hardware
- Definition of the project databases to be used. The project databases contains, along with other information, the name of the messages, the cycle time and the data length and represents the basis for the interpretation of layer-2 messages.
- Display of bus and controller status
- Creation, loading and saving of the analysis configuration

The control panel (Fig. 1.1) provides the following displays:

- Module list with available analysis modules
- Configuration tree with the current analysis configuration
- Bus status window
- Layout list window
- Error protocol window

The following information is shown in the configuration tree:

- Name of the loaded configuration
- List of virtual busses
- Assignment of the controllers with their settings
- Assignment of the analysis modules to the individual virtual busses

In order to link an analysis module from the module overview to a controller, the corresponding module is dragged with the mouse from the module overview onto the required controller using the drag and drop functionality.

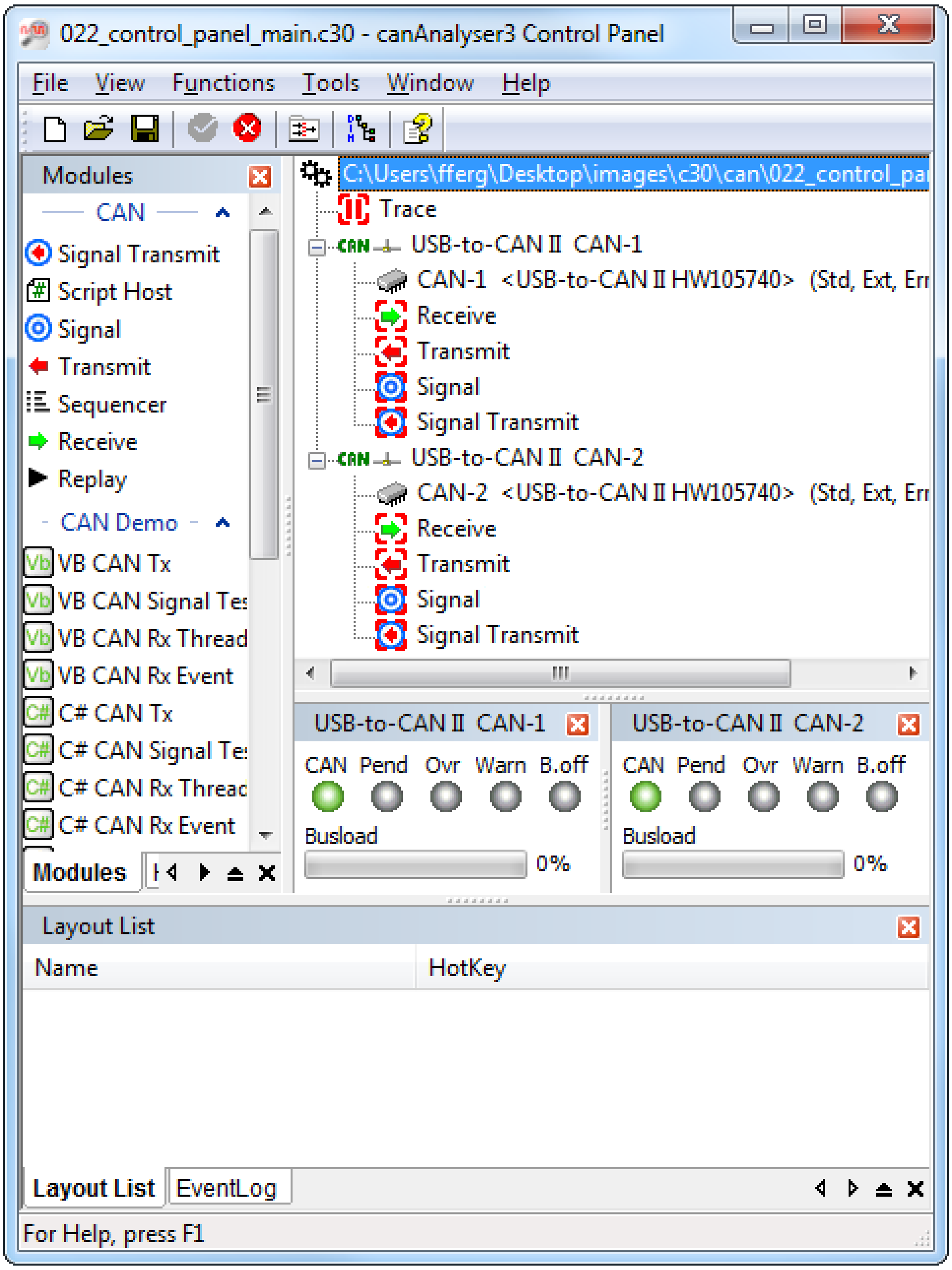
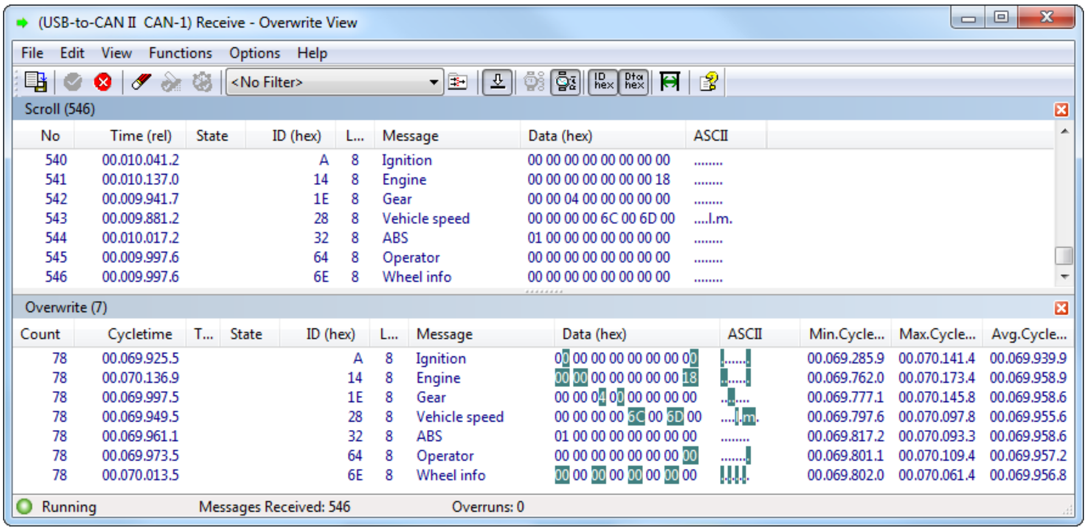


Figure 1.1: Control panel of the canAnalyser



Scroll (546)

No	Time (rel)	State	ID (hex)	L...	Message	Data (hex)	ASCII
540	00.010.041.2		A	8	Ignition	00 00 00 00 00 00 00 00	....
541	00.010.137.0		14	8	Engine	00 00 00 00 00 00 00 18	....
542	00.009.941.7		1E	8	Gear	00 00 04 00 00 00 00 00	....
543	00.009.881.2		28	8	Vehicle speed	00 00 00 00 6C 00 6D 00	....l.m.
544	00.010.017.2		32	8	ABS	01 00 00 00 00 00 00 00	....
545	00.009.997.6		64	8	Operator	00 00 00 00 00 00 00 00	....
546	00.009.997.6		6E	8	Wheel info	00 00 00 00 00 00 00 00	....

Overwrite (7)

Count	Cycletime	T...	State	ID (hex)	L...	Message	Data (hex)	ASCII	Min.Cycle...	Max.Cycle...	Avg.Cycle...
78	00.069.925.5			A	8	Ignition	00 00 00 00 00 00 00 00	....	00.069.285.9	00.070.141.4	00.069.939.9
78	00.070.136.9			14	8	Engine	00 00 00 00 00 00 00 18	....	00.069.762.0	00.070.173.4	00.069.958.9
78	00.069.997.5			1E	8	Gear	00 00 04 00 00 00 00 00	....	00.069.777.1	00.070.145.8	00.069.958.6
78	00.069.949.5			28	8	Vehicle speed	00 00 00 00 6C 00 6D 00	....l.m.	00.069.797.6	00.070.097.8	00.069.955.6
78	00.069.961.1			32	8	ABS	01 00 00 00 00 00 00 00	....	00.069.817.2	00.070.093.3	00.069.958.6
78	00.069.973.5			64	8	Operator	00 00 00 00 00 00 00 00	....	00.069.801.1	00.070.109.4	00.069.957.2
78	00.070.013.5			6E	8	Wheel info	00 00 00 00 00 00 00 00	....	00.069.802.0	00.070.061.4	00.069.956.8

Running Messages Received: 546 Overruns: 0

Figure 1.2: Receive module

Receiving messages

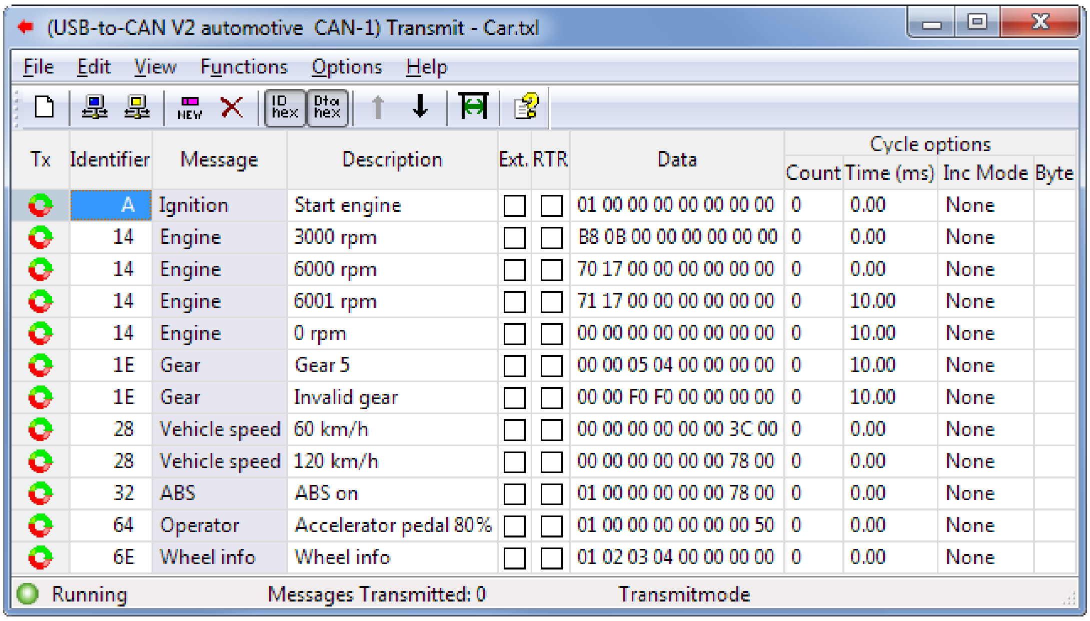
The Receive module provides the following analysis functions:

- Reception and display of layer-2 messages in the order of their time of reception (scroll mode)
- Display of the received messages sorted according to identifier (overwrite mode)
- Show and hide any messages (filter function)
- Display of modifications in the data fields of the received messages
- Monitoring of cycle time of individual messages
- Display of bus errors/error frames
- Display of data in different forms (hexadecimal, decimal, ASCII)
- Display of the total number of messages received (scroll mode) or the number per identifier (overwrite mode)

For easier identification of the messages, the name assigned to the identifier in the project databases is displayed in each analysis module. Fig. 1.2 shows a Receive module in overwrite mode.

Transmitting messages

With the Transmit module (Fig. 1.3) transmit messages can be specified and transmitted individually or cyclically. With cyclic transmission, the number of messages, their cycle time and an increment can be defined.



Tx	Identifier	Message	Description	Ext.	RTR	Data	Cycle options			
							Count	Time (ms)	Inc Mode	Byte
●	A	Ignition	Start engine	□	□	01 00 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	3000 rpm	□	□	B8 0B 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	6000 rpm	□	□	70 17 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	6001 rpm	□	□	71 17 00 00 00 00 00 00	0	10.00	None	
●	14	Engine	0 rpm	□	□	00 00 00 00 00 00 00 00	0	10.00	None	
●	1E	Gear	Gear 5	□	□	00 00 05 04 00 00 00 00	0	10.00	None	
●	1E	Gear	Invalid gear	□	□	00 00 F0 F0 00 00 00 00	0	10.00	None	
●	28	Vehicle speed	60 km/h	□	□	00 00 00 00 00 00 3C 00	0	0.00	None	
●	28	Vehicle speed	120 km/h	□	□	00 00 00 00 00 00 78 00	0	0.00	None	
●	32	ABS	ABS on	□	□	01 00 00 00 00 00 78 00	0	0.00	None	
●	64	Operator	Accelerator pedal 80%	□	□	01 00 00 00 00 00 00 50	0	0.00	None	
●	6E	Wheel info	Wheel info	□	□	01 02 03 04 00 00 00 00	0	0.00	None	

Running   Messages Transmitted: 0   Transmitmode

Figure 1.3: Transmit module

Message recording (Trace) from several buses

The Trace module (Fig. 1.4) enables parallel message recording of several buses onto the hard disk.
Message recording (Trace) is carried out for the configured bus systems in separate files. After recording, the messages are displayed in relation to one another in terms of time and color-coded.
Control and configuration of the trace and the display of the recorded messages are carried out by the Trace module.

Filtering messages

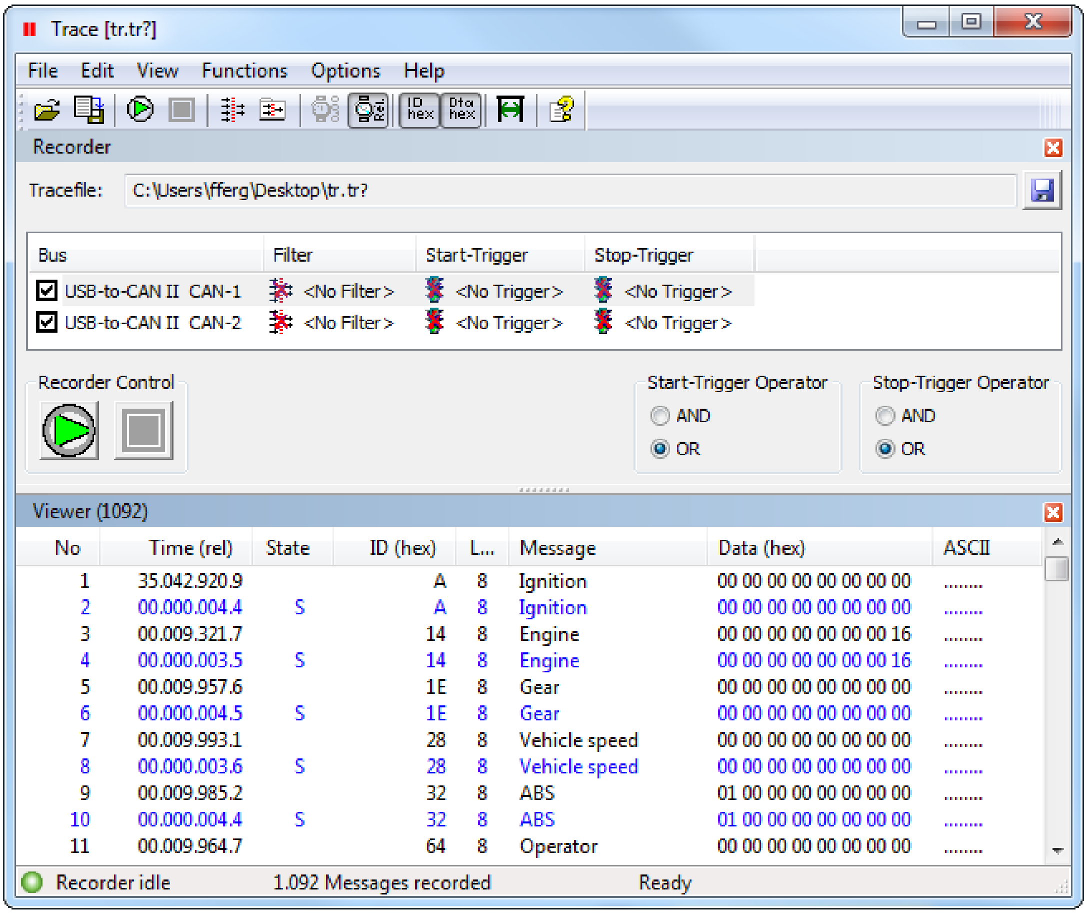
Filters are used within analysis modules to reduce the amount of incoming messages. The user configures different filters with userdefined analysis criterions to view the messages stream with specific aspects. Filters are available applicationwide and are identified by a userdefined name. Within an analysis module the user simply selects a filter by it's name to activate it or to switch between different filter configurations.

Interpretation of messages

With the Signal module it is possible to interpret the data of received layer-2 messages. Interpretation is based on databases, which must be specified when the Analysis environment is configured. Database could be created with the Database editor or with other appropriate tools. Further information is given in the manual of the Database editor.
In the Signal module, messages can also be displayed in order of the time of their reception (scroll mode) or pre-configured in overwrite mode (Fig. 1.5). In overwrite mode, the display of signal value modifications and monitoring of the cycle time are supported.

Time-synchronous analysis

The display of layer-2 messages and interpreted signals of the various analysis modules can be synchronized by means of the time stamp.



Tracefile: C:\Users\fferg\Desktop\tr.tr?

Bus	Filter	Start-Trigger	Stop-Trigger
☑ USB-to-CAN II CAN-1	<No Filter>	<No Trigger>	<No Trigger>
☑ USB-to-CAN II CAN-2	<No Filter>	<No Trigger>	<No Trigger>

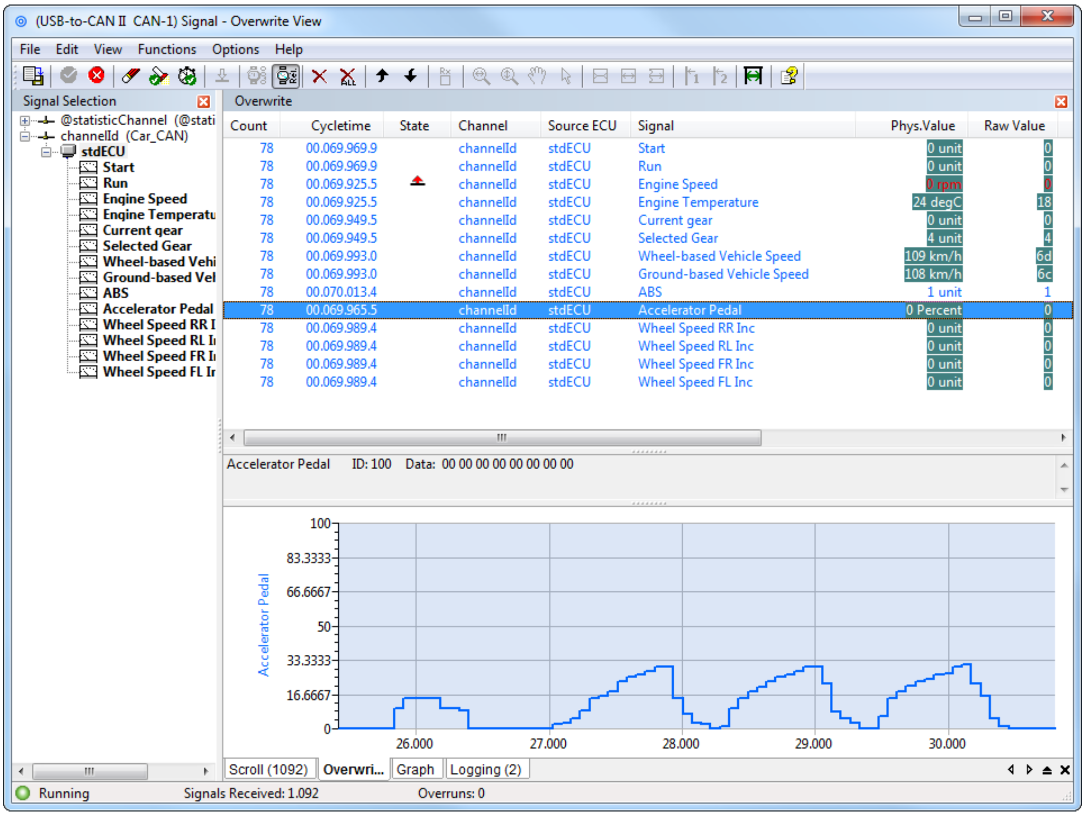
Start-Trigger Operator: ○ AND ● OR

Stop-Trigger Operator: ○ AND ● OR

No	Time (rel)	State	ID (hex)	L...	Message	Data (hex)	ASCII
1	35.042.920.9		A	8	Ignition	00 00 00 00 00 00 00 00	....
2	00.000.004.4	S	A	8	Ignition	00 00 00 00 00 00 00 00	....
3	00.009.321.7		14	8	Engine	00 00 00 00 00 00 00 16	....
4	00.000.003.5	S	14	8	Engine	00 00 00 00 00 00 00 16	....
5	00.009.957.6		1E	8	Gear	00 00 00 00 00 00 00 00	....
6	00.000.004.5	S	1E	8	Gear	00 00 00 00 00 00 00 00	....
7	00.009.993.1		28	8	Vehicle speed	00 00 00 00 00 00 00 00	....
8	00.000.003.6	S	28	8	Vehicle speed	00 00 00 00 00 00 00 00	....
9	00.009.985.2		32	8	ABS	01 00 00 00 00 00 00 00	....
10	00.000.004.4	S	32	8	ABS	01 00 00 00 00 00 00 00	....
11	00.009.964.7		64	8	Operator	00 00 00 00 00 00 00 00	....

Recorder idle 1.092 Messages recorded Ready

Figure 1.4: Trace module

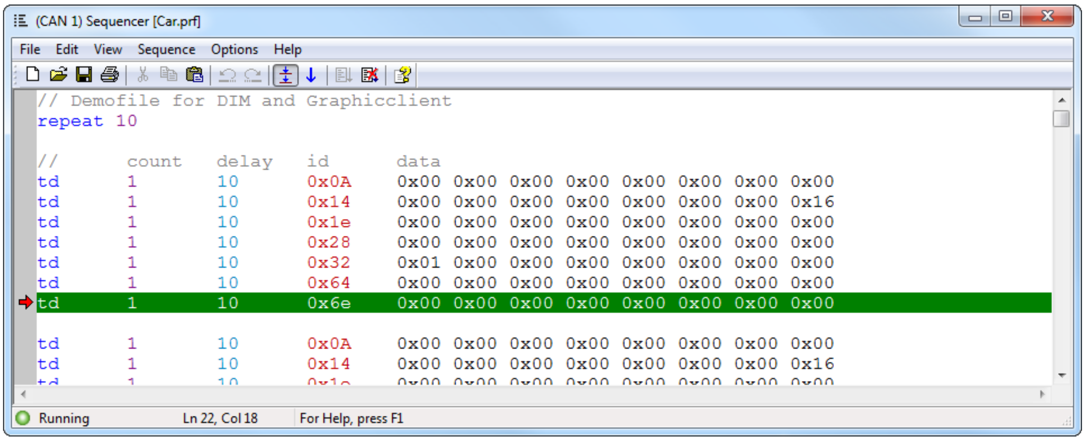


Count	Cycletime	State	Channel	Source ECU	Signal	Phys.Value	Raw Value
78	00.069.969.9		channelId	stdECU	Start	0 unit	0
78	00.069.969.9		channelId	stdECU	Run	0 unit	0
78	00.069.925.5	▲	channelId	stdECU	Engine Speed	0 rpm	0
78	00.069.925.5		channelId	stdECU	Engine Temperature	24 degC	18
78	00.069.949.5		channelId	stdECU	Current gear	0 unit	0
78	00.069.949.5		channelId	stdECU	Selected Gear	4 unit	4
78	00.069.993.0		channelId	stdECU	Wheel-based Vehicle Speed	109 km/h	6d
78	00.069.993.0		channelId	stdECU	Ground-based Vehicle Speed	108 km/h	6c
78	00.070.013.4		channelId	stdECU	ABS	1 unit	1
78	00.069.965.5		channelId	stdECU	Accelerator Pedal	0 Percent	0
78	00.069.989.4		channelId	stdECU	Wheel Speed RR Inc	0 unit	0
78	00.069.989.4		channelId	stdECU	Wheel Speed RL Inc	0 unit	0
78	00.069.989.4		channelId	stdECU	Wheel Speed FR Inc	0 unit	0
78	00.069.989.4		channelId	stdECU	Wheel Speed FL Inc	0 unit	0

Accelerator Pedal ID: 100 Data: 00 00 00 00 00 00 00 00

Running Signals Received: 1.092 Overruns: 0

Figure 1.5: Signal module, overwrite mode



```
// Demofile for DIM and Graphicclient
repeat 10

//      count   delay   id      data
td      1       10      0x0A    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x14    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x16
td      1       10      0x1e    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x28    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x32    0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x64    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x6e    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

td      1       10      0x0A    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x14    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x16
td      1       10      0x1e    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

Figure 1.6: Sequencer module

By double-clicking on a received message or on a received signal, the display of the other analysis module is screened to the nearest entry of the marked message in terms of time and this entry is marked.
Time-synchronous analysis is particularly useful when the message traffic of different bus systems with different levels of bus traffic has to be analysed and when correlations have to be set up between different bus systems.

Display of the bus load

For a quick overview over the busload and error flags of the busses you can use the bus state dialogs available in the control panel. A graphical analysis of the busload could be done in the signal module by utilizing the appropriate statistic signals.

Emulation of nodes or protocols by running sequences

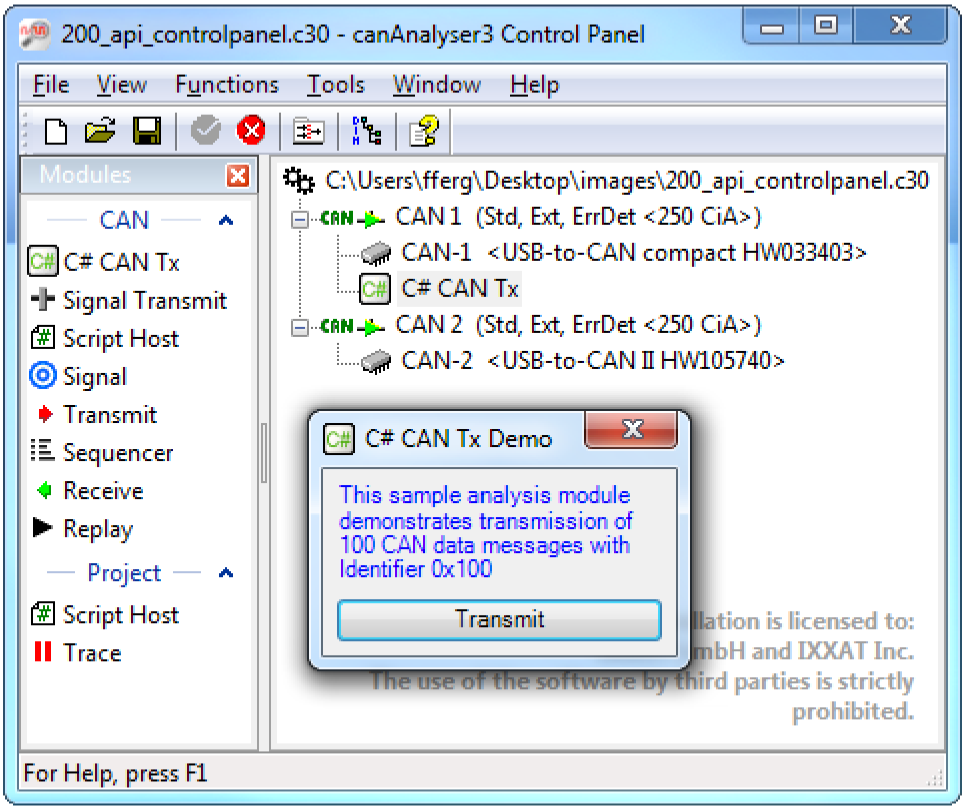
The Sequencer module (Fig. 1.6) provides processing of command-controlled message sequences and can be used to emulate nodes or protocol sequences or to generate a certain bus load.
In addition, the Sequencer module enables a so-called trace replay. A trace file recorded by the Trace module can be converted to a message sequence and processed.

Graphic display of data contents

The Signal module supports the display of signals in the form of y-t diagrams (line writers).

Integrating own analysis modules

Via the open .NET programming interface the user has the possibility to extend the canAnalyser by own modules and user interfaces. Own, autonomous, on .NET Framework based modules can be written by using common Windows development environments (e.g. Visual Studio .NET, Delphi) and can then be integrated to the canAnalyser. Consequently it's possible to create user interfaces for own systems respectively for devices and tools with system specific analysis functions.
.NET modules must have a specific layout (see examples and .NET API documentation) and will be searched for in the following directories on start of the canAnalyser:


200_api_controlpanel.c30 - canAnalyser3 Control Panel
File View Functions Tools Window Help
Modules
CAN
C# CAN Tx
Signal Transmit
Script Host
Signal
Transmit
Sequencer
Receive
Replay
Project
Script Host
Trace
C:\Users\fferg\Desktop\images\200_api_controlpanel.c30
CAN 1 (Std, Ext, ErrDet <250 CiA>)
CAN-1 <USB-to-CAN compact HW033403>
C# CAN Tx
CAN 2 (Std, Ext, ErrDet <250 CiA>)
CAN-2 <USB-to-CAN II HW105740>
C# CAN Tx Demo
Transmit
For Help, press F1

This sample analysis module demonstrates transmission of 100 CAN data messages with Identifier 0x100

…llation is licensed to: …mbH and IXXAT Inc. The use of the software by third parties is strictly prohibited.

Figure 1.7: Control Panel with "C# CAN Tx" sample analysis module

1. In the installation folder
   (e.g. `c:\Program Files (x86)\IXXAT\canAnalyser3 standard`)

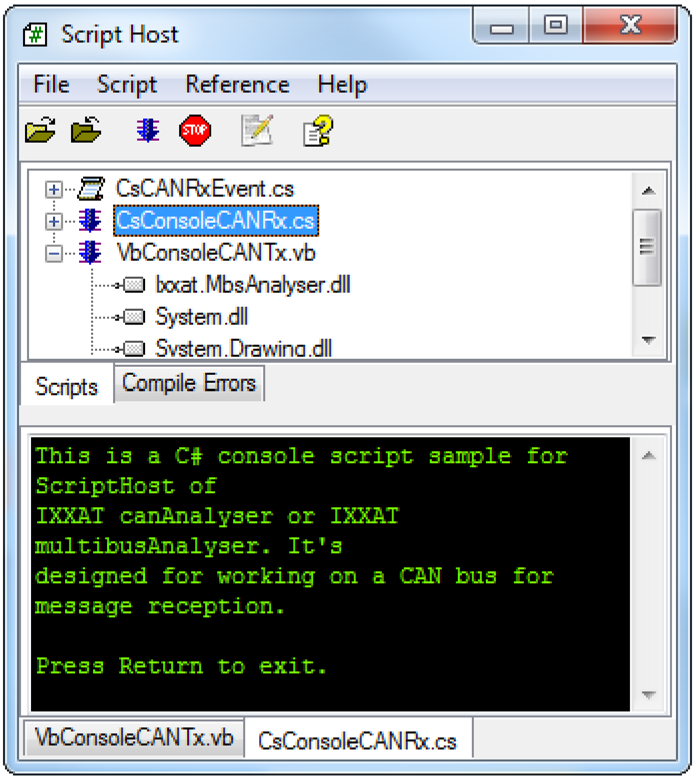
2. In %UserDocs%\IXXAT\canAnalyser\3.0\API\UDModules
   (e.g. `c:\Users\John\Documents\IXXAT\canAnalyser\3.0\API\UDModules`)

3. In %PublicDocs%\IXXAT\canAnalyser\3.0\API\UDModules
   (e.g. `C:\Users\Public\Documents\IXXAT\canAnalyser\3.0\API\UDModules`)

An analysis module is provided the canAnalyser in the form of an assembly. User defined modules that are automatically detected at application startup are displayed beside the standard modules within the Modules window of the Control Panel and can be started via Drag-and-Drop (Fig. 1.7).

Executing scripts

Because creation and modification of scripts is very flexible and cost effective they ease off the work of developers during the testing phase as well as searching errors by service engineers on-site. At this it's not mandatory having an installed development environment and each modification can be tested immediately.
For configuring and executing scripts the Control Panel provides a Script Host as analysis module within the Modules window (Fig. 1.8). In here executable scripts are based on the same .NET programming interface as used for integration of own analysis modules. The Script Host supports console based scripts as well as scripts with graphical user interface (GUI).



```
This is a C# console script sample for
ScriptHost of
IXXAT canAnalyser or IXXAT
multibusAnalyser. It's
designed for working on a CAN bus for
message reception.

Press Return to exit.
```

Figure 1.8: Script Host with programming samples

1.4 The term analysis configuration

An analysis configuration denotes the bulk of Control Panel configuration data and the settings of all herein configured analysis modules. That are the hardware parameters as well as filter settings and module specific settings like the list of send messages within the Transmit module. Furthermore an analysis configuration includes layout informations like position and size of all canAnalyser windows.

Via the Control Panel the complete analysis configuration can be centrally saved to a configuration file form where it can be restored later.

To adopt solely parts of an analysis configuration (e.g. the list of send messages of a Transmit module or the filer settings) to a second analysis configuration the canAnalyser provides the feature to **export** such settings to a separate file. The exported settings can be restored at the same place into another analysis configuration by **importing** that file.

Chapter 2

canAnalyser lite/standard

Two versions of the canAnalyser are available:

- canAnalyser lite
- canAnalyser standard

The available functionality is determined by license entries in CodeMeter sticks. If no license entries are accessible the access to CAN devices is blocked. In this so called demo mode the only supported device is an emulated CAN device (DemoAdapter), which is sufficient to test the functionality of the canAnalyser.
The canAnalyser lite differs from the canAnalyser standard in that it has a restricted scope of functions. The differences between the two versions are described in the following table:

Module	lite	standard
Multi-channel capability	Only one channel can be analysed	Simultaneous analysis of multiple channels possible.
Analysis modules	Each analysis module can only be opened once	Analysis modules can be opened more than once and configured differently
Delivery specification modules	Receive, Transmit, Trace, Sequencer, Replay, SignalReceive, SignalTransmit	Receive, Transmit, Trace, Sequencer, Replay, SignalRecieve
Restrictions in SignalReceive module	Supports to analyse up to 5 signals	no restrictions

Chapter 3

Installation and start-up

3.1 System requirements

canAnalyser is a 32-bit program. To operate the canAnalyser, the following system requirements must be fulfilled:

- x86 compatible processor with minimum 800 MHz or higher
- Windows XP, Windows Vista, Windows 7, Windows 8.x
- At least 1 GB RAM
- Installed IXXAT VCI-driver, version 3.5 or higher

Before installing the canAnalyser, the IXXAT VCI-driver must be installed which allows to access the hardware.
To install the CAN hardware, please read the hardware installation manual.
For installation of the required IXXAT VCI-driver, please consult the VCI installation manual.

3.2 Installation

To install the canAnalyser, insert the program CD supplied in the CD drive of your computer. If the installation program is not starting automatically please run the setup file(e.g. `canAna30_XXXX.exe`) on the CD. Follow the instructions of the installation program.

3.3 Starting the canAnalyser

The canAnalyser is started by clicking on the program icon created on the desktop or via the Windows Start menu.

Chapter 4

Software protection

4.1 Overview

Since version 2.5 of canAnalyser the protection scheme has been changed from node locked registration to protection by USB hardware dongle. The software protection uses the CodeMeter stick by Wibu Systems AG. For more information about additional features for the user see the CodeMeter portal at `http://www.codemeter.com`.

4.2 Installation

The installation of the CodeMeter user runtime is integrated into the canAnalyser installation.
However, if you want to uninstall the canAnalyser you have to uninstall the CodeMeter user runtime separately.
The CodeMeter Runtime Kit can be uninstalled easily on Microsoft Windows operating systems. Just open the control panel and choose "Add/Remove Programs", select CodeMeter Runtime Installer and choose "CodeMeter Runtime Installer Remove". All driver files and the entries in the registry will be removed from the computer automatically.

4.3 Usage

The available program functionality depends on the availability of valid license entries. During operation of the canAnalyser the CodeMeter stick with the appropriate license has to be plugged into the USB port. If no valid license entry is accessible the program functionality is reduced to demo mode without hardware access.

4.4 Virtual disk

Please note that the removable disk that is created by the CodeMeter-Stick must not be used to store data on it! The displayed 2 MB Memory are only a virtual disk space, which is needed by your system to identify the CM-Stick correctly.

Chapter 5

The modules of the canAnalyser

This section describes the individual components of the canAnalyser in detail. The controls, menus and dialogs are described in each case in connection with the individual modules.

The control panel represents the central control of the canAnalyser. Via the control panel the analysis configurations can be set and the individual modules are started. The actual analysis functionality is provided by these *analysis modules*.

canAnalyser comes with a number of those modules, which are being described in the course of this section. Further modules for Layer7 message interpretation can be obtained.

As soon as a module instance is added to a Virtual Bus (see also section 5.1.6), it gets *all* received messages of that bus. Even if a module is minimized to the task bar or closed, its message reception goes on.

A common feature of all modules is the status bar. It begins with a LED icon indicating the status of the control panel resp. of the module:

LED color	Meaning
Green	Control panel and module are started
Blinking Green	Limited or no connectivity to a physical bus, controller might be disabled
Blinking Red	Control panel is stopped
Red	Module is stopped

5.1 Control panel

5.1.1 Start/Stop of the hardware

The analysis can be started via the toolbar. By clicking on the **Start Communication** button in the toolbar of the control panel, all fieldbus controllers contained in the configuration are started. To stop the hardware, click on the **Stop Communication** button. Objects can only be transmitted or received when the hardware has been started.
The controllers/hardware can also be started/stopped via the menu items **Functions | Start Communication** and **Functions | Stop Communication**.

5.1.2 Control panel main window

The program window of the control panel (Fig. 5.1) is sub-divided into the following areas:

- Module list with all available analysis modules
- Configuration tree with the current analysis configuration
- Status window for each available bus/controller
- Event log window that records internal events
- Layout window, manages named window layouts

5.1.3 Status window

The control panel provides a status window for each available controller (Fig. 5.2). By right-clicking on a controller in the configuration tree, a pop-up menu (Fig. 5.7) appears, which enables the corresponding status window to be shown or hidden.

CAN status window

The CAN status window comprises the following lights:

Meaning	Light off	Light on
CAN	CAN controller is stopped	CAN controller is started
Pend (Transmit pending)	All messages transmitted, transmit queue is empty	Messages not yet transmitted are in the hardware transmit queue
Ovr (Data overrun)	-	CAN controller overrun
Warn (Warning level)	-	CAN controller error counter in Error Warning Level
B.off (Bus off)	-	CAN-Controller in Bus off

Online analysis is only possible if controller is started.
After the first occurrence of a data overrun, the data overrun light remains activated. It only goes out when the controller is stopped and restarted.
If the Bus Off light is on, the CAN controller is in Bus Off mode, i.e. it was disconnected from the bus and therefore no longer participates in network communication. The hardware must be stopped and restarted in order to restore CAN communication.

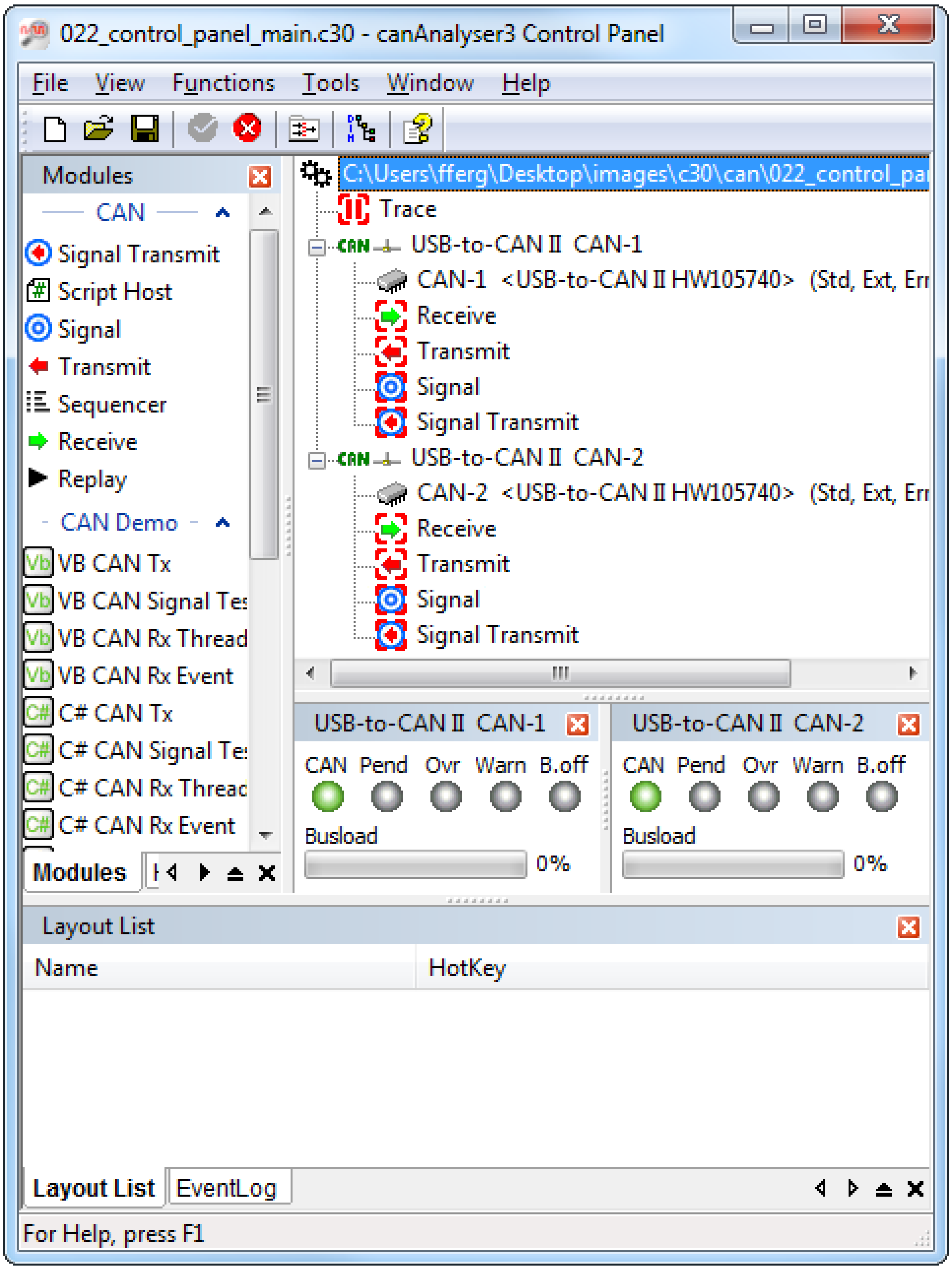
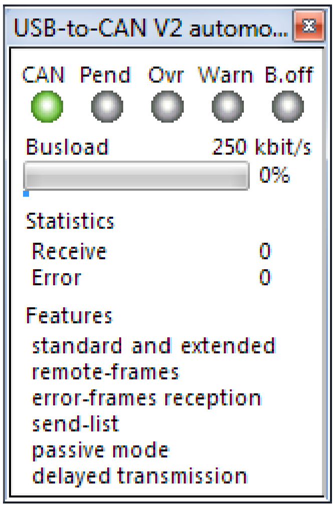


Figure 5.1: The visible fields of the control panel



USB-to-CAN V2 automo...

CAN Pend Ovr Warn B.off

Busload 250 kbit/s

0%

Statistics

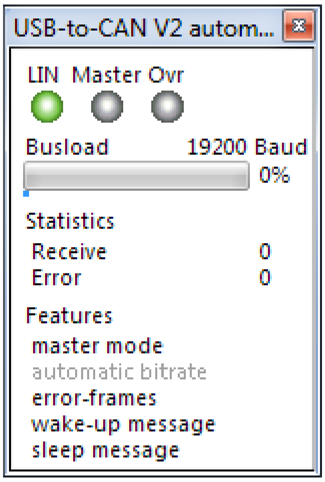
Receive 0

Error 0

Features

standard and extended
remote-frames
error-frames reception
send-list
passive mode
delayed transmission

Figure 5.2: CAN status window



USB-to-CAN V2 autom...

LIN Master Ovr

Busload 19200 Baud

0%

Statistics

Receive 0

Error 0

Features

master mode
automatic bitrate
error-frames
wake-up message
sleep message

Figure 5.3: LIN status window

The bus load display shows the load of the bus in per cent. Its measurement is a hardware function which is not supported by all interface boards, hence is only visible if this is supported by the interface.

LIN status window

The LIN status window (Fig. 5.3) comprises the following lights:

Meaning	Light off	Light on
LIN	LIN controller is stopped	LIN controller is started
Master	LIN operates in Slave mode	LIN operates in Master mode
Ovr (Data overrun)	-	LIN controller overrun

Online analysis is only possible if controller is started.
After the first occurrence of a data overrun, the data overrun light remains activated. It only goes out when the controller is stopped and restarted.
The bus load display shows the load of the bus in per cent. Its measurement is a hardware function which is not supported by all interface boards, hence it is only visible if this is supported by the interface.

5.1.4 Event Log

The control panel has its own logging facility that records internal events and errors. It can be made visible by menu command **View | Event Log** and contains the following information:

Column	Meaning
Icon	Kind of event: Success, Information, Warning, Error, or subsequent message line
Timestamp	Date and Time of the event
Sequence	Message number based on the canAnalyser session
Code	Hexadecimal errorcode
Thread	Hexadecimal thread identifier
Module	Name of canAnalyser module that reported the event
Message	Message text

The eventlog is a comma separated text file which is located in the user folder (e.g. in C:\Users\John\AppData\Local\IXXAT\canAnalyser\3.0\Log\*\canAnalyser.log)
Use **View** main menu to configure which event kinds should be shown in the Event Log window. Menu command **View | Clear Eventlog** empties the Event Log.

5.1.5 Manage window layouts

Within the Layout window you can save the current window layout and give it a name and an associated hotkey. The hotkeys can be used to switch between different window layouts.

5.1.6 Adding of modules to the configuration

Insertion of modules using drag and drop

The module list (Fig. 5.4) shows all available analysis functions and is relevant in connection with the creation of a configuration. Using the drag and drop functionality, the selected icon of an analysis function is dragged from the module list into the configuration tree and onto a bus and can be activated in this configuration as an analysis module. An analysis function can be activated any number of times for a certain configuration by repeated dragging from the module list into the configuration tree (only possible with canAnalyser standard).

Insertion of modules using the menu of the configuration tree

By hitting the **Insert** key in the configuration tree a menu (Fig. 5.5) is displayed which allows to add a module to the selected controller. The menu contains a list of all supported modules.

5.1.7 Configuration tree

In the configuration tree (Fig. 5.6) the current analysis configuration is displayed hierarchically.
An analysis configuration defines the following objects:

- Project with name of the project file/configuration file
- Bus with communication parameters
- Analysis modules

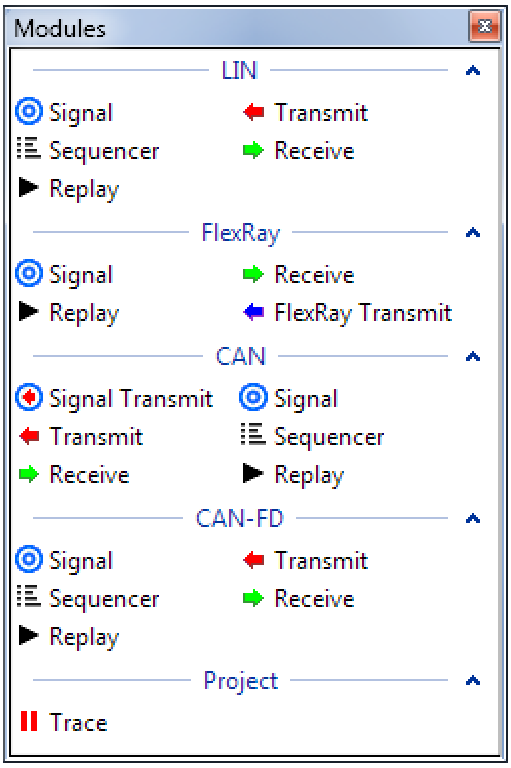


Figure 5.4: Control panel, module list

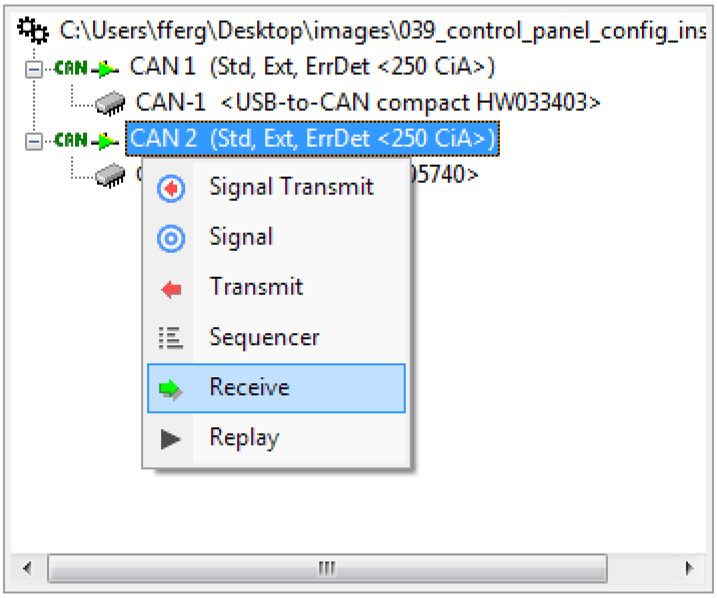
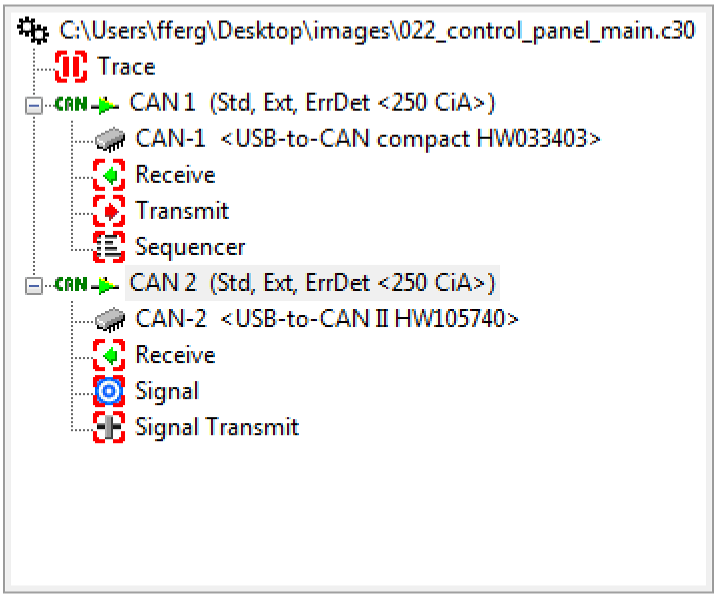

C:\Users\fferg\Desktop\images\039_control_panel_config_ins
CAN 1 (Std, Ext, ErrDet <250 CiA>)
CAN-1 <USB-to-CAN compact HW033403>
CAN 2 (Std, Ext, ErrDet <250 CiA>)
05740>
Signal Transmit
Signal
Transmit
Sequencer
Receive
Replay

Figure 5.5: Pop-up menu of the configuration tree to add modules



C:\Users\fferg\Desktop\images\022_control_panel_main.c30
- Trace
- CAN 1 (Std, Ext, ErrDet <250 CiA>)
  - CAN-1 <USB-to-CAN compact HW033403>
  - Receive
  - Transmit
  - Sequencer
- CAN 2 (Std, Ext, ErrDet <250 CiA>)
  - CAN-2 <USB-to-CAN II HW105740>
  - Receive
  - Signal
  - Signal Transmit

Figure 5.6: Configuration tree of the control panel

This section describes the settings that can be configured via the pop-up menu of the configuration tree. The complete analysis configuration can be saved and loaded.
To create an analysis configuration, the required analysis functions are dragged using drag and drop from the module list into the configuration tree onto a bus/controller. Then the elements of the configuration tree are configured. For this, each icon in the configuration tree has its own pop-up menu (Fig. 5.7), which can be activated using the right mouse button.
The following settings can be made:

- Setting the properties of the project (i.e. assigning controllers)
- Setting the properties of the bus
- Setting the properties of the controller
- User-defined designation of the analysis modules

After configuration settings are changed, an active configuration is automatically temporarily stopped and then restarted with the updated settings.

Creating a Bus

If you start up (see chapter 5.1.8) with the default configuration, or with the recent configuration, the needed buses are already at hand.
To create additional buses in the current configuration, select the item **Create Bus...** from the popup-menu of the project (Fig. 5.8) or double-click the project node. The simple create bus dialog (Fig. 5.9) will open up. Select the desired bus type from the drop-down list and click OK.
Next, you will see the Controller assignment dialog (Fig. 5.10) with the new bus appearing as "VBus" in the upper list. In the hardware list below the matching available bus controllers are presented. By clicking on a checkbox or double-clicking one of them, the Controller gets assigned to the bus.
Note the already assigned fieldbus controllers drawn in gray.
Finish the controller assignment by clicking OK, and your new bus is ready for use.

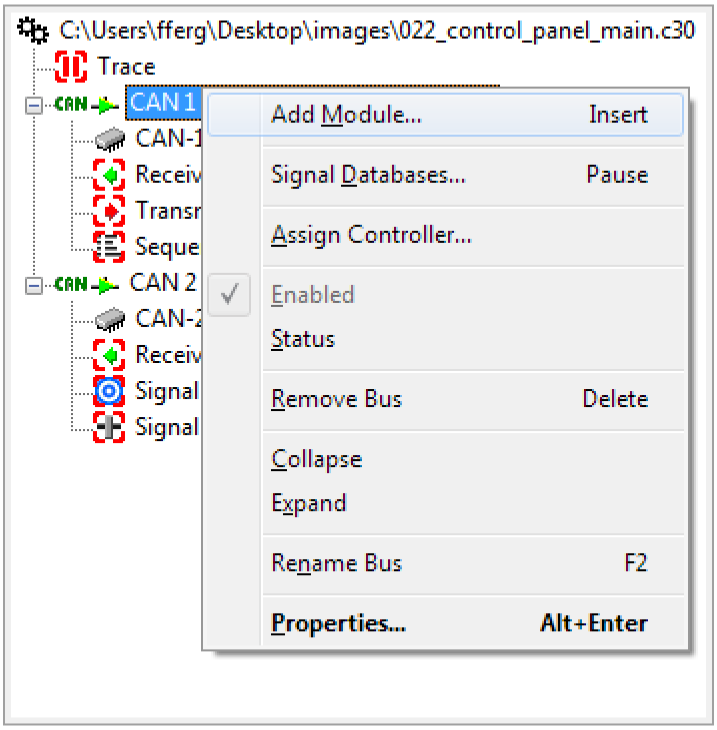


Figure 5.7: Pop-up menu of a CAN bus

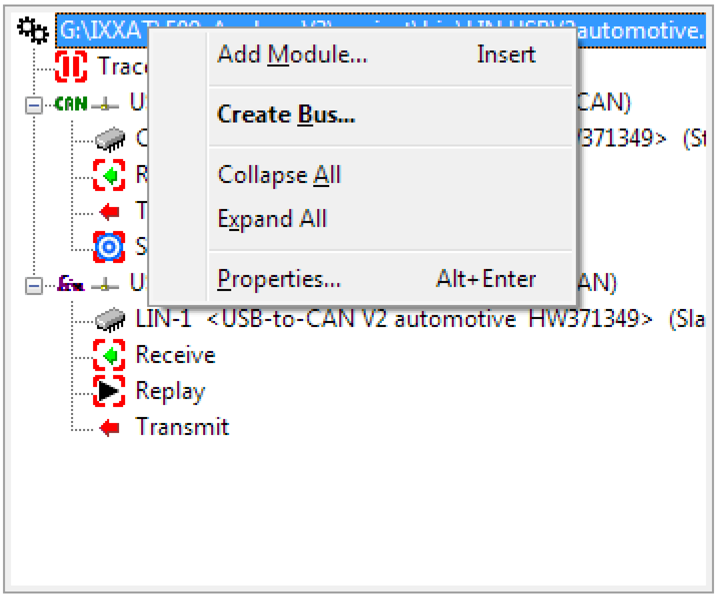
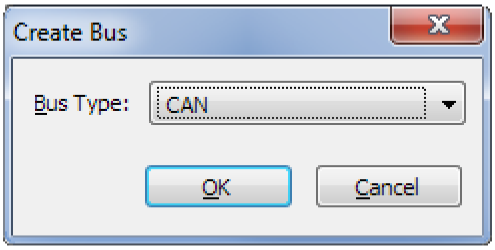
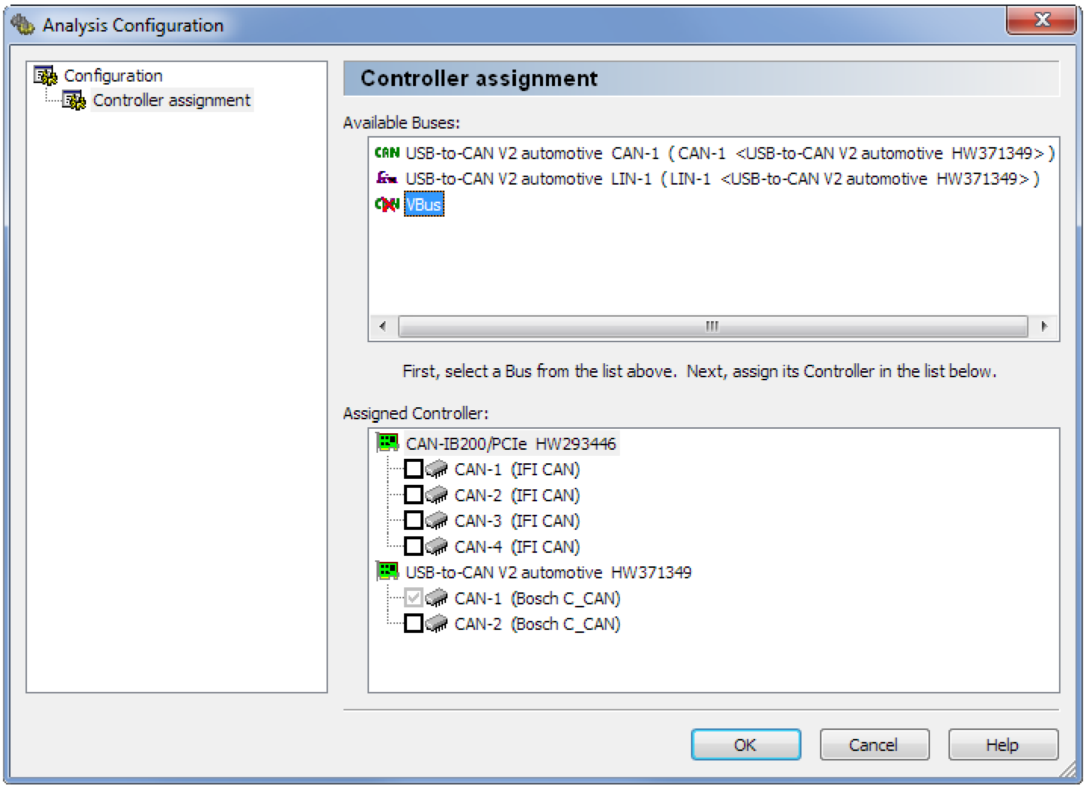


Figure 5.8: Pop-up menu of the project


Create Bus

Bus Type: CAN

Figure 5.9: Create bus dialog



Controller assignment

Available Buses:

- USB-to-CAN V2 automotive CAN-1 (CAN-1 <USB-to-CAN V2 automotive HW371349>)
- USB-to-CAN V2 automotive LIN-1 (LIN-1 <USB-to-CAN V2 automotive HW371349>)
- VBus

First, select a Bus from the list above. Next, assign its Controller in the list below.

Assigned Controller:

- CAN-IB200/PCIe HW293446
  - □ CAN-1 (IFI CAN)
  - □ CAN-2 (IFI CAN)
  - □ CAN-3 (IFI CAN)
  - □ CAN-4 (IFI CAN)
- USB-to-CAN V2 automotive HW371349
  - ☑ CAN-1 (Bosch C_CAN)
  - □ CAN-2 (Bosch C_CAN)

Figure 5.10: Controller assignment dialog

Setting a symbolic bus name

In the entry field **Name** of the branch **Bus** in the bus properties dialog (Fig. 5.11), the user can define a symbolic name for the bus, which is then displayed in the configuration tree of the control panel.

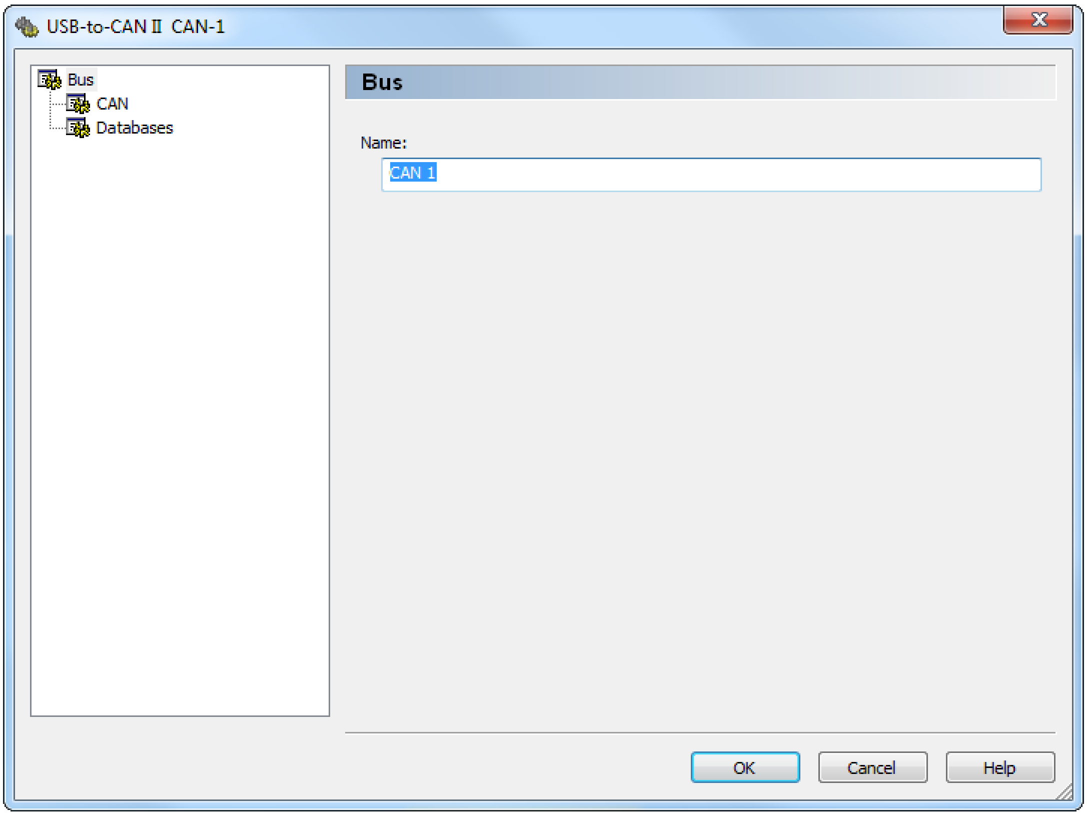
Selection of an interpretation database

With the Database editor (see database editor manual), it is possible to assign a symbolic name to individual identifiers and to interpret the data transmitted with their identifier.
If no database exists yet, the Database editor can be started directly from the menu **Tools** | **DIM Editor** of the control panel. By pressing the **Open** button in the field **Database**, a dialog box opens for selection of the database on which the bus is to be based. This database can either be in DIM(*.xml), FIBEX(*.xml) or in CANDB(*.dbc) format. You can chose the file format of the database in the file type dropdown list of the "Load database" dialog.
All analysis modules then show in their **Messages** column the symbolic name which is assigned to the individual identifier in the database. The Signal module shows the complete interpretation of a received layer-2 message based on the database.
An interpretation database is always based on a certain message format. If the message formats of the bus and database do not match, telegrams are not interpreted or are interpreted incorrectly. It must be ensured that the message formats of the bus and database match.

CAN Settings

The settings of the CAN controller are defined via branch **CAN** of the properties dialog of a CAN bus. These are:

- Message format

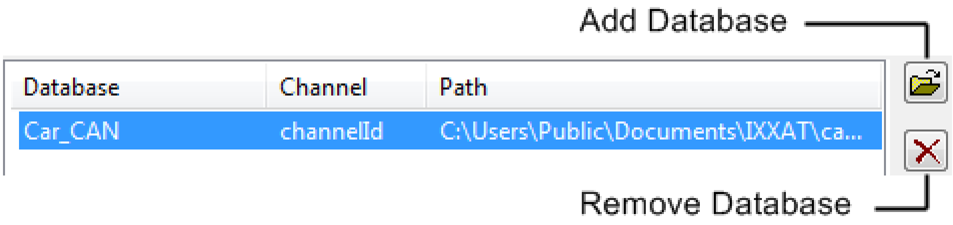


USB-to-CAN II CAN-1

Bus

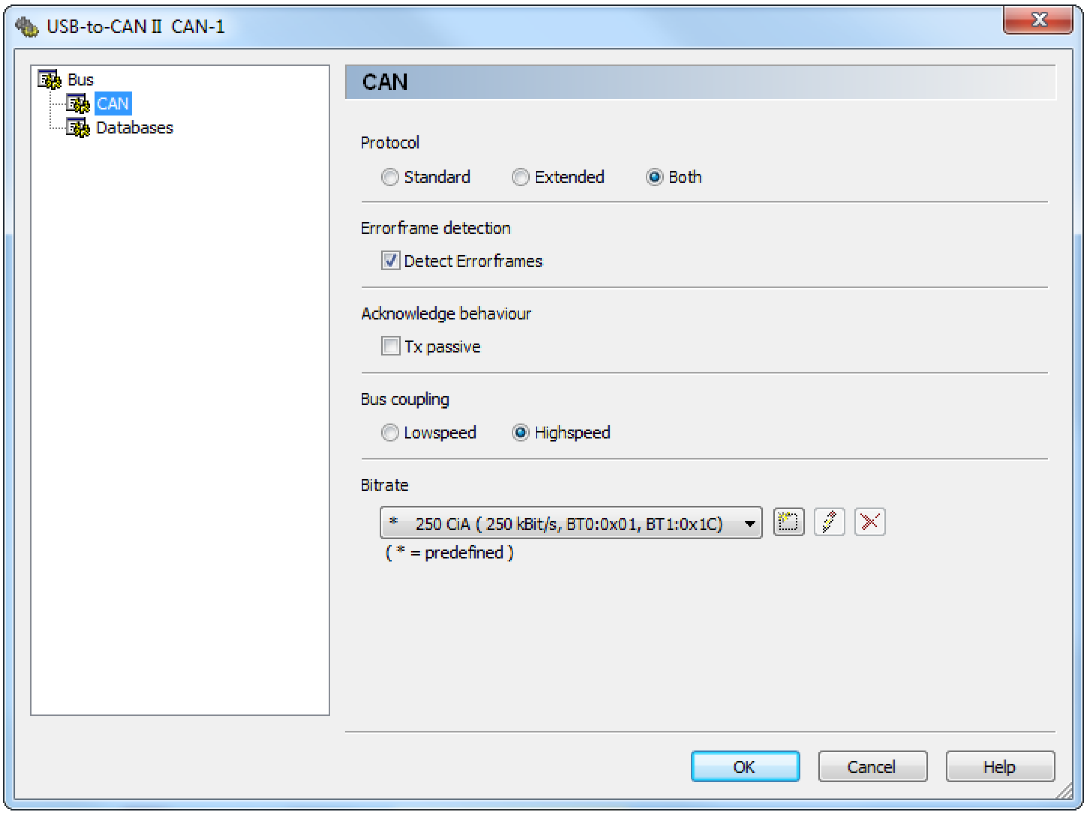
Name: CAN 1

Figure 5.11: Setting a symbolic bus name



Database	Channel	Path
Car_CAN	channelId	C:\Users\Public\Documents\IXXAT\ca...

Figure 5.12: Select/create interpretation database



Protocol

○ Standard ○ Extended ● Both

Errorframe detection

☑ Detect Errorframes

Acknowledge behaviour

□ Tx passive

Bus coupling

○ Lowspeed ● Highspeed

Bitrate

* 250 CiA (250 kBit/s, BT0:0x01, BT1:0x1C)

(* = predefined)

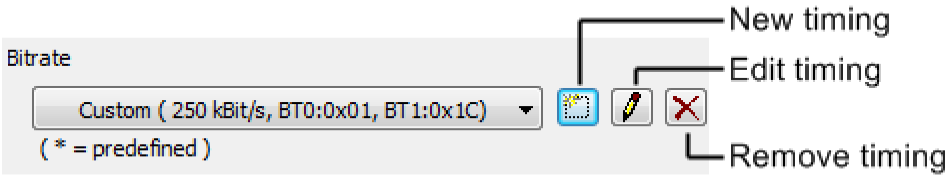
Figure 5.13: CAN Settings

- Error frame detection
- Acknowledge behavior
- Bus coupling
- Timing parameters

Fig. 5.13 shows the dialog to set the CAN controller parameters. In order to identify timing parameters more easily, they are managed via symbolic names. Using the button symbols next to the name, the parameters which are configured for this name can be altered, new entries can be added and old ones can be deleted.

The meaning of the parameters:

Setting	Function
Protocol	Defines the message format with which the CAN controller works (standard 11-bit identifier and/or extended 29-bit identifier)
Detect Errorframes	If this checkbox is set, error frames are passed on to the associated analysis modules
Tx passive	If this checkbox is set, the CAN controller is initialized in Tx-passive mode, i.e. it listens on the bus but behaves passively and therefore does not transmit any acknowledgements or error frames.
Bus coupling	Selects the physical bus coupling of the CAN controller (Highspeed by default, Lowspeed if available). Lowspeed is a fault-tolerant 2-wire standard with max 125 kBit/sec bitrate acc.to ISO 11898-3.

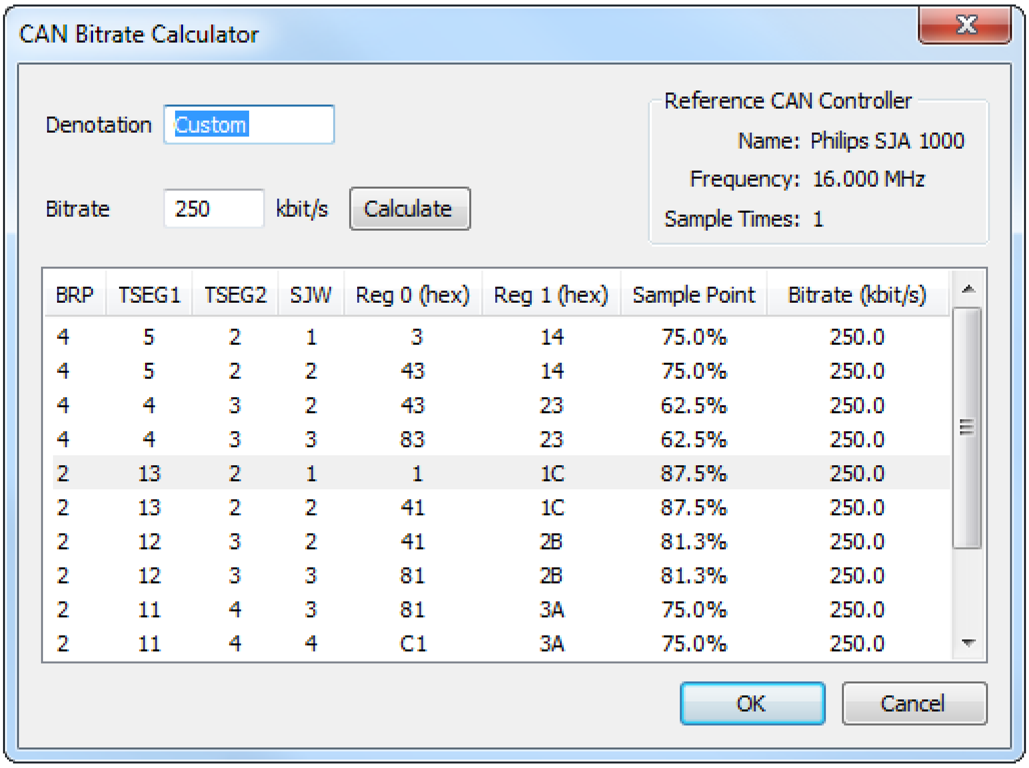


Bitrate

Custom (250 kBit/s, BT0:0x01, BT1:0x1C)

(* = predefined)

Figure 5.14: Create new entry in the Timings dialog or delete entry



CAN Bitrate Calculator

Denotation: Custom

Bitrate: 250 kbit/s

Reference CAN Controller
Name: Philips SJA 1000
Frequency: 16.000 MHz
Sample Times: 1

BRP	TSEG1	TSEG2	SJW	Reg 0 (hex)	Reg 1 (hex)	Sample Point	Bitrate (kbit/s)
4	5	2	1	3	14	75.0%	250.0
4	5	2	2	43	14	75.0%	250.0
4	4	3	2	43	23	62.5%	250.0
4	4	3	3	83	23	62.5%	250.0
2	13	2	1	1	1C	87.5%	250.0
2	13	2	2	41	1C	87.5%	250.0
2	12	3	2	41	2B	81.3%	250.0
2	12	3	3	81	2B	81.3%	250.0
2	11	4	3	81	3A	75.0%	250.0
2	11	4	4	C1	3A	75.0%	250.0

Figure 5.15: The CAN bitrate calculator

Setting a bitrate

The bitrate is selected via the symbolic name of the timing. The timing parameters assigned to the name can be altered, new parameter sets can be added and old ones can be deleted. For this, the buttons next to the symbolic name (Fig. 5.14) are pressed.

CAN Bitrate Calculator

The CAN bitrate calculator (Fig. 5.15) can be opened via the **New** or **Edit** button in the CAN Settings dialog. Here you can choose the timing parameters fitting a desired bitrate.
Once you enter the desired bitrate and press the **Calculate** button, the table displays all suitable combinations of the CAN controller's registers. Choose one by moving the highlighted line up and down, and press **OK** to accept these timing parameters.

Description of the CAN bitrate calculator input fields:

Field	Description
Denotation	Symbolic name of the timing
Bitrate (kbit/s)	Bitrate to be calculated in kBit per second

Description of the columns in the list of calculated values:

Column	Description
BRP	Baudrate Prescaler
TSEG1	Timing Segment 1
TSEG2	Timing Segment 2
SJW	Synchronisation Jump Width
Reg 0 (hex)	Bus timing register 0 (hexadecimal format)
Reg 1 (hex)	Bus timing register 1 (hexadecimal format)
Sample Point	Sample location
Bitrate (kbit/s)	Calculated bitrate with the values of the marked line

Please note: Columns *Reg 0* and *Reg 1* summarize the values of the following five columns: BRP, TSEG1, TSEG2, SJW, and Sample Point, bitcoded in hexadecimal format. Also, column *Bitrate* displays the resulting actual bitrate, which is expected to be equal to the entered desired bitrate.

CAN-FD Settings

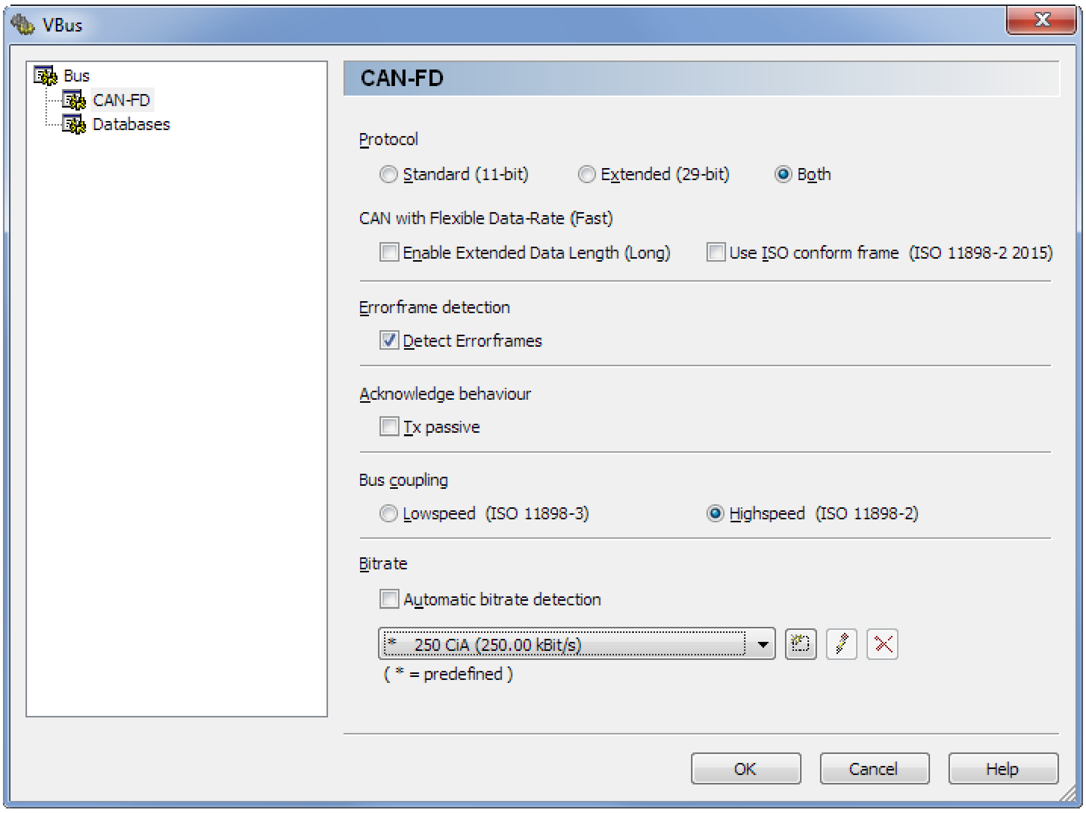
The settings of the CAN-FD controller are defined via branch **CAN-FD** of the properties dialog of a CAN-FD bus. They include the CAN settings as well. These are:

- Message format
- Error frame detection
- Acknowledge behavior
- Buscoupling
- Timing parameters

Fig. 5.16 shows the dialog to set the CAN-FD controller parameters. In order to identify timing parameters more easily, they are managed via symbolic names. Using the button symbols next to the name, the parameters which are configured for this name can be altered, new entries can be added and old ones can be deleted.

The meaning of the parameters:

Setting	Function
Protocol	Defines the message format with which the CAN-FD controller works (standard 11-bit identifier and/or extended 29-bit identifier)
CAN with Flexible Data-Rate (Fast)	Enables the usage of Extended Data Length (Long) and allows to force ISO conform CAN-FD frames according to ISO 11898-2 2015
Detect Errorframes	If this checkbox is set, error frames are passed on to the associated analysis modules
Tx passive	If this checkbox is set, the CAN-FD controller is initialized in Tx-passive mode, i.e. it listens on the bus but behaves passively and therefore does not transmit any acknowledgements or error frames.
Bus coupling	Selects the physical bus coupling of the CAN-FD controller (Highspeed by default, Lowspeed if available). Lowspeed is a fault-tolerant 2-wire standard with max 125 kBit/sec bitrate acc.to ISO 11898-3.



CAN-FD

Protocol

○ Standard (11-bit) ○ Extended (29-bit) ● Both

CAN with Flexible Data-Rate (Fast)

□ Enable Extended Data Length (Long) □ Use ISO conform frame (ISO 11898-2 2015)

Errorframe detection

☑ Detect Errorframes

Acknowledge behaviour

□ Tx passive

Bus coupling

○ Lowspeed (ISO 11898-3) ● Highspeed (ISO 11898-2)

Bitrate

□ Automatic bitrate detection

* 250 CiA (250.00 kBit/s)

(* = predefined)

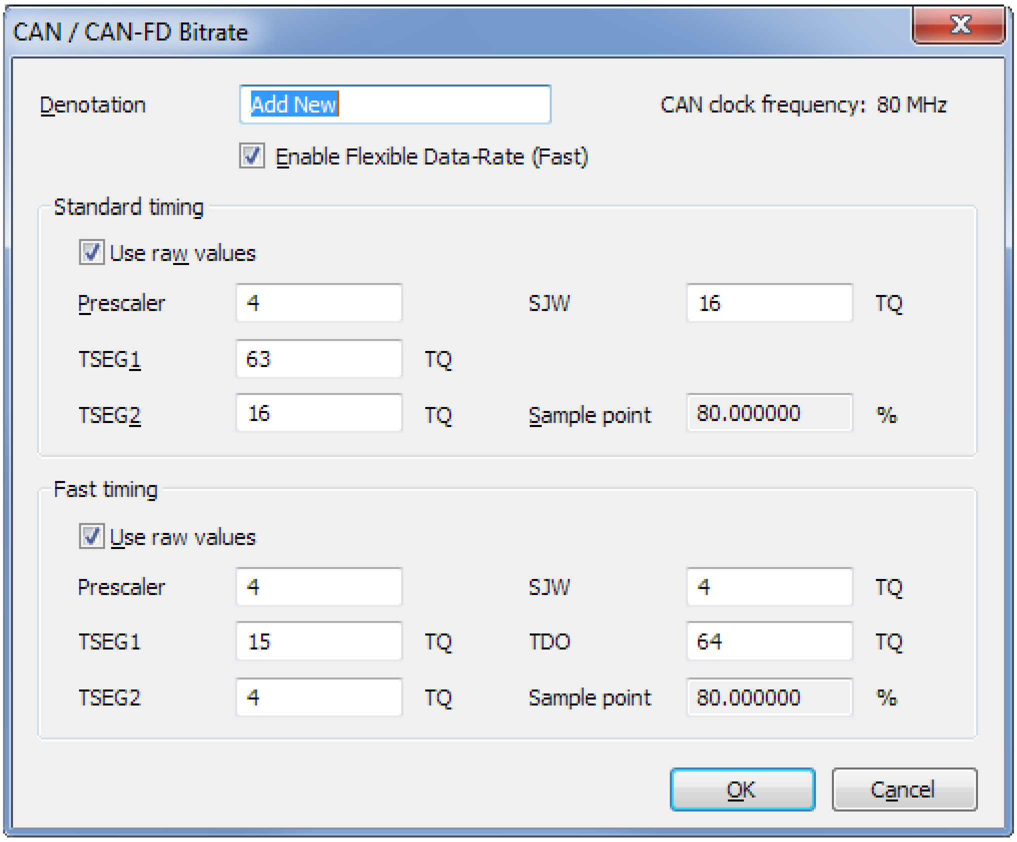
Figure 5.16: CAN-FD Settings

Please note: Running CAN-FD e.g. on a low speed line makes no sense of course, but the CAN-FD controller can be configured to behave like a plain CAN controller if the following conditions are met: Enabling neither Long nor ISO frames, and abstaining from fast bit timings (as shown in figure 5.16).

CAN-FD Bitrate Dialog

The CAN-FD bitrate dialog (Fig. 5.17) can be opened via the **New** or **Edit** button in the CAN-FD Settings dialog.
Firstly, there are two timing sets: **Standard Timing**, and **Fast Timing**. This matches the concept of CAN-FD. As the name says, CAN-FD transmits only the data field of a message in fast speed. The rest of the message, like e.g. the identifier, in normal speed. The speed switch happens in transmission, during every single message. Accordingly, there are two timings, one for normal speed (Standard Timing), and one for fast speed (Fast Timing). **Fast Timing** is accessible if **Enable Fast Data** is checked.
By the checkboxes **Use raw values** the controller dependent native mode (Raw Mode) can be selected. In this mode the CAN-FD controllers' register values are set straightly, rather than being calculated by VCI as intermediary based on the bit rate entered.

Description of the CAN-FD bitrate dialog input fields:



CAN / CAN-FD Bitrate

Denotation: Add New

CAN clock frequency: 80 MHz

☑ Enable Flexible Data-Rate (Fast)

Standard timing

☑ Use raw values

Prescaler: 4

SJW: 16 TQ

TSEG1: 63 TQ

TSEG2: 16 TQ

Sample point: 80.000000 %

Fast timing

☑ Use raw values

Prescaler: 4

SJW: 4 TQ

TSEG1: 15 TQ

TDO: 64 TQ

TSEG2: 4 TQ

Sample point: 80.000000 %

Figure 5.17: The CAN-FD bitrate dialog

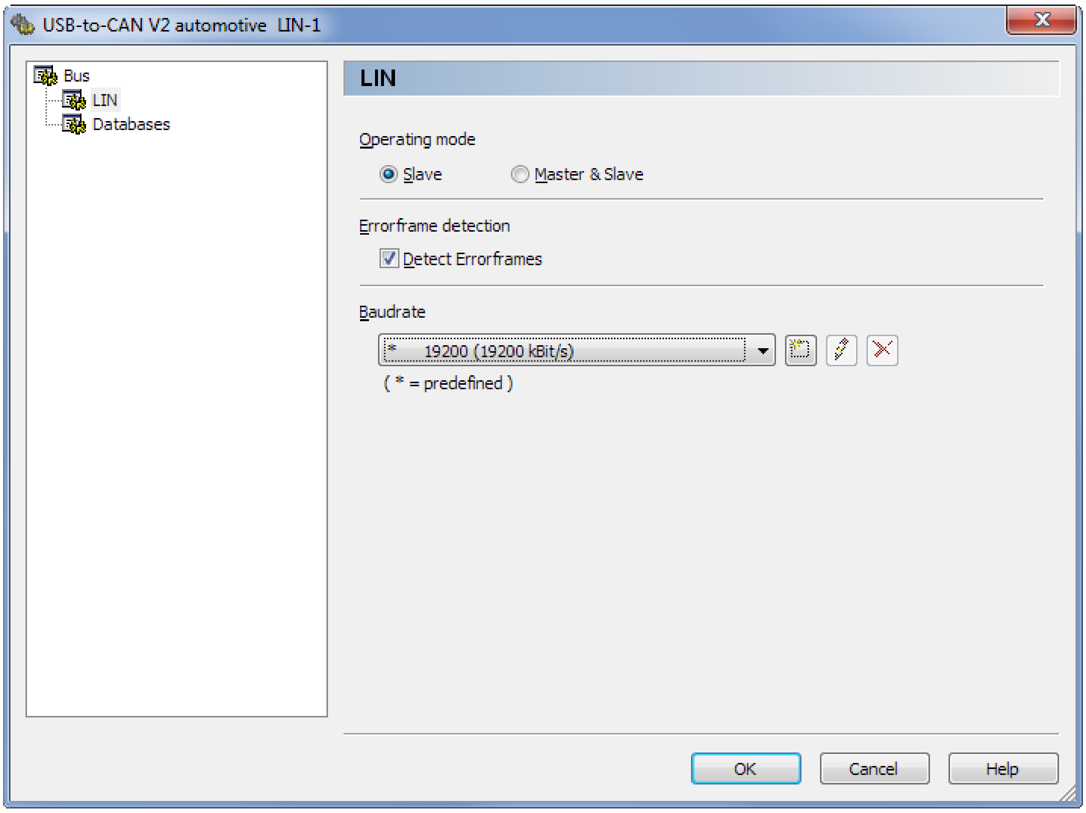
Field	Description
Prescaler	Preceding prescaler in the CAN-FD controller. Only visible if Use raw values is checked.
Bitrate	Desired Bitrate. Only visible if Use raw values is UNchecked.
TSEG1	Length of Time Segment 1 in time quantas. If Use raw values is UNchecked, it comprises the bit timing segments PROP und PHASE1. If Use raw values is checked, it comprises the bit timing segments SYNC, PROP und PHASE1.
TSEG2	Length of Time Segment 2 in time quantas.
SJW	Sync Jump Width for (re-)synchronisation in time quantas.
TDO	Transceiver Delay Offset in time quantas.

Please note: The displayed *Sample point* are calculated from the ratio of *TSEG1* and *TSEG2*. Please find further explanations in the VCI programming manual (PDF), located in its installation folder.

LIN Settings

The settings of the LIN controller are defined via branch **LIN** of the properties dialog of a LIN bus (Fig. 5.18). These are:

- Operating mode
- Errorframe detection
- Baudrate



USB-to-CAN V2 automotive LIN-1

- Bus
  - LIN
  - Databases

LIN

Operating mode

● Slave ○ Master & Slave

Errorframe detection

☑ Detect Errorframes

Baudrate

* 19200 (19200 kBit/s)

(* = predefined)

Figure 5.18: LIN Settings

The meaning of the parameters in the **LIN** section:

Setting	Function
Operating mode	Switches between Slave mode and Master mode. Since the LIN controller Response Table is active in Master mode too, it is denoted as Master & Slave here.
Detect errorframes	If this checkbox is set, error frames are passed on to the associated analysis modules.
Baudrate	Selects the physical serial baudrate of the LIN controller.

Setting a baudrate

The baudrate is selected from the combobox. New baudrates can be defined and old ones can be deleted. For this, the buttons next to the symbolic name are pressed. In order to identify user baudrates more easily, they are managed via symbolic names.

Analysis module pop-up menu

By right-clicking on the icon of an analysis module in the configuration tree, a pop-up menu appears (Fig. 5.19). Every analysis module can be started, renamed or removed from the configuration via this popup menu. In addition it is possible to alter the window size of a module.
A double-click on the icon of an analysis module has the same effect as the **Restore Module** command in the pop-up menu and moves the window of the corresponding module into the foreground.

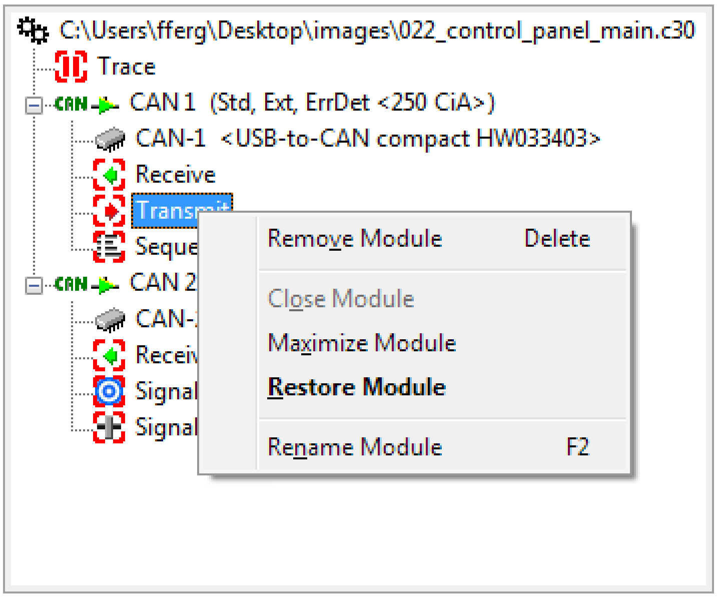
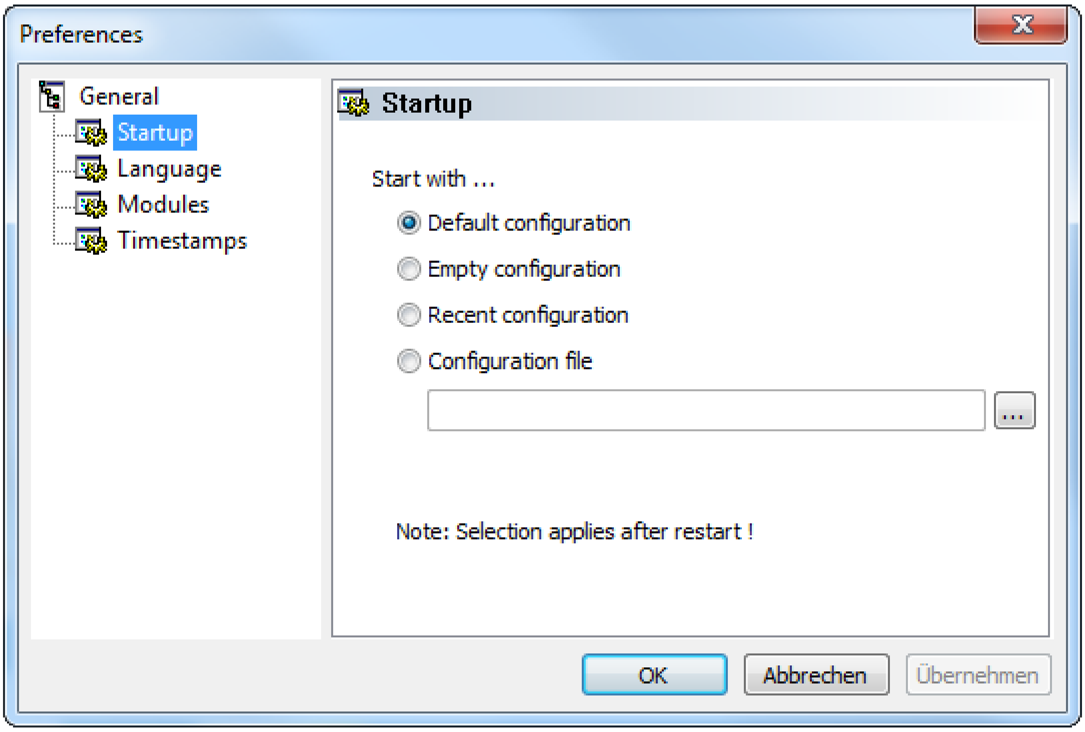


Figure 5.19: Pop-up menu of the analysis modules



Start with ...

● Default configuration
○ Empty configuration
○ Recent configuration
○ Configuration file

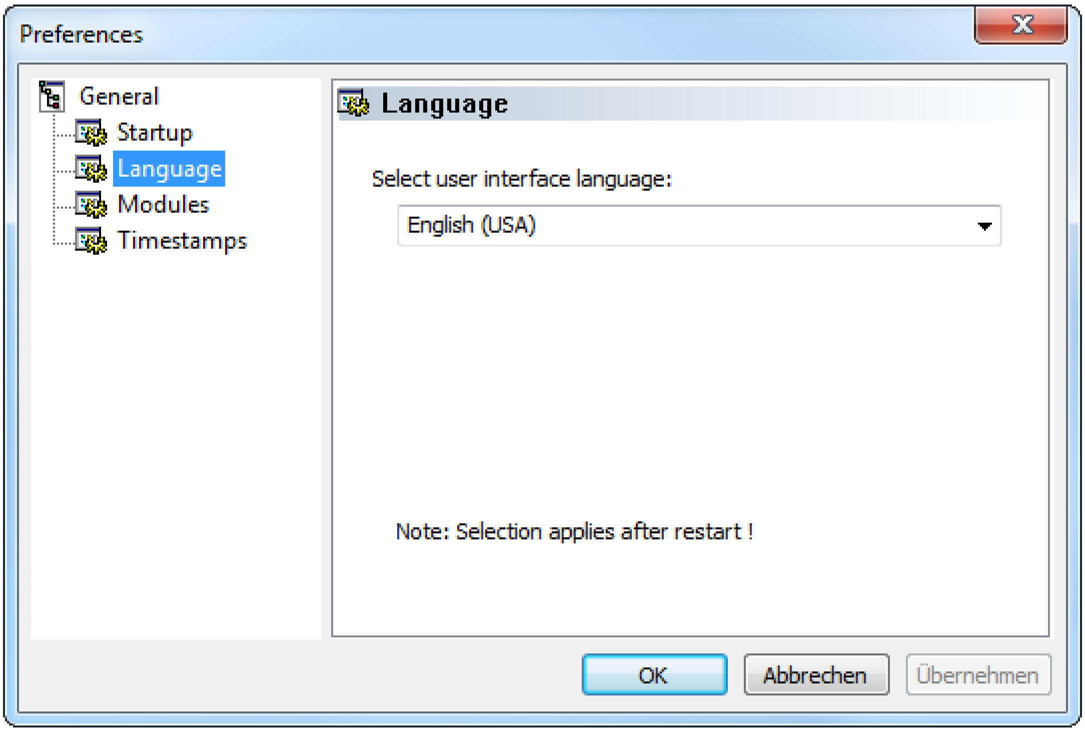
Note: Selection applies after restart !

Figure 5.20: Startup settings dialog

5.1.8 Startup configuration

Within the preferences dialog you can specify, which configuration the program should use on startup (Fig. 5.20). Valid options are:

- The default configuration. Creates a typical configuration containing all the IXXAT interfaces that are connected. For each Bus the Receive Module, the Transmit Module, and their Signal counterparts is added. The Trace module is added to the project.
- An empty configuration
- The recent configuration from the previous canAnalyser session.



Select user interface language:
English (USA)

Note: Selection applies after restart !

Figure 5.21: Language Selection dialog

- The configuration from a given configuration file

The settings are applied during the next start of the canAnalyser.

5.1.9 Language Selection

The language setting of the canAnalyser can be altered with the **Language Selection** dialog (Fig. 5.21). The available languages are shown in a list. After changing the language settings, the canAnalyser has to be restarted for the change to become effective.

5.1.10 Module settings

It is possible to change module specific settings with the **Module settings** dialog (Fig. 5.22). The two options are:

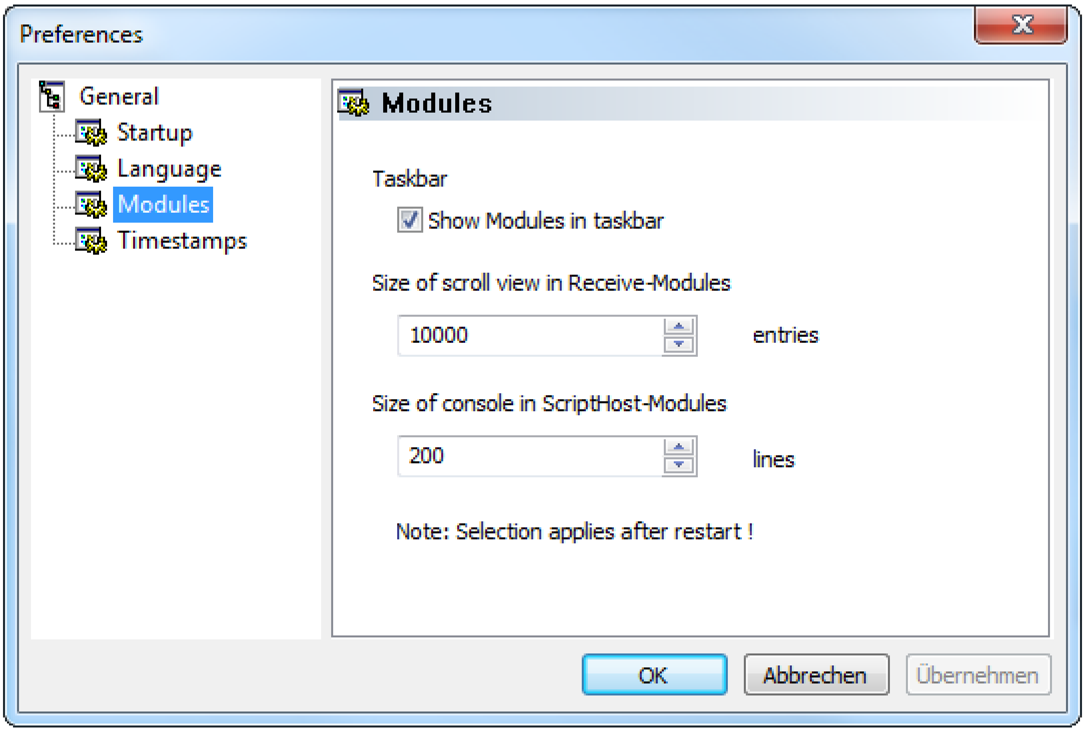
- visibility of the module in the taskbar
- number of rows in the scroll view of receive modules (default 10000, max. 50000 rows)
- size of console in script host programs (default 200, max 20000 rows)

Note that canAnalyser has to be restarted for these options to become effective.

5.1.11 Timestamp settings

You can change the timestamp base time within the **Timestamp settings** dialog (Fig. 5.23). The two options are:

- Reset timestamp on communication start



Modules

Taskbar

☑ Show Modules in taskbar

Size of scroll view in Receive-Modules

10000 entries

Size of console in ScriptHost-Modules

200 lines

Note: Selection applies after restart !

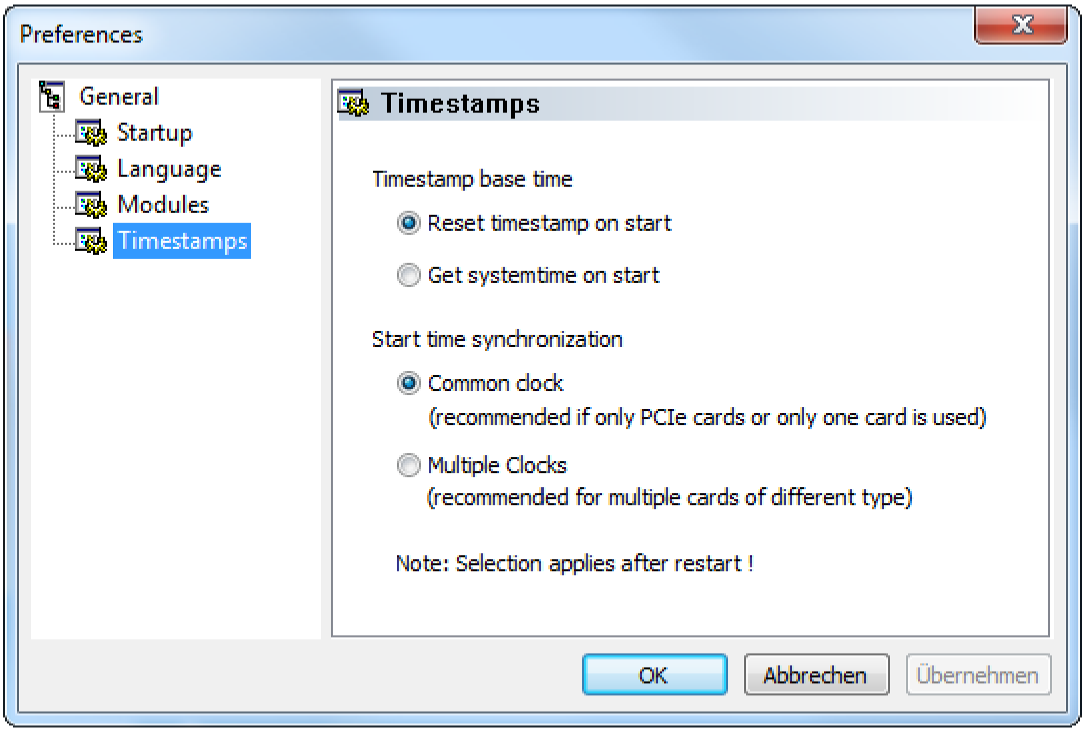
Figure 5.22: Module settings dialog

- Get systemtime on communication start

With the first option the timestamp is set to zero on communication start and subsequent timestamps are relative to this reference time. When using the second option on communication start the system time is taken as reference for the following timestamps.
There are two techniques for the synchronization of the start time, switched between by the options:

- Common clock
- Multiple clocks

The default setting "Common Clock" should be used if only one CAN-Card is used, or if all used CAN-Cards are PCIe based Cards (They share a common clock). Because all sources use a common clock, the first timestamp is derived from the start time of the first source.
The option "Multiple clocks" should be used when the start time of different CAN-Cards with no common clock should be synchronized. Because of runtime delays in the system a small skew between the different sources is possible. Moreover because only the start time is synchronized the drift of different sources could accumulate over the time and lead to wrong assumptions about the order of events (messages) from different sources.
Note that communication has to be restarted for these options to become effective.



Timestamps

Timestamp base time

● Reset timestamp on start

○ Get systemtime on start

Start time synchronization

● Common clock
(recommended if only PCIe cards or only one card is used)

○ Multiple Clocks
(recommended for multiple cards of different type)

Note: Selection applies after restart !

Figure 5.23: Timestamp settings dialog

5.1.12 Menu reference

File menu

Menu item	Function
New	Creates a new, empty configuration
Open...	Opens an existing configuration and sets it up
Save	Saves the current configuration under the file name already specified
Save As...	Saves the current configuration under a new file name
Recent files	Displays the analysis configurations last opened
Exit	Exits the canAnalyser

View menu

Menu item	Function
Modules	Shows/hides the module list
Hardware Controllers	Shows/hides the hardware controllers pane
Layout List	Shows/hides the layout list pane
Event Log	Event Log settings
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar

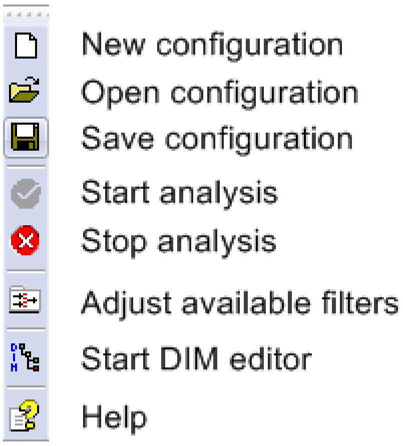


Figure 5.24: Toolbar of the control panel

Functions menu

Menu item	Function
Start Communication	Starts the online analysis
Stop Communication	Stops the online analysis
Available Filters...	Adjust application wide available message filters (see section 5.9.1)
Preferences...	Opens the dialog to select the common canAnalyser settings

Tools menu

Menu item	Function
DIM Editor	Instanciates the additional DIM Database Editor

Windows menu

Menu item	Function
Close All	Closes all modules
Restore All	Restores all modules
Cascade	Arranges the modules one behind the other
Tile horizontally	Splits the modules next to each other (horizontal split)
Tile vertically	Splits the modules next to each other (vertical split)
Save Layout	Remembers current window positions and sizes
Recent Layouts	Lists the last saved layouts

Help menu

Menu item	Function
Help topics	Opens the online help of the control panel
About...	Opens the display of the version information of the canAnalyser

5.1.13 Toolbar

The most important functions of the control panel can also be called via the toolbar (Fig. 5.24).

5.1.14 Hotkeys

Ctrl+N	Create new configuration
Ctrl+O	Open existing configuration
Ctrl+S	Save current configuration
Ctrl+M	Show/hide modules
Ctrl+L	Show/hide layout list
F5	Start communication
Shift+F5	Stop communication
Shift+Insert	Create new bus
Alt+L	Remember layout
Alt+S	Show controller status
Alt+C	Assign controller to bus
Space	Enable/Disable bus
F1	Online-Help
Del	Remove Module

5.2 Receive module

5.2.1 Overview

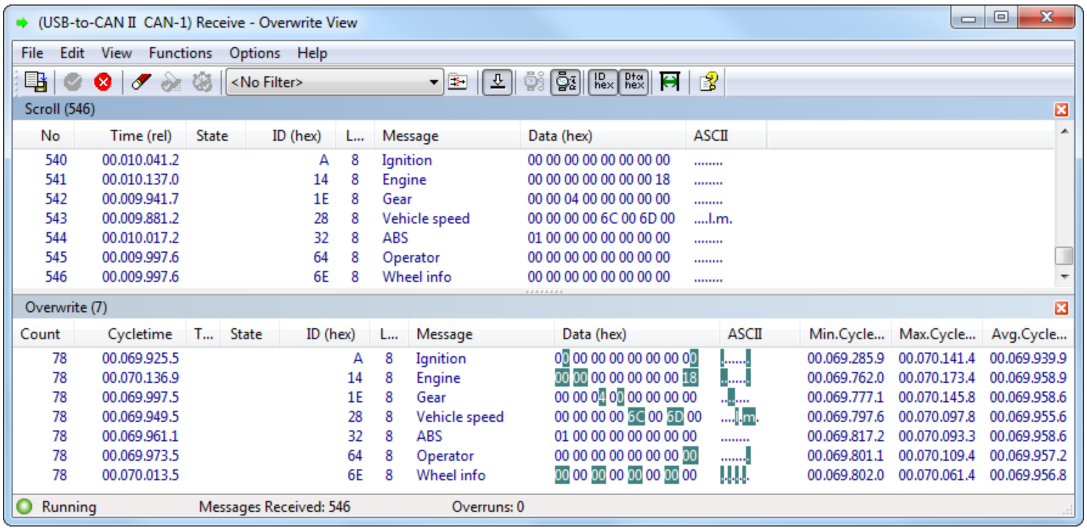
The Receive module represents a central module for the analysis of layer-2 messages. It provides the following analysis functions:

- Reception and display of layer-2 messages in order of the time of reception (Scroll View)
- Statistic display of the received messages (Overwrite View)
- Show/hide any messages (filter function)
- Display of changes in the data field of the received messages (Overwrite View)
- Monitoring of the cycle time of individual messages (Overwrite View)
- Display of bus errors/error frames
- Display of data in different forms (hexadecimal, decimal, ASCII)
- Display of total number of received messages (Scroll View) or number of received messages per identifier (Overwrite View)
- Display of the names of messages with the definitions of the project database

The Overwrite and Scroll Views (Fig. 5.25) are updated simultaneously. In brackets behind the name of the View the number of available received messages (lines) is shown.
As various instances of the Receive module can be started by the control panel, every Receive module can be adapted individually to the messages or message groups to be analysed.

5.2.2 Scroll View

Messages are listed on the **Scroll** view (Fig. 5.25, top) in the order of reception with the following information:



Scroll (546)

No	Time (rel)	State	ID (hex)	L...	Message	Data (hex)	ASCII
540	00.010.041.2		A	8	Ignition	00 00 00 00 00 00 00 00	....
541	00.010.137.0		14	8	Engine	00 00 00 00 00 00 00 18	....
542	00.009.941.7		1E	8	Gear	00 00 04 00 00 00 00 00	....
543	00.009.881.2		28	8	Vehicle speed	00 00 00 00 6C 00 6D 00	....l.m.
544	00.010.017.2		32	8	ABS	01 00 00 00 00 00 00 00	....
545	00.009.997.6		64	8	Operator	00 00 00 00 00 00 00 00	....
546	00.009.997.6		6E	8	Wheel info	00 00 00 00 00 00 00 00	....

Overwrite (7)

Count	Cycletime	T...	State	ID (hex)	L...	Message	Data (hex)	ASCII	Min.Cycle...	Max.Cycle...	Avg.Cycle...
78	00.069.925.5			A	8	Ignition	00 00 00 00 00 00 00 00	....	00.069.285.9	00.070.141.4	00.069.939.9
78	00.070.136.9			14	8	Engine	00 00 00 00 00 00 00 18	....	00.069.762.0	00.070.173.4	00.069.958.9
78	00.069.997.5			1E	8	Gear	00 00 04 00 00 00 00 00	....	00.069.777.1	00.070.145.8	00.069.958.6
78	00.069.949.5			28	8	Vehicle speed	00 00 00 00 6C 00 6D 00	....l.m.	00.069.797.6	00.070.097.8	00.069.955.6
78	00.069.961.1			32	8	ABS	01 00 00 00 00 00 00 00	....	00.069.817.2	00.070.093.3	00.069.958.6
78	00.069.973.5			64	8	Operator	00 00 00 00 00 00 00 00	....	00.069.801.1	00.070.109.4	00.069.957.2
78	00.070.013.5			6E	8	Wheel info	00 00 00 00 00 00 00 00	....	00.069.802.0	00.070.061.4	00.069.956.8

Running Messages Received: 546 Overruns: 0

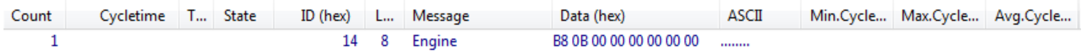
Figure 5.25: Receive module

Column	Meaning
No	Consecutive number of the received object
Time (abs/rel)	Time stamp of reception, optionally absolute in UTC time format or relative to the previously received message; by right-clicking on the column heading, the display of hours and minutes can be switched on or off
State	Display of the reception status flags
ID (hex/dec)	Identifier of the received message
DLC	Data length code, codifies the number of data bytes
Message	Name assigned to the identifier of the received message in the database
Data (hex/dec)	Display of the received data in byte interpretation
ASCII	Display of the received data in ASCII interpretation

Display of the receive status flags

The receive status is displayed in the Receive module in the column **Status** with various letters. If the letter is visible, the status is set:

Status	Bustype	Meaning
C	-	Controller overrun: Messages were lost.
D	-	Driver queue overrun: The PC could not read out the driver queue fast enough. Messages were lost.
Q	-	Software queue overrun: The PC could not read out the internal software queue fast enough. Messages were lost.
S	-	Self-reception: Transmit and receive module use the same controller
E	CAN	Extended CAN frame: If E is not displayed, a standard CAN frame was received.
F	CAN-FD	A Fast Data frame was received.
L	CAN-FD	Frame having CAN-FD Extended Data Length (12, 16, 20, 24, 32, 48, 64 Bytes)"
E	LIN	Enhanced CRC: A frame in enhanced CRC format acc.to LIN 2.0+ was received.
I	LIN	ID only: An ID only (i.e. a LIN Master request) message was received.



Count	Cycletime	T...	State	ID (hex)	L...	Message	Data (hex)	ASCII	Min.Cycle...	Max.Cycle...	Avg.Cycle...
1				14	8	Engine	B8 0B 00 00 00 00 00 00	....			

Figure 5.26: First reception of an identifier in the Overwrite View

5.2.3 Overwrite View

This display mode is useful to obtain an overview of the status of the messages in the network.
In the **Overwrite** view of the Receive module (Fig. 5.25, bottom), the messages are sorted according to identifiers or reception rate. The information of the last message received is always shown in each case. When change monitoring is switched on, changed nibbles are highlighted in color.
The maximum individual line heights are kept to avoid shaking and "pumping" of the View. To reset the stored line heights, do clear the View, do open the font dialog, or do toggle the Word wrap feature.

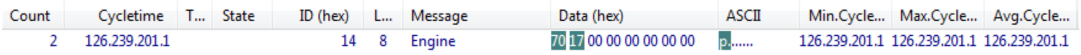
The **Overwrite** view is divided into the following columns:

Column	Meaning
Count	Number of received messages with this identifier
Cycletime / Time (abs)	Optionally last cycle time of the message or absolute time stamp of the last reception in relation to the start time of the hardware; by right-clicking on the column heading, the display hours and minutes can be switched on or off
Timeout	If a cycle time is defined in the database, timeouts of the database are displayed by a ⏱ icon
State	Display of the receive status (such as Scroll View)
ID (hex/dec)	Identifier of the received message
DLC	Data length code, codifies the number of data bytes
Message	Name of the message assigned in the database
Data (hex/dec)	Display of the received data in byte interpretation. If change monitoring is enabled, the data contents which have been changed once are highlighted in color
ASCII	Display of the received data in ASCII interpretation
Min.Cycletime	Smallest measured cycle time of the message
Max.Cycletime	Largest measured cycle time of the message
Avg.Cycletime	Mean cycle time of the message

An up arrow icon ▲ marks the currently sorted column.
Two different sorting orders are possible at this time: Sort ascendingly by Identifier, or Sort descendingly by reception count (most common identifiers on top).
Click on the corresponding column header to switch the sort order.

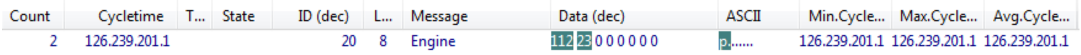
5.2.4 Data change detection

The data change detection occurs in the Overwrite View of the Receive module and can be enabled and disabled via the menu item **Options**.
With the first reception of an identifier, a new message is added in the Overwrite View (Fig. 5.26).
The receive counter (**Number**) counts one received message. No **Cycletime** is displayed yet, as for for the calculation of this the same identifier must have been received at least twice.
After the second reception of the identifier, the changed data contents compared with the first reception are highlighted in color in the column **Data** if the data change display is switched on.



Count	Cycletime	T...	State	ID (hex)	L...	Message	Data (hex)	ASCII	Min.Cycle...	Max.Cycle...	Avg.Cycle...
2	126.239.201.1			14	8	Engine	70 17 00 00 00 00 00 00	p.......	126.239.201.1	126.239.201.1	126.239.201.1

Figure 5.27: Hexadecimal data change display



Count	Cycletime	T...	State	ID (dec)	L...	Message	Data (dec)	ASCII	Min.Cycle...	Max.Cycle...	Avg.Cycle...
2	126.239.201.1			20	8	Engine	112 23 0 0 0 0 0 0	p.......	126.239.201.1	126.239.201.1	126.239.201.1

Figure 5.28: Decimal data change display

If the display of data is set to hexadecimal, the changed nibbles (top or bottom 4 bits of a data byte) are shown in color (Fig. 5.27).
With decimal display of the data field, changes to the received data bytes can only be highlighted as a whole. (Fig. 5.28).
The data change display and the signalling of timeouts can be reset in the context menu of the display area.
By clicking on a message with the right mouse button, it is possible to select whether the data change display should be reset for this or for all messages (Fig. 5.29).
The data change display always occurs in relation to the data bytes of the first reception of an identifier or in relation to the content of the data field at the time of resetting the data change display. If a different value was already received for a data byte than in the reference message (first reception or message when resetting), this data byte remains highlighted, even if an identical data field to the reference message is received again.

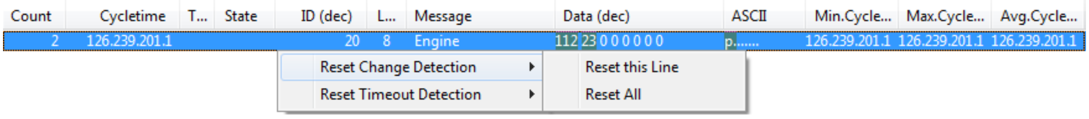
5.2.5 Display of errors

Faulty or defective messages are treated as normal messages in the Receive module. Instead of the identifier or symbolic name, **Error** is displayed in the identifier column.
The reason for the occurrence of an error is given in the following format in the **Data** column:
Error <ECC errorcode> : <Description of the set errorcode bits>
The **ECC errorcode** is read from the **Error Code Capture** register of a SJA1000 CAN controllers. Behind the colon the descriptions of the set errorcode bits are displayed each separated by a '|' character:

Error type	Meaning
Bit error	Bit monitoring error
Stuff error	Bit-stuffing error: A format field coded via bit-stuffing contains a sequence of 6 or more equivalent bits
Form error	Form error: Bit-field with fixed value has inadmissible value
Other error	Error other than the above-mentioned errors

5.2.6 Filtering of messages

For each Receive module it is possible to configure which messages it should receive. With the aid of a filter, certain messages become visible or invisible to the Receive module. Via the menu



Count	Cycletime	T...	State	ID (dec)	L...	Message	Data (dec)	ASCII	Min.Cycle...	Max.Cycle...	Avg.Cycle...
2	126.239.201.1			20	8	Engine	112 23 0 0 0 0 0 0	p.......	126.239.201.1	126.239.201.1	126.239.201.1

Figure 5.29: Context menu for monitoring display

item **Functions | Available Filters...** global message filters can be created which can accept or reject certain messages for reception. The filter selection is done by menu item **Functions | Select Filter**.

5.2.7 Menu reference

File menu

Menu item	Function
Export Messages...	Exports the received messages to a file
Exit	Exits the Receive module

Edit menu

Menu item	Function
Copy CSV	Copies marked lines CSV formatted to clipboard
Toggle Marker *	Sets or Removes a Marker for selected message
Previous Marker *	Jumps to previous Marker (no wraparound)
Next Marker *	Jumps to next Marker (no wraparound)
Set/Release Time Reference *	Sets Timestamp Zero for selected message / Releases previously set Timestamp Zero
Jump to Time Reference *	Jumps to previously set Timestamp Zero message

* Only available in Scroll View

View menu

Menu item	Function
Time relative	Display of the time stamp absolute or relative to the previously displayed telegram
ID hex	Display of the message identifiers in hexadecimal or decimal notation
ID representation	Representation options of the identifier column
Data hex	Display of the data of layer-2 messages in hexadecimal or decimal notation
Data representation	Representation options of the data column
Data granulation	Displays the data BYTE-wise or combines two bytes each as a WORD. With WORD display, it is possible to choose between Little Endian (Intel) or Big Endian format (Motorola).
Draw Guides	Draws additional horizontal guides between the lines in grey
Word wrap lines	Displays the data over several lines if the specified column width is not sufficient for the display of the complete data field.
Show recent Frames	Always the most recent telegrams are displayed in Scroll View
Scroll View	Shows/hides the scroll view
Overwrite View	Shows/hides the overwrite view
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar

Functions menu

Menu item	Function
Start	Starts message reception of the Receive module
Stop	Stops message reception
Available Filters...	Adjust application wide available message filters (see section 5.9.1)
Select Filter	Selects a message filter
Reset Change Detection All	Resets the data change detection
Reset Timeout Detection All	Resets cycle time detection
Clear All	Deletes the display and resets the receive counter
Autosize Columns	Optimizes the column widths

Options menu

Menu item	Function
Data Change Detection	Switches the data change display on or off
Timeout Detection	Switches cycle time monitoring on or off
Change Detection Color...	Opens a dialog to select the color with which changed data are highlighted
Fonts...	Opens a dialog to select the font type in which the data are displayed in all Views

Help menu

Menu item	Function
Help Topics	Opens the online help of the Receive module
About...	Opens the display of the version information of the Receive module

5.2.8 Toolbar

The most important functions of the Receive module can also be called via the toolbar (Fig. 5.30).

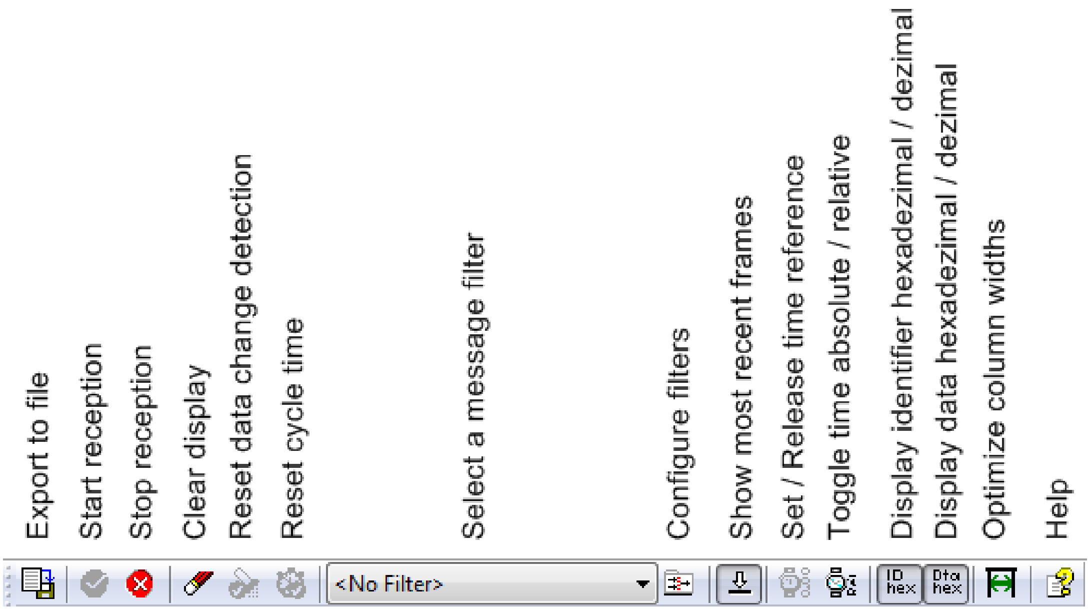


Figure 5.30: Toolbar of the Receive module

5.2.9 Hotkeys

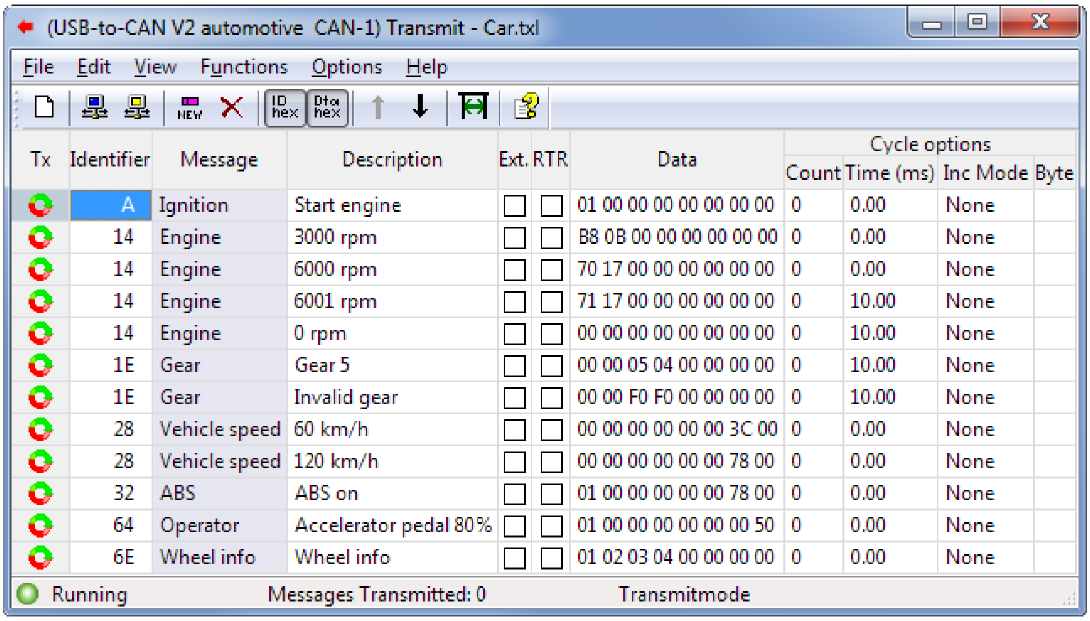
Ctrl+E	Export the available received messages to a file
Ctrl+C	Copy marked lines CSV formatted to clipboard
Ctrl+W	Close the Receive Module
Ctrl+F2	Toggle Marker in Scroll View
Shift+F2	Go to Previous Marker in Scroll View
F2	Go to Next Marker in Scroll View
Ctrl+0	Jump to Time Reference message
F5	Start message reception
Shift+F5	Stop message reception
Ctrl+I	Configure Filter
F8	Clear all Views
F1	Online-Help
Ctrl+TAB	Switch between Scroll View and Overwrite View
PageUp	Scroll one page backward in current View
PageDown	Scroll one page ahead in current View
Ctrl+PageUp	Scroll 1000 messages backward in current View
Ctrl+PageDown	Scroll 1000 messages ahead in current View
Ctrl+1..9	Jump to 10%..90% of current View

5.3 Transmit module

5.3.1 Overview

The Transmit module provides functions for the manual and cyclic transmission of bus messages and can be used to simulate nodes or to test the reaction of nodes to certain messages.
The following functionality is provided by the Transmit module:

- Transmission of individual data and remote messages
- Transmission of any number of data or remote messages


(USB-to-CAN V2 automotive CAN-1) Transmit - Car.txt
File Edit View Functions Options Help
NEW
ID hex
Dta hex

Tx	Identifier	Message	Description	Ext.	RTR	Data	Cycle options			
							Count	Time (ms)	Inc Mode	Byte
●	A	Ignition	Start engine	□	□	01 00 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	3000 rpm	□	□	B8 0B 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	6000 rpm	□	□	70 17 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	6001 rpm	□	□	71 17 00 00 00 00 00 00	0	10.00	None	
●	14	Engine	0 rpm	□	□	00 00 00 00 00 00 00 00	0	10.00	None	
●	1E	Gear	Gear 5	□	□	00 00 05 04 00 00 00 00	0	10.00	None	
●	1E	Gear	Invalid gear	□	□	00 00 F0 F0 00 00 00 00	0	10.00	None	
●	28	Vehicle speed	60 km/h	□	□	00 00 00 00 00 00 3C 00	0	0.00	None	
●	28	Vehicle speed	120 km/h	□	□	00 00 00 00 00 00 78 00	0	0.00	None	
●	32	ABS	ABS on	□	□	01 00 00 00 00 00 78 00	0	0.00	None	
●	64	Operator	Accelerator pedal 80%	□	□	01 00 00 00 00 00 00 50	0	0.00	None	
●	6E	Wheel info	Wheel info	□	□	01 02 03 04 00 00 00 00	0	0.00	None	

● Running   Messages Transmitted: 0   Transmitmode

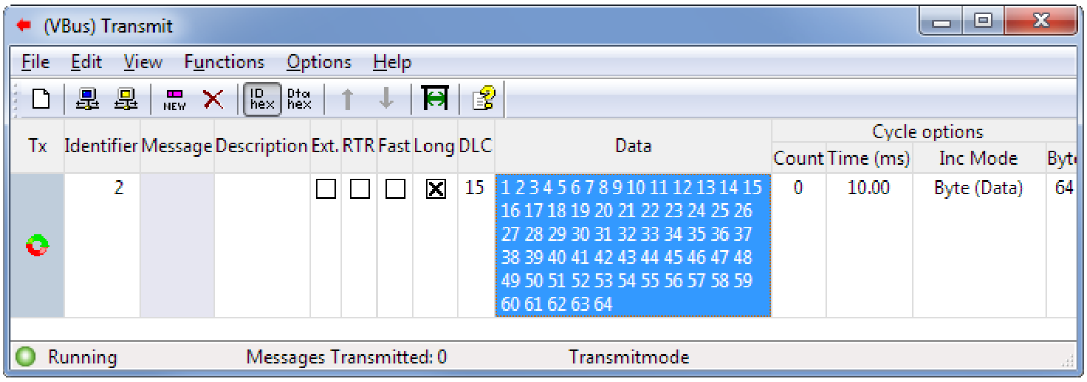
Figure 5.31: Transmit module CAN

- with a certain cycle time
- with incrementing of the identifier or of any data byte or word

5.3.2 Program window CAN

In the Transmit module (Fig. 5.31), the objects to be transmitted are entered in a table, that is displayed centrally in the Transmit module. The entries are transmitted by selecting the message and executing command **Transmit Single Message** or **Transmit Cyclic Message**.
The name of a transmit object is taken from the project database specified in the canAnalyser control panel and automatically added.

The CAN transmit table has the following columns:



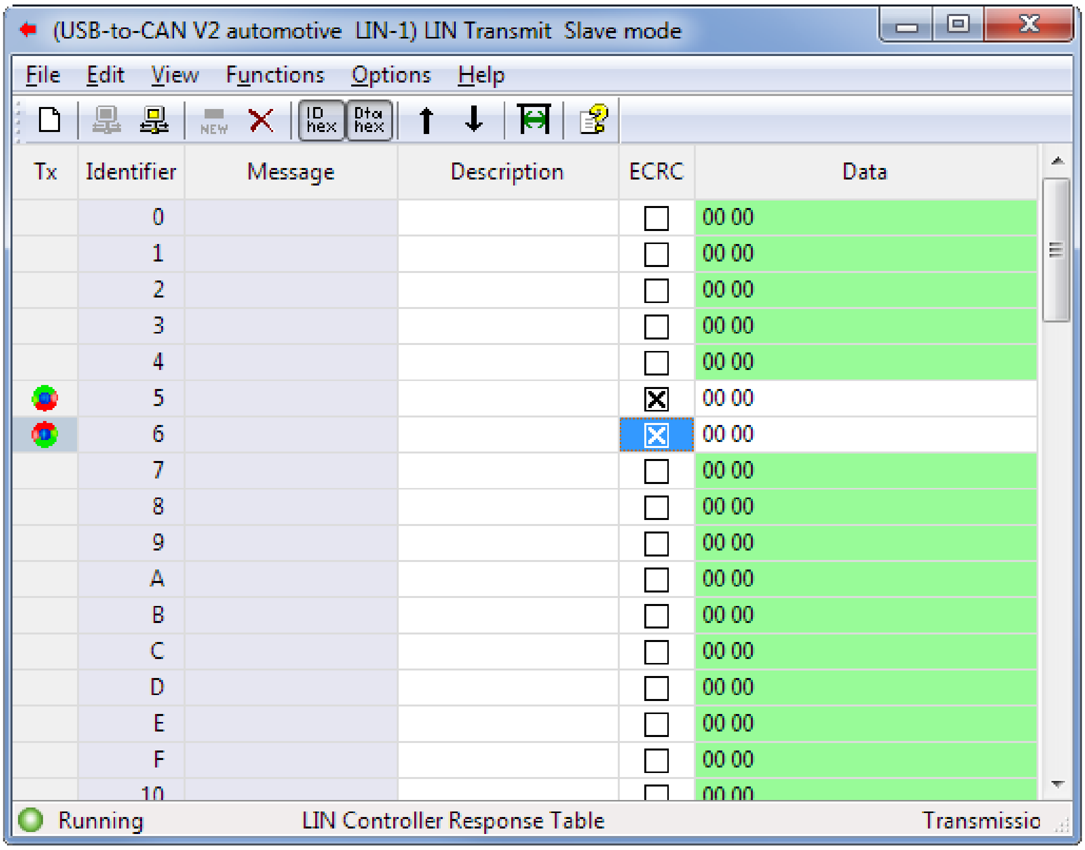
Tx	Identifier	Message	Description	Ext.	RTR	Fast	Long	DLC	Data	Cycle options			
										Count	Time (ms)	Inc Mode	Byt
⟳	2			□	□	□	☒	15	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64	0	10.00	Byte (Data)	64

Figure 5.32: Transmit module CAN-FD

Column	Meaning
Tx	Icon ⟳ for transmission state visualization. It's rotating as long as the message's cyclic transmission is active. Icon ● shows that cyclic transmission is done directly by the hardware.
Identifier	Identifier of the transmit object
Message	Name assigned to the identifier in the database used
Description	Additional user-defined description of this transmit object. This description allows differentiation of the transmit objects with the same identifier and is only valid in the Transmit module.
Ext.	Defines whether a telegram is transmitted in extended frame format (29 bit identifier). This does NOT override the protocol setting in the CAN settings dialog.
RTR	Defines whether a data or a remote telegram is transmitted (only CAN)
Data	Data of the layer-2 message
Cycle options	The settings for cyclic transmit objects are specified in this column
Count	Number of transmit repeats; 0 stands for continual transmission
Time	Cycletime in milliseconds
Inc Mode	Operating mode of cyclic transmission (with/without increment). **None**: No incrementing. **Identifier**: Incrementing of identifier with each transmission. **Byte (Data)**: Incrementing of the databyte defined in the column **Byte** with each transmission. **Word (Data)**: Incrementing of a 16-bit value (compiled from 2 databytes), beginning with the databyte defined in the column **Byte** with each transmission
Byte	Start byte, with which incrementing of the data field is carried out when an increment mode is switched on (see Mode column).

5.3.3 Program window CAN-FD

The CAN-FD transmit table (Abb. 5.32) has the following columns in addition to the ones of the CAN transmit table:


(USB-to-CAN V2 automotive LIN-1) LIN Transmit Slave mode
File
Edit
View
Functions
Options
Help
NEW
ID hex
Dta hex
Running
LIN Controller Response Table
Transmissio

Tx	Identifier	Message	Description	ECRC	Data
	0			□	00 00
	1			□	00 00
	2			□	00 00
	3			□	00 00
	4			□	00 00
●	5			☒	00 00
●	6			☒	00 00
	7			□	00 00
	8			□	00 00
	9			□	00 00
	A			□	00 00
	B			□	00 00
	C			□	00 00
	D			□	00 00
	E			□	00 00
	F			□	00 00
	10			□	00 00

Figure 5.33: Transmit module LIN (Slave mode)

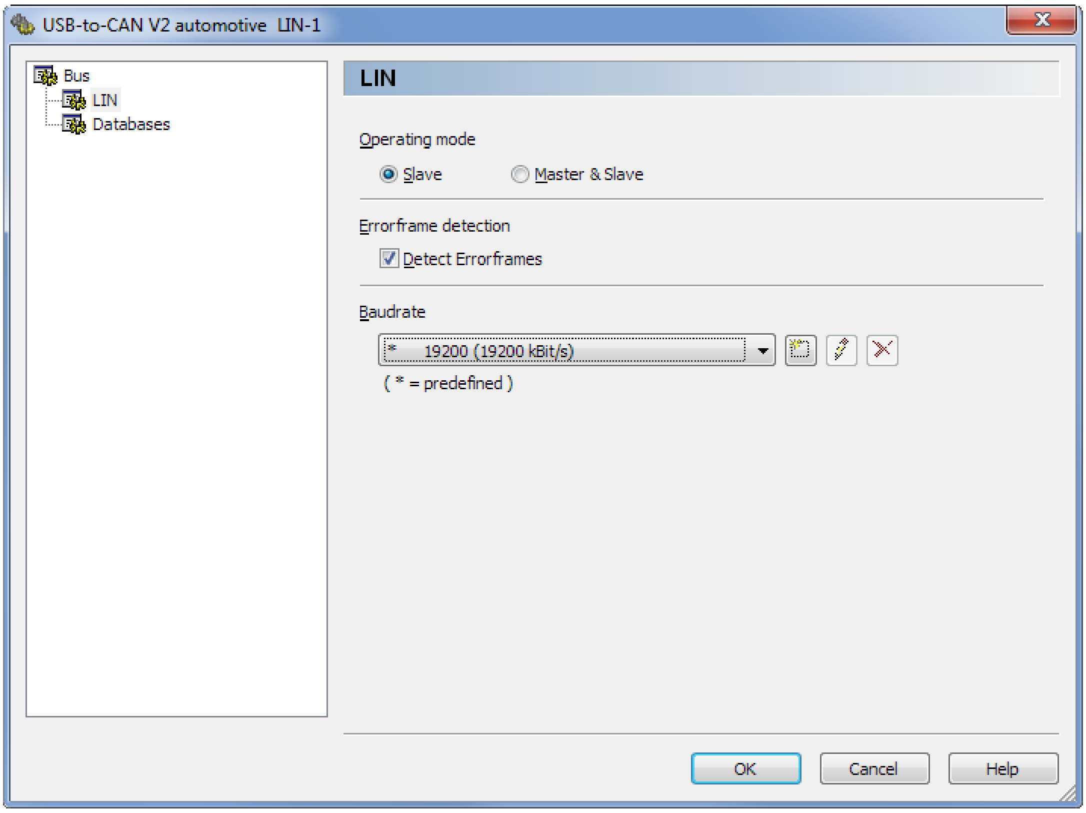
Column	Meaning
Fast	Defines whether a telegram is transmitted in fast speed (FD).
Long	Defines whether a long telegram is transmitted i.e. with extended data length acc.to CAN-FD. This is only possible if the option **Extended Data Length (Long)** in the CAN-FD settings dialog of the control panel is enabled.
DLC	Codifies the length of the data. The value range is 0 to 15. Values 0 to 8 correspond to the actual byte length, for the values 9 to 15 these increments apply: 12, 16, 20, 24, 32, 48, 64 bytes data length. The input is being quantised accordingly. This column and the column Data are mutually adjusting.

5.3.4 Program window LIN

The Transmit module running on LIN (Fig. 5.33) shows a static table with all 64 possible LIN identifiers sorted ascendingly. Special messages fall into line with them.
The name of a transmit object is taken from the assigned .LDF in the canAnalyser control panel and automatically added.
Depending on the LIN operating mode both the layout and the behaviour are slightly different. The LIN operating mode is set in the hardware configuration dialog of the LIN Controller in the canAnalyser control panel. It can be switched at any time (Abb. 5.34).

There is a separate configuration set for LIN Master mode and for LIN Slave mode.

Contrary to CAN and LIN Master mode, messages can not spontaneously be sent in LIN Slave mode. A LIN Slave responds to an external LIN Master request (IDO), which is handled by the hardware controller itself. The latter uses a so-called Response Table, that is visualised by the



Operating mode

● Slave ○ Master & Slave

Errorframe detection

☑ Detect Errorframes

Baudrate

* 19200 (19200 kBit/s)

(* = predefined)

Figure 5.34: LIN settings

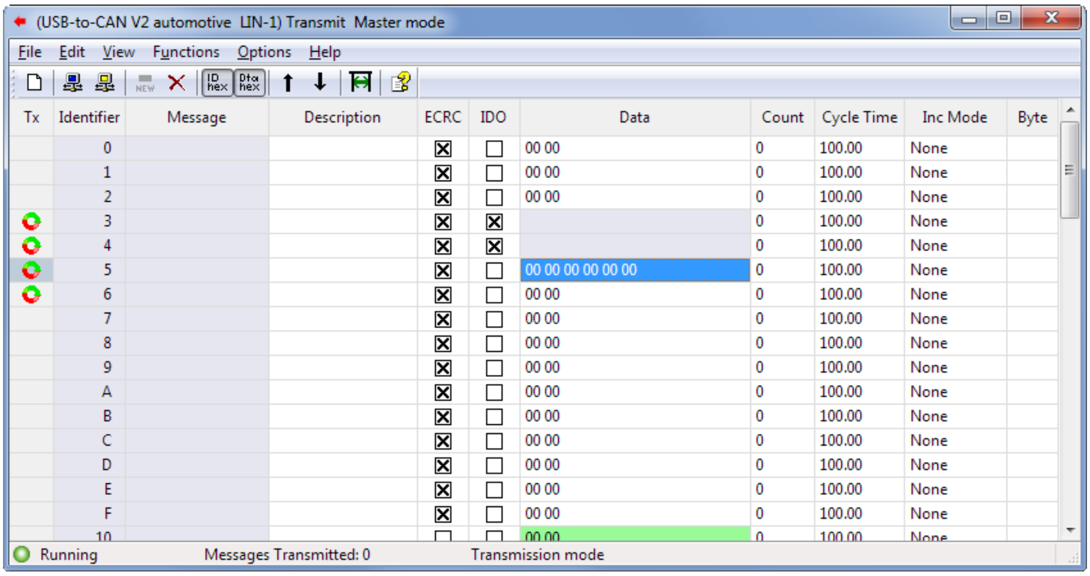
Transmit module in Slave mode (Abb. 5.33). This hardware based processing is also called *auto response* or *auto transmit* in the following.

Even in LIN Master mode, slave behaviour is implemented in firmware by means of an *implicit Response Table*. This can make for the curious situation where the Master responds to its own requests. Hence, operation and presentation of the Response Table in LIN Master mode shall be addressed particularly here. See also the popup menu description below.

By default, all LIN identifiers of the Response Table are disabled. This is illustrated by an empty **Tx** column. A LIN identifier needs to be enabled explicitly both in Slave Mode and in Master mode to allow for transmitting it automatically. An enabled identifier is one with a ● resp ● icon in the **Tx** column. In LIN Slave mode, simply click it, or use the popup menu to enable it.

In LIN Master mode, when manual and cyclic transmission as with CAN is possible, not the *Response Table* of the LIN Controller, but a *transmit table* is displayed. Handling of the *implicit Response Table* is woven into it. A Response Table entry clearly has less parameters than a transmit table entry, only the **data** field (bytes and length). More on that later.
The entries are transmitted by selecting the row and then clicking main menu items **Transmit Single Message** resp **Transmit Cyclic Message** or their toolbar matches respectively.

When a Response Table entry in LIN Master mode is activated, its presentation alters: The **Data** cell turns to royal blue, the **IDO** box gets checked, and the send icon becomes ●. So, the contents of the auto response is entered in the data cell, which is the trick of weaving the Response Table entry into the transmit table, since the data cell is unoccupied for a checked **IDO** cell, and is available for entering the auto response around it.
Once again, the **IDO** checkbox allows for switching the presentation of the response table entry and the transmit table entry of a LIN identifier in LIN Master mode. Physically both are existing independently and simultaneously, and can be configured differently, of course. Even if the cells depicting the cycle options (**Count**, **Cycle Time** etc) are shown with such a Response Table



(USB-to-CAN V2 automotive LIN-1) Transmit Master mode

Tx	Identifier	Message	Description	ECRC	IDO	Data	Count	Cycle Time	Inc Mode	Byte
	0			☒	☐	00 00	0	100.00	None	
	1			☒	☐	00 00	0	100.00	None	
	2			☒	☐	00 00	0	100.00	None	
●	3			☒	☒		0	100.00	None	
●	4			☒	☒		0	100.00	None	
●	5			☒	☐	00 00 00 00 00 00	0	100.00	None	
●	6			☒	☐	00 00	0	100.00	None	
	7			☒	☐	00 00	0	100.00	None	
	8			☒	☐	00 00	0	100.00	None	
	9			☒	☐	00 00	0	100.00	None	
	A			☒	☐	00 00	0	100.00	None	
	B			☒	☐	00 00	0	100.00	None	
	C			☒	☐	00 00	0	100.00	None	
	D			☒	☐	00 00	0	100.00	None	
	E			☒	☐	00 00	0	100.00	None	
	F			☒	☐	00 00	0	100.00	None	
	10			☐	☐	00 00	0	100.00	None	

Running Messages Transmitted: 0 Transmission mode

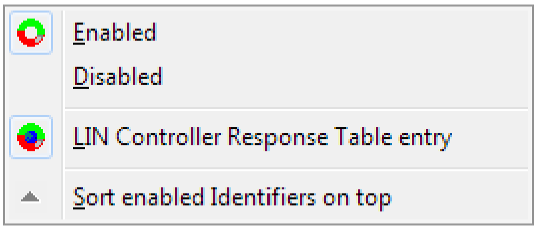
Figure 5.35: Transmit module LIN (Master mode)

entry, they refer to the corresponding transmit table entry (otherwise they would be colored in royal blue). Alas, the data field of an auto response cannot be configured to cyclic changes !

The LIN (Master mode) transmit table (Fig. 5.35) has the following columns:

Column	Meaning
Tx	Icon ● signals an enabled identifier. It is rotating as long as the message's cyclic transmission is active. Icon ● shows that a LIN Response Table entry is enabled which is handled directly by the hardware. It is permanently rotating.
Identifier	Identifier of the transmit object
Message	Name assigned to the identifier according to the .LDF in use
Description	Additional user-defined description of this transmit object. This description allows differentiation of the transmit objects with the same identifier and is only valid in the Transmit module.
ECRC	Defines whether a message is transmitted in enhanded CRC format (LIN 2.0+)
IDO	Defines whether an Identifier only frame is transmitted (Master mode required)
Data	Data of the layer-2 message
Count	Number of transmit repeats; 0 stands for continual transmission
Cycle Time	Cycletime in milliseconds
Inc Mode	Operating mode of cyclic transmission (with/without increment). **None**: No incrementing. **Identifier**: Incrementing of identifier with each transmission. **Byte (Data)**: Incrementing of the databyte defined in the column **Byte** with each transmission. **Word (Data)**: Incrementing of a 16-bit value (compiled from 2 databytes), beginning with the databyte defined in the column **Byte** with each transmission
Byte	Start byte, with which incrementing of the data field is carried out when an increment mode is switched on (see Mode column).

There are different background colors used to illustrate the input rules of a cell:
Light lavender colored cells are for informational purposes only. They are readonly and cannot be selected.



Enabled
Disabled
LIN Controller Response Table entry
Sort enabled Identifiers on top

Figure 5.36: Context menu LIN (full)

The data column is usually highlighted in green, to indicate a fixed data length.
A royal blue colored cell signals that LIN Controller Response Table data is shown in Master mode.

The popup menu (Fig. 5.36) of the LIN transmit table has the following entries:

Menu item	Function
Enabled	Indicates an enabled Response Table entry. Only enabled entries will be auto transmitted by the LIN Controller. For LIN Slave mode only !
Disabled	Indicates a disabled Response Table entry. For LIN Slave mode only !
LIN Controller Response Table entry	Enable Response Table entry. In addition to the manual and cyclic transmission, this LIN identifier will be transmitted automatically by the LIN Controller upon Master request (IDO). For LIN Master mode only !
Sort enabled Identifiers on top	Brings all enabled rows to the top of the transmit table

5.3.5 Creation of transmit objects and Editing the transmit table

Note: This section does not apply to LIN.

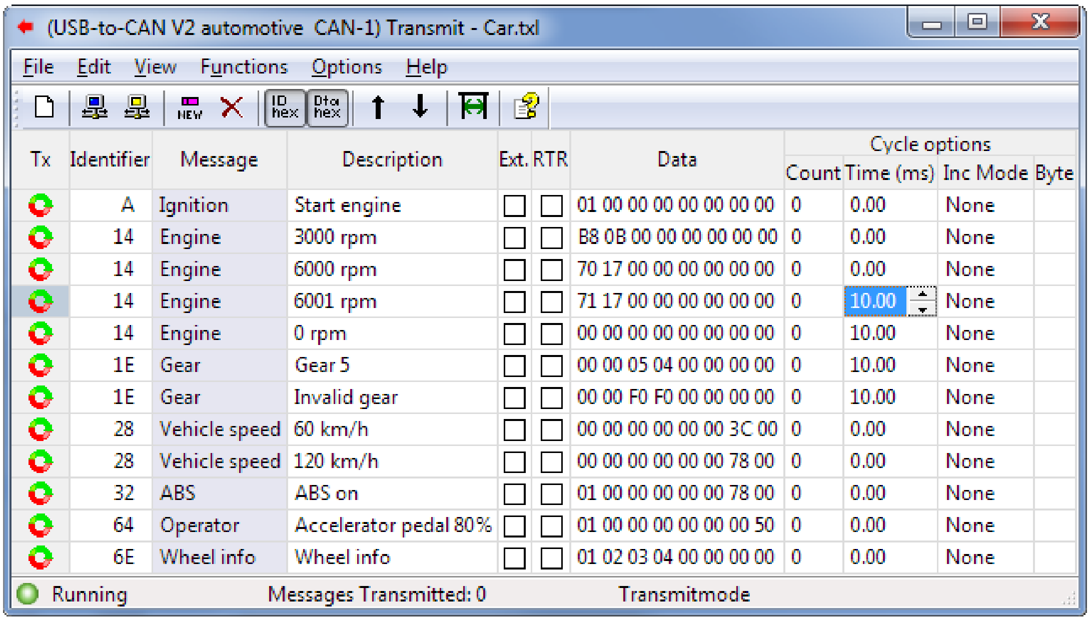
In order to define a new message, a free line is selected in the transmit list. If no free line is available, a new entry is generated via the Menu command **Edit | Insert new Message**. A new entry is also generated when the cursor key is pressed in the last line of the transmit table ↓. The individual columns can be selected with the mouse or with the cursor keys ← and → on the keyboard. A transmit object is defined by entering the identifier, the description and the data to be transmitted.

For editing of the transmit table, the following functions are available under the main menu **Edit**:

Function	Description
Insert new Message	Create a new message
Duplicate Message	Create a copy of the selected message
Delete Message	Delete the selected message

5.3.6 Editing the fields

The editable fields change automatically to edit mode as soon as a numerical or alphanumerical key resp the F2 or the SPACE key is pressed (Fig. 5.37). There is a difference between non-destructive and destructive editing. By pressing F2 or SPACE the cursor will be placed at the



Tx	Identifier	Message	Description	Ext.	RTR	Data	Cycle options			
							Count	Time (ms)	Inc Mode	Byte
●	A	Ignition	Start engine	□	□	01 00 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	3000 rpm	□	□	B8 0B 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	6000 rpm	□	□	70 17 00 00 00 00 00 00	0	0.00	None	
●	14	Engine	6001 rpm	□	□	71 17 00 00 00 00 00 00	0	10.00	None	
●	14	Engine	0 rpm	□	□	00 00 00 00 00 00 00 00	0	10.00	None	
●	1E	Gear	Gear 5	□	□	00 00 05 04 00 00 00 00	0	10.00	None	
●	1E	Gear	Invalid gear	□	□	00 00 F0 F0 00 00 00 00	0	10.00	None	
●	28	Vehicle speed	60 km/h	□	□	00 00 00 00 00 00 3C 00	0	0.00	None	
●	28	Vehicle speed	120 km/h	□	□	00 00 00 00 00 00 78 00	0	0.00	None	
●	32	ABS	ABS on	□	□	01 00 00 00 00 00 78 00	0	0.00	None	
●	64	Operator	Accelerator pedal 80%	□	□	01 00 00 00 00 00 00 50	0	0.00	None	
●	6E	Wheel info	Wheel info	□	□	01 02 03 04 00 00 00 00	0	0.00	None	

Running Messages Transmitted: 0 Transmitmode

Figure 5.37: Edit mode of the cells

end of the field keeping the present values, whilst simply starting to type at an editable field will overwrite the current contents. In either case, the editing can be aborted pressing the ESC key. Editing is finished by pressing the ENTER key, or by clicking on another cell of the transmit table. Readonly fields are identified by a different background color (lilac).

5.3.7 Manual transmission

Individual messages from the table are transmitted by selecting the message and triggering the transmit command.
A message is selected by:

- Clicking on the message with the mouse
- Moving the marking bar with the cursor keys ↑ or ↓ on the keyboard.

Once a message is selected, it can be transmitted by:

- Pressing the key F5
- Selecting the menu item **Functions | Transmit Single Message**
- Clicking the **Transmit single message** button in the toolbar
- Clicking with the left mouse button on the transmit icon ● in the first column

The status bar of the Transmit module displays whether a message could be transmitted successfully.

5.3.8 Cyclic transmission

To be able to transmit messages cyclically, values must be entered in the fields **Count** and **Time** of the column **Cycle options**. A cyclic message can be transmitted both cyclically (automatically) and individually (manually).
Cyclic transmission is carried out by:

- Pressing the key F6
- Selecting the menu item **Functions | Transmit Cyclic Message**
- Clicking the **Transmit cyclic message** button in the toolbar
- Holding the Ctrl-key and at the same time clicking with the left mouse button on the transmit icon in the first column
- Holding the Ctrl-key and at the same time clicking with the left mouse button on the **Transmit cyclic message** button in the toolbar to begin cyclic transmission of all messages

As long as the selected message is transmitted cyclically, its icon rotates in the transmit table . When the number of messages specified under **Count** has been transmitted, no further messages of this transmit object are transmitted and the icon stops rotating.
The cyclic transmission of a selected message can be stopped manually by:

- Clicking again on the **Transmit cyclic message** button in the tool bar
- Pressing again the F6 key

5.3.9 Drag-and-Drop

Received CAN messages might be dragged from the Scroll View of any canAnalyser module to the transmit table of the TransmitModule. Upon dropping, a new transmit message will be created preserving all the message attributes like CAN-identifier, -data, RTR and frame format.

5.3.10 Menu reference

File menu

Menu item	Function
New	Creates a new, empty transmit table
Import Transmit Data...	Load transmit table from a file
Export Transmit Data...	Save transmit table to a file
Recent files	Displays the last transmit tables opened
Exit	Exits the Transmit module

Edit menu

Menu item	Function
Insert new Message	Creates a new entry in the transmit table
Duplicate Message	Creates a copy of the selected transmit object below the selected object
Delete Message	Deletes the selected transmit object from the list

View menu

Menu item	Function
ID Hex	Displays the identifier column in hexadecimal or decimal notation
Data Hex	Displays the data column in hexadecimal or decimal notation
Data granulation	Displays the data BYTE-wise or combines two bytes each as a WORD. With WORD display, it is possible to choose between Little Endian (Intel) or Big Endian format (Motorola).
Columns	Shows/hides the columns: Message, Description, Cycle options
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar

Functions menu

Menu item	Function
Transmit Single message	Transmission of a selected individual message
Transmit Cyclic message	Transmission of the selected cyclic message
Move selected line up	Move current line up by one position
Move selected line down	Move current line down by one position
Autosize Columns	Regulates ideal column widths

Options menu

Menu item	Function
Sort enabled Identifiers on top	Brings all enabled rows to the top of the transmit table. An enabled row is one with a icon in the Tx column.
Font...	Opens a dialog to select the font used to display data

Help menu

Menu item	Function
Help Topics	Opens the online help of the Transmit module
About...	Opens the display of the version information of the Transmit module

5.3.11 Toolbar

The most important functions of the Transmit module can also be called via the toolbar (Fig. 5.38).

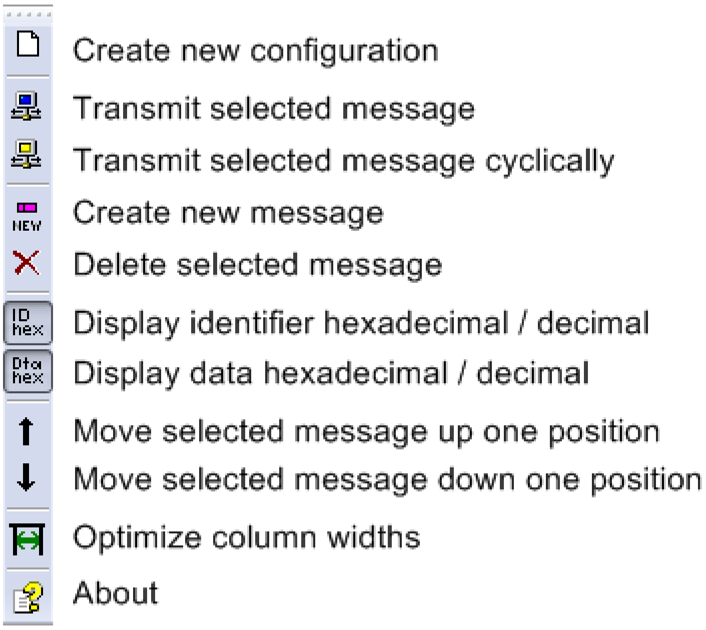

Create new configuration
Transmit selected message
Transmit selected message cyclically
NEW
Create new message
Delete selected message
ID hex
Display identifier hexadecimal / decimal
Dta hex
Display data hexadecimal / decimal
Move selected message up one position
Move selected message down one position
Optimize column widths
About

Figure 5.38: Toolbar of the Transmit module

5.3.12 Hotkeys

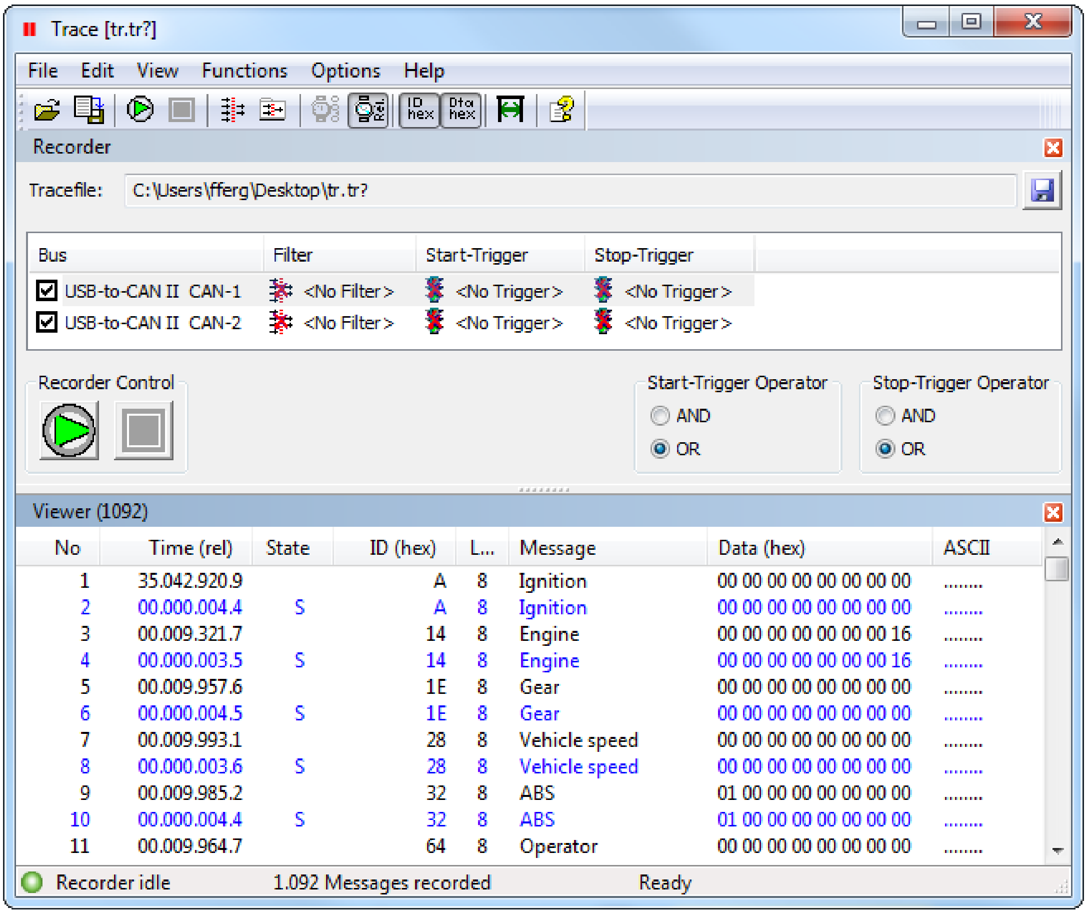
Ctrl+Ins	Insert message
Ctrl+D	Duplicate message
Ctrl+Del	Remove message
F2	Start editing
Space	Start editing RESP (Un)Check a checkbox
F5	Send message
F6	Send cyclic message
Shift+Up	Move row up
Shift+Down	Move row down
Ctrl+Up	Increase Cycle Time by 1
Ctrl+Down	Decrease Cycle Time by 1 RESP Display drop-down list
F1	Online-Help

5.4 Trace module

5.4.1 Overview

The Trace module is used for the simultaneous recording of bus traffic of several buses. It is therefore possible to analyse the message traffic encompassing all buses and after expiry of a complete communication cycle.
Using various triggers, the trace recording can be automatically started and stopped again in order to restrict the trace to certain data and maintain an overview.
The following functionality is provided by the Trace module

- Recording of bus traffic of several buses
- Configuration of various triggers for automatic starting or stopping of trace recording
  - Triggering on a configurable number of received messages
  - Triggering on Identifiers definable via bit-masks and/or databyte values



Tracefile: C:\Users\fferg\Desktop\tr.tr?

Bus	Filter	Start-Trigger	Stop-Trigger
☑ USB-to-CAN II CAN-1	<No Filter>	<No Trigger>	<No Trigger>
☑ USB-to-CAN II CAN-2	<No Filter>	<No Trigger>	<No Trigger>

Start-Trigger Operator: ○ AND ● OR

Stop-Trigger Operator: ○ AND ● OR

No	Time (rel)	State	ID (hex)	L...	Message	Data (hex)	ASCII
1	35.042.920.9		A	8	Ignition	00 00 00 00 00 00 00 00	....
2	00.000.004.4	S	A	8	Ignition	00 00 00 00 00 00 00 00	....
3	00.009.321.7		14	8	Engine	00 00 00 00 00 00 00 16	....
4	00.000.003.5	S	14	8	Engine	00 00 00 00 00 00 00 16	....
5	00.009.957.6		1E	8	Gear	00 00 00 00 00 00 00 00	....
6	00.000.004.5	S	1E	8	Gear	00 00 00 00 00 00 00 00	....
7	00.009.993.1		28	8	Vehicle speed	00 00 00 00 00 00 00 00	....
8	00.000.003.6	S	28	8	Vehicle speed	00 00 00 00 00 00 00 00	....
9	00.009.985.2		32	8	ABS	01 00 00 00 00 00 00 00	....
10	00.000.004.4	S	32	8	ABS	01 00 00 00 00 00 00 00	....
11	00.009.964.7		64	8	Operator	00 00 00 00 00 00 00 00	....

Recorder idle   1.092 Messages recorded   Ready

Figure 5.39: Trace module

  - Triggering on CAN remote and/or error frames

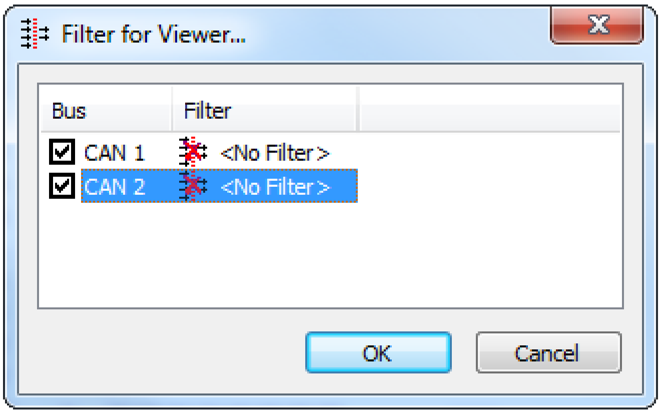
- Linking of the triggers of the buses to be recorded

5.4.2 Program window

The program window of the Trace module (Fig. 5.39) shows two views:

View	Use
Recorder	Setting of parameters for trace recording (selection of the buses/controllers to be recorded, configuration of the filters and trigger events)
Viewer	Displays the recorded (and not yet filtered out) messages in the order in which they were received. This corresponds to the scroll mode of the Receive module. After ending a trace recording, the display changes automatically to this view.

In the **Recorder** view (Fig. 5.39 top) first the file name is to be selected for each trace recording under which the recording is to be saved. For this, a file name with a complete path must be entered in the field **Tracefile**. By clicking with the mouse on the **floppy disk** icon, a dialog is displayed to define the location where the name file is to be saved.
In the field **Bus Configuration**, the buses to be recorded, possible pre-filtering and trigger events are to be defined.
A completed trace recording is output in the Trace module in the **Viewer** (Fig. 5.39 bottom). The recorded telegrams are displayed in order of the time they were received and marked in color



Bus	Filter
☑ CAN 1	<No Filter>
☑ CAN 2	<No Filter>

Figure 5.40: Viewer filter

according to the bus from which they originate. The color of the marking is assigned in the menu **Options | Viewer options**.

The number of visible telegrams can be reduced via the menu command **Functions | Viewer filter...** (Fig. 5.40) in the filter dialog by filtering messages or hiding a complete bus/controller.

5.4.3 Trace recording

The Trace module records the bus traffic of several buses simultaneously. Therefore this module is not assigned to an individual bus/controller, but with higher order to all buses/controllers in the current configuration. The bus traffic is saved by the PC online onto the hard disk and the trace recording is carried out for all bus systems in separate files. After a trace is completed, the messages or all recorded buses are merged and then displayed in relation to each other in terms of time.

After the compulsory definition of the location for saving the trace file(s), the trace recording can be started with the button **Start**. If no start trigger is set, the Trace module begins immediately with the recording to the hard disk. Without a configured stop trigger, the recording must be stopped again manually with the button **Stop**.

Alternatively, buses/controllers can be excluded from the recording in the column **Bus**. Buses/controllers which are not to be recorded are displayed in gray and have no influence on triggering.

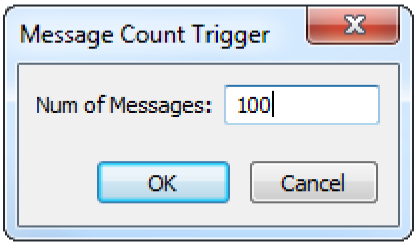
A filter can be selected by clicking on the entry in the column **Filter** of a bus/controller. This reduces the message traffic to be recorded. Please read section 5.9.1 for creating and configuring message filters.

However, for filter configuration it is to be noted that filtering has no influence on triggering. The unfiltered message flow is used to check the trigger conditions, i.e. the trigger is upstream of the filter.

5.4.4 Triggering

The trace recording can be automatically started and stopped by defining start and/or stop trigger events depending on the bus traffic. This means that the Trace module only begins recording after a start trigger event occurs or stops recording automatically after a stop trigger event occurs. When recording is ended, the Trace module automatically switches to the **Viewer**.

Checking of the stop trigger conditions only begins after recording is begun, i.e. after a start trigger event has occurred (if a start trigger was configured).


Message Count Trigger

Num of Messages: 100

Figure 5.41: Message count trigger

The start and stop triggers of the buses/controllers to be recorded are linked to each other via **AND**- or **OR**-operators.

Operator	Function
OR	The trigger event occurs as soon as the trigger condition of one of the start or stop triggers is fulfilled
AND	The trigger event occurs when the trigger conditions of all start or stop triggers are fulfilled

The Trace module provides a selection of several triggers. The triggers are selected and configured via a pop-up menu, which is opened via the right mouse button. The selected trigger can be configured with a double-click.

Trigger	Function
Message Count Trigger	Triggering on a configurable number of received messages
ID/Data Mask Trigger	Triggering on identifiers and/or databyte values, which are specified via bit-masks
CAN RTR/Error Trigger	Triggering on CAN remote and/or error frames

Message count trigger

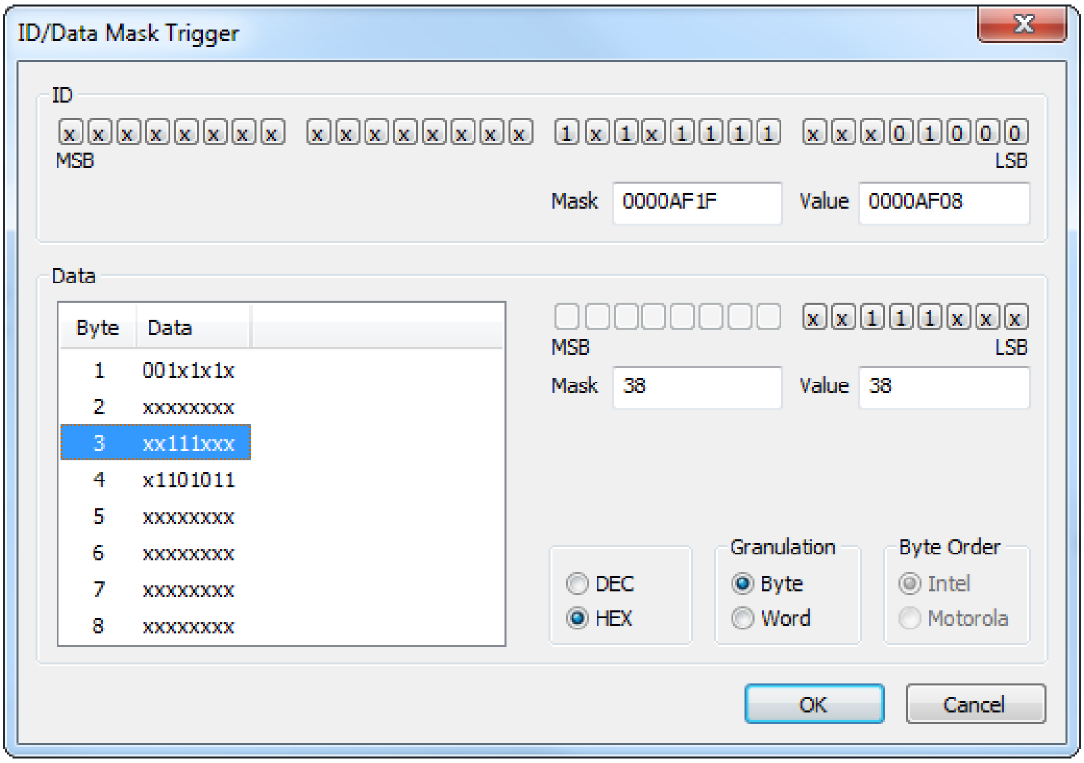
The trigger condition of the message count trigger (Fig. 5.41) is fulfilled after reception of the configured number of messages, i.e. as a stop trigger, the message count trigger defines the number of messages to be recorded.

ID/Data mask trigger

With the ID/Data mask trigger (Fig. 5.42), triggering on certain identifiers and/or databytes is possible. The trigger condition is defined via bit-masks, which are later compared with each receive message. The trigger condition of the individual bits can be altered by clicking with the left mouse button.

Bit	Meaning
0	Bits marked with 0 must have the value 0 in order that the trigger condition is fulfilled
1	Bits marked with 1 must have the value 1 in order that the trigger condition is fulfilled
x	Bits marked with x are not relevant for the trigger condition and are not fulfilled

The bit-masks of the trigger conditions can also be altered manually via the input fields **Mask** and **Value** . In **Mask** the bits with the value 1 are marked as relevant for the trigger condition. In **Value** these relevant bits receive their nominal value (0 or 1). The input/display of the input fields can be made according to the settings in the box **HEX/DEC** in hexadecimal or decimal form.



ID/Data Mask Trigger

ID

x x x x x x x x x x x x x x x x 1 x 1 x 1 1 1 1 x x x 0 1 0 0 0

MSB … LSB

Mask 0000AF1F Value 0000AF08

Data

Byte	Data
1	001x1x1x
2	xxxxxxxx
3	xx111xxx
4	x1101011
5	xxxxxxxx
6	xxxxxxxx
7	xxxxxxxx
8	xxxxxxxx

□ □ □ □ □ □ □ □ x x 1 1 1 x x x

MSB … LSB

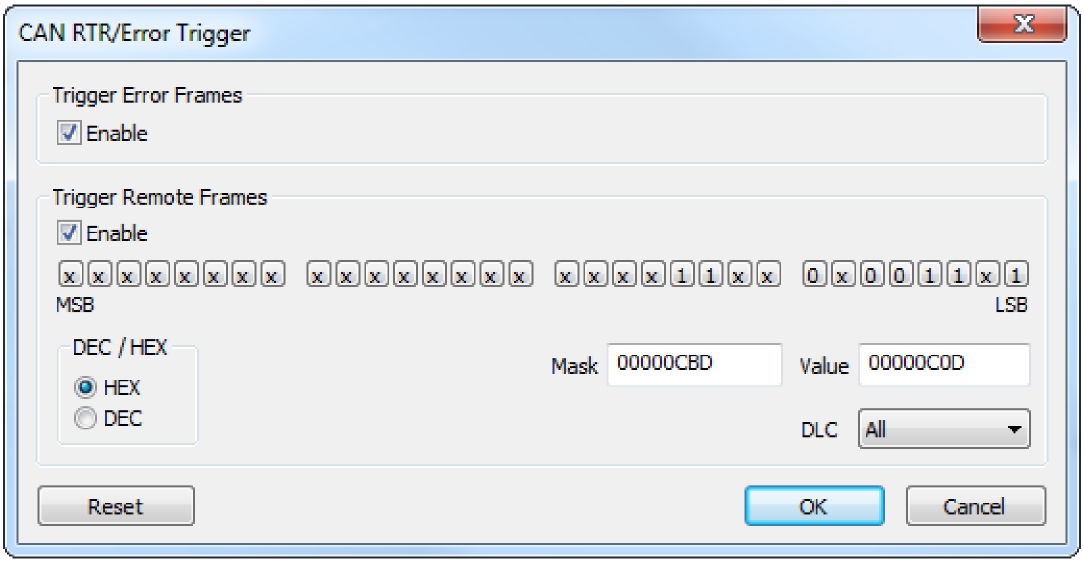
Mask 38 Value 38

○ DEC ● HEX

Granulation: ● Byte ○ Word

Byte Order: ● Intel ○ Motorola

Figure 5.42: ID/Data mask trigger



CAN RTR/Error Trigger

Trigger Error Frames

☑ Enable

Trigger Remote Frames

☑ Enable

x x x x x x x x x x x x x x x x x x x x 1 1 x x 0 x 0 0 1 1 x 1

MSB … LSB

DEC / HEX: ● HEX ○ DEC

Mask 00000CBD Value 00000C0D

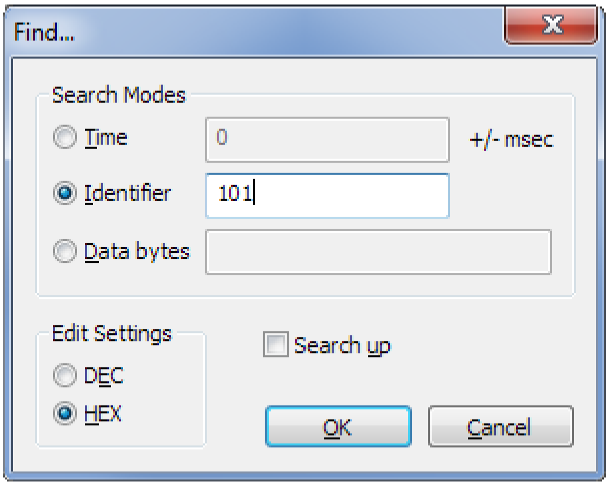
DLC All

Figure 5.43: CAN RTR/error trigger

The buttons **Byte** and **Word** define the granulation of the data field. If **Word** granulation is selected, **Intel** (Little Endian) or **Motorola** (Big Endian) format can be set for the byte order.
The trigger condition is fulfilled when all bits marked as relevant have the nominal value defined for them. However, this also means that a trigger condition in which all bits are marked as not relevant is fulfilled for every telegram.

CAN RTR/error trigger

With the CAN RTR/error trigger (Fig. 5.43), it is possible to trigger on CAN remote frames and/or CAN error frames. Triggering on the relevant frame type can be switched on or off via the



Find...

Search Modes

○ Time 0 +/- msec

● Identifier 101

○ Data bytes

Edit Settings

○ DEC

● HEX

□ Search up

Figure 5.44: Find Dialog: Searching the Viewer

corresponding button **Enable**.
The remote identifier is defined via bit-masks (see Section 5.4.4) and via the field **DLC** the data length. With the setting **All**, the data length is not considered when checking the trigger condition. If neither **Trigger Error Frames** nor **Trigger Remote Frames** is activated, the trigger condition is fulfilled for every received CAN data telegram.

5.4.5 Find Dialog

The Find dialog (Abb. 5.44) offers three different ways of searching for a particular message in the merged trace recording shown in the Viewer:

- **Search mode Time**

  Finds the closest time stamp to given time lag. It's a search for a message relative to the currently marked (absolute) time stamp. The input is entered in milliseconds. If the sign is positive (or omitted), a later time stamp is looked for, in case of a negative sign, an earlier one.

- **Search mode Identifier**

  Finds a particular Identifier, which can be entered in hexadecimal or decimal format.

- **Search mode Data bytes**

  Finds a particular byte pattern in the data field of the messages. One up to sixteen consecutive byte values separated by spaces can be entered in decimal or hexadecimal format. The absolute position of the byte pattern in the data field is ignored while doing so, i.e. the given byte pattern will be located anywhere in the data field. Wildcards or placeholders are not supported.

Searching follows the chronological sequence of the messages, that is from top to bottom of the Viewer. It starts at the currently marked line, not at the beginning of the recording, nor in the currently visible/displayed part of it. Checkbox **Search up** will revert the search direction, i.e. search upwards from currently marked line.
Since the search process can take some time, especially when large trace recordings are worked up, it can be aborted as usual with the corresponding button in the status bar.
The search options are not persistent, they last only for the current canAnalyser session.
If nothing was found, a messagebox will be shown.

5.4.6 Further processing of a trace recording

A saved trace recording can be imported into the Trace module for another analysis via the menu item **File | Open Trace...**.
In the Replay module, a trace recording can be played back again. This allows to analyse the recording in the different canAnalyser modules any number of times.
The Sequencer module also provides the import of trace files. There, the CAN telegrams of the trace file are converted to transmit commands of the Sequencer module.
If a trace recording is to be further processed in another program, the currently displayed trace recording can be exported to a text file. The export is carried out via the menu entry **File | Export Messages...** or via the button in the toolbar. The generated text file contains all columns of the Trace module. By separating the columns with commas or semi-colons, the ASCII trace file is easily imported as a CSV (Comma Separated Value) file in standard tools, such as Excel, and processed further.

5.4.7 Menu reference

File menu

Menu item	Function
Open Trace...	Opens a previously saved trace recording
Close Trace	Closes the currently open trace recording and clears the Viewer
Export Messages...	Exports the displayed trace recording to a text file
Exit	Exits the Trace module

Edit menu

Menu item	Function
Find...	Opens a dialog field to define the search options for a certain message in a trace file
Find Next	Shows the next message that corresponds to the search options
Copy CSV	Copies marked lines CSV formatted to clipboard
Toggle Marker	Sets or Removes a Marker for selected message
Previous Marker	Jumps to previous Marker (no wraparound)
Next Marker	Jumps to next Marker (no wraparound)
Set/Release Time Reference	Sets Timestamp Zero for selected message / Releases previously set Timestamp Zero
Jump to Time Reference	Jumps to previously set Timestamp Zero message

View menu

Menu item	Function
Time rel.	Displays the time in the Viewer absolute or relative to the previously displayed message
ID hex	Displays the message identifiers in the Viewer in hexadecimal or decimal notation
ID representation	Representation options of the identifier column in the Viewer
Data hex	Displays the message data in the Viewer in hexadecimal or decimal notation
Data representation	Representation options of the data column in the Viewer
Data granulation	Displays the data in the Viewer BYTE-wise or combines two bytes each as a WORD. With WORD display, it is possible to choose between Little Endian (Intel) or Big Endian format (Motorola).
Word wrap lines	Displays the data in the Viewer over several lines if the specified column width is not sufficient for the display of the complete data field.
Recorder View	Shows/hides the Recorder view
Viewer View	Shows/hides the Viewer view
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar

Functions menu

Menu item	Function
Start recordging	Begins a trace recording
Stop recording	Ends a trace recording
Available Filters...	Adjust application wide available message filters (see section 5.9.1)
Viewer Filter...	Opens a dialog to filter the message recording
Autosize Columns	Regulates ideal column widths

Options menu

Menu item	Function
Viewer Options...	Opens a dialog to select the colors in which the messages of the various buses are displayed in the Viewer
Fonts...	Opens a dialog to select the font type in which the messages are displayed in the Viewer

Help menu

Menu item	Function
Help Topics	Opens the online help of the Trace module
About...	Opens the display of the version information of the Trace module

5.4.8 Toolbar

The most important functions of the Trace module can also be called via the toolbar (Fig. 5.45).

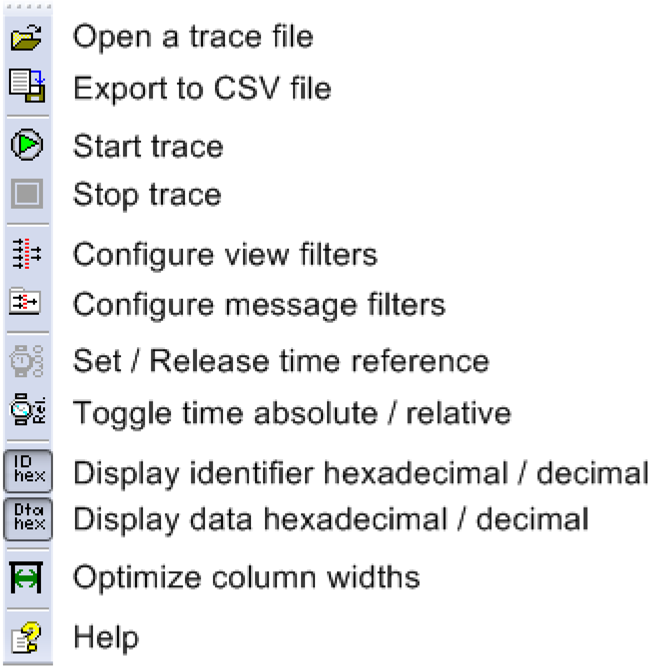

Open a trace file
Export to CSV file
Start trace
Stop trace
Configure view filters
Configure message filters
Set / Release time reference
Toggle time absolute / relative
Display identifier hexadecimal / decimal
Display data hexadecimal / decimal
Optimize column widths
Help

Figure 5.45: Toolbar of the Trace module

5.4.9 Hotkeys

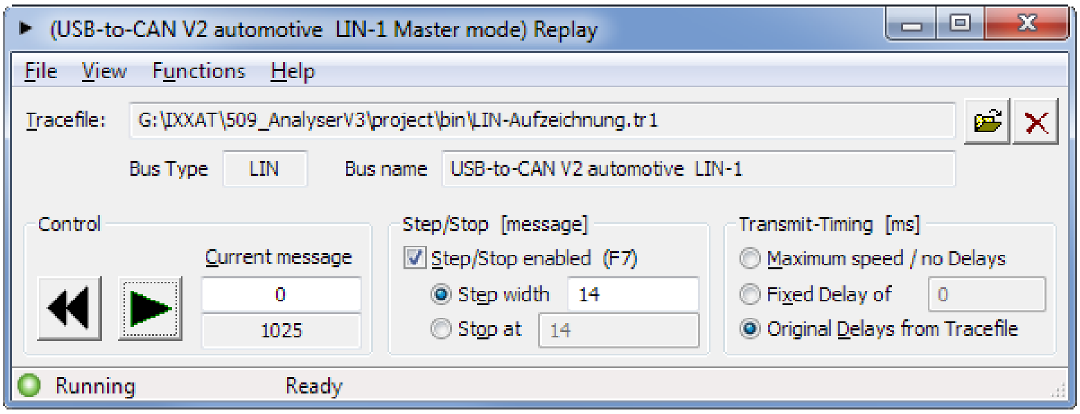
Ctrl+O	Open an existing trace recording
Ctrl+E	Export the received messages to a file
Ctrl+F	Open the search dialog
F3	Search next
Ctrl+C	Copy marked lines CSV formatted to clipboard
Ctrl+F2	Toggle Marker in Trace View
Shift+F2	Go to Previous Marker in Trace View
F2	Go to Next Marker in Trace View
Ctrl+0	Jump to Time Reference message in Trace View
F5	Start message reception
Shift+F5	Stop message reception
Ctrl+I	Configure Filter
F1	Online-Help
Ctrl+TAB	Switch between Record View and Trace View
PageUp	Scroll one page backward in Trace View
PageDown	Scroll one page ahead in current Trace View
Ctrl+PageUp	Scroll 1000 messages backward in Trace View
Ctrl+PageDown	Scroll 1000 messages ahead in Trace View
Ctrl+1..9	Jump to 10%..90% of Trace View

5.5 Replay module

5.5.1 Overview

You can use the Replay module to replay trace files. It depends on the controller state how the replayed messages are handled:

- If the controller is online the messages will be sent to the bus, and after that received via selfreception. The timestamp of the messages is set to the receive time.



(USB-to-CAN V2 automotive LIN-1 Master mode) Replay

Tracefile: G:\IXXAT\509_AnalyserV3\project\bin\LIN-Aufzeichnung.tr1

Bus Type LIN Bus name USB-to-CAN V2 automotive LIN-1

Control
Current message 0
1025

Step/Stop [message]
☑ Step/Stop enabled (F7)
● Step width 14
○ Stop at 14

Transmit-Timing [ms]
○ Maximum speed / no Delays
○ Fixed Delay of 0
● Original Delays from Tracefile

Running Ready

Figure 5.46: Replay module

- If the controller is in offline mode the messages will be distributed to the connected modules. In this mode the timestamps from the tracefile will be used.

5.5.2 Replay window

The Replay module has several options in the replay-window (Fig. 5.46) which influences the replay.
In the upper section you can see the trace filename along with some information from the tracefile.
In the **Speed** section you can adjust the replay speed. The following options are possible:

- Maximum speed
- Constant delay between messages
- Calculate delay from the timestamps stored in the tracefile

With the **Step/Stop enabled** option you are able to do a stepwise replay with adjustable step size.
The **Stop at** option may be used to stop the replay at a given message index.
In the **Control** section there is an input field, which contains the current message index. You may edit this field manually to start the replay at a specific position.
The two buttons on the left are for rewind or start replay.

5.5.3 Menu reference

File menu

Menu item	Function
Open Trace...	Opens a trace file
Close Trace	Close the trace file
Exit	Exits the Replay module

View menu

Menu item	Function
Status Bar	Shows/hides the status bar

Help menu

Menu item	Function
Help Topics	Opens the online help of the Replay module
About...	Opens the display of the version information of the Replay module

5.5.4 Hotkeys

Ctrl+O	Open trace
F5	Play trace
Shift+F5	Stop playback
F4	Rewind
F7	Stepwise playback
F1	Online-Help

5.6 Signal module

5.6.1 Overview

The Receive and the Trace module of the canAnalyser display the messages transmitted in a fieldbus system with their layer-2 information (identifier and data). With an interpretation of the transmitted data and their display in plaintext, the analysis of a system becomes much more transparent.

With the Signal module (Fig. 5.47) it is possible to display received layer-2 message contents in interpreted form. This analysis module is therefore particularly suitable for the installation, testing and maintenance of fieldbus systems, as it enables easy handling of physical and logical parameters.

The interpreted signals can be monitored in terms of changes in their values and observance of the cycle time defined in the database. Also, the compliance with the defined range values of a signal will be checked, and a violation of the limits is shown both by text color and status icon.

Prior to working with the Signal module, a project database together with its desired channel needs to be assigned to the Virtual Bus in the Control Panel.

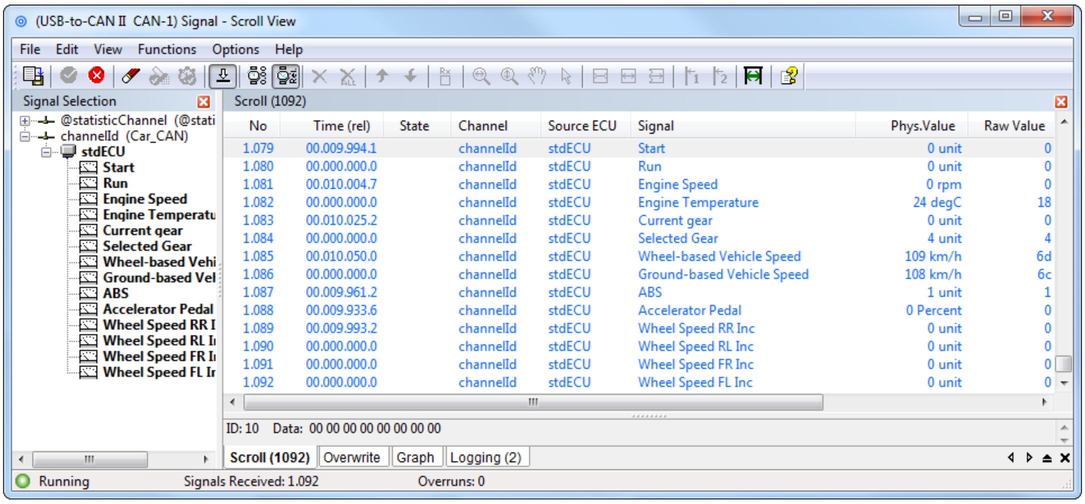
The main window of the Signal module is divided into two halves: On the left side, there is the pivotal Signal Selection Tree that shows the structure of the project database in a hierarchical order. The major part of the window is used by the single views of the received messages right to it:

- Scroll View
- Overwrite View
- Graph View
- Logging View

Each of the views can be configured in such a way that it only receives certain signals. This is achieved by the pivotal **Signal Selection Tree**.

The **Logging View** represents a specialised form of the Scroll View which at this time lists range violation events.

As various instances of the Signal module can be started by the control panel, each Signal module can be adapted individually to the signals or signal groups to be analysed.



No	Time (rel)	State	Channel	Source ECU	Signal	Phys.Value	Raw Value
1.079	00.009.994.1		channelId	stdECU	Start	0 unit	0
1.080	00.000.000.0		channelId	stdECU	Run	0 unit	0
1.081	00.010.004.7		channelId	stdECU	Engine Speed	0 rpm	0
1.082	00.000.000.0		channelId	stdECU	Engine Temperature	24 degC	18
1.083	00.010.025.2		channelId	stdECU	Current gear	0 unit	0
1.084	00.000.000.0		channelId	stdECU	Selected Gear	4 unit	4
1.085	00.010.050.0		channelId	stdECU	Wheel-based Vehicle Speed	109 km/h	6d
1.086	00.000.000.0		channelId	stdECU	Ground-based Vehicle Speed	108 km/h	6c
1.087	00.009.961.2		channelId	stdECU	ABS	1 unit	1
1.088	00.009.933.6		channelId	stdECU	Accelerator Pedal	0 Percent	0
1.089	00.009.993.2		channelId	stdECU	Wheel Speed RR Inc	0 unit	0
1.090	00.000.000.0		channelId	stdECU	Wheel Speed RL Inc	0 unit	0
1.091	00.000.000.0		channelId	stdECU	Wheel Speed FR Inc	0 unit	0
1.092	00.000.000.0		channelId	stdECU	Wheel Speed FL Inc	0 unit	0

ID: 10 Data: 00 00 00 00 00 00 00 00

Running Signals Received: 1.092 Overruns: 0

Figure 5.47: Signal module Scroll View

5.6.2 Project Databases

The interpretation and symbolic display of the data transmitted in layer-2 messages is based on a project database. This is created by the user with the aid of a Database editor. The Database editor is part of the canAnalyser. It is being launched by the Control Panel.
In the project database, every layer-2 message can be assigned a symbolic name. Within a message, individual data (signals) can be defined. Each signal is defined by a bit position and length in the layer-2 message and can contain analog information, status information or a string:

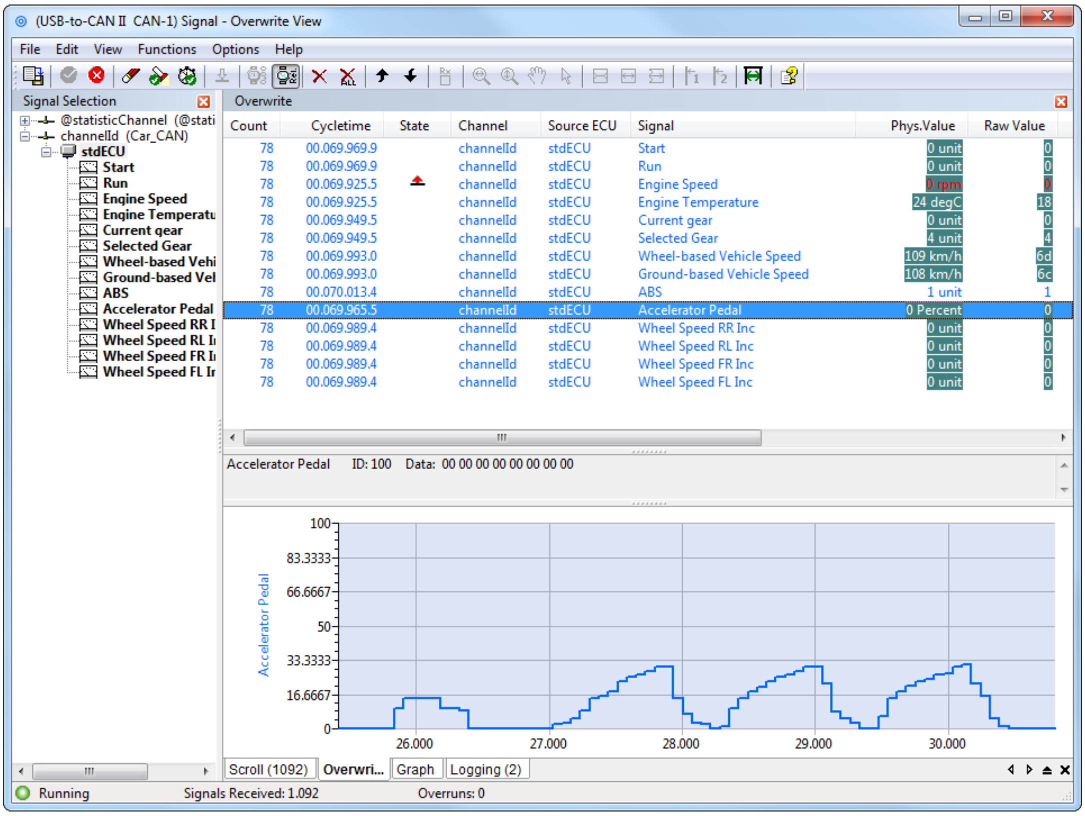
- Analog signals are described by their data type and data format (Intel/Motorola), scaling, offset, value range and physical unit
- Individual values of status signals can be assigned symbolic names, which are displayed as text for interpretation (statuses)
- With string signals, part of the layer-2 message is interpreted as an ASCII string

Important: The frames are defined in the interpretation database with a fixed data length. Telegrams whose identifiers are defined in the database but whose data length does not correspond to that defined in the data base are not interpreted. Frames which are not interpreted are not listed by the Signal module in the Scroll View or are marked gray in the overwrite display.

5.6.3 Scroll View

The **Scroll** page (Fig. 5.47) displays the signals in the order in which they were received.

The individual columns of Scroll View have the following meaning:



Count	Cycletime	State	Channel	Source ECU	Signal	Phys.Value	Raw Value
78	00.069.969.9		channelId	stdECU	Start	0 unit	0
78	00.069.969.9		channelId	stdECU	Run	0 unit	0
78	00.069.925.5	▲	channelId	stdECU	Engine Speed	0 rpm	0
78	00.069.925.5		channelId	stdECU	Engine Temperature	24 degC	18
78	00.069.949.5		channelId	stdECU	Current gear	0 unit	0
78	00.069.949.5		channelId	stdECU	Selected Gear	4 unit	4
78	00.069.993.0		channelId	stdECU	Wheel-based Vehicle Speed	109 km/h	6d
78	00.069.993.0		channelId	stdECU	Ground-based Vehicle Speed	108 km/h	6c
78	00.070.013.4		channelId	stdECU	ABS	1 unit	1
78	00.069.965.5		channelId	stdECU	Accelerator Pedal	0 Percent	0
78	00.069.989.4		channelId	stdECU	Wheel Speed RR Inc	0 unit	0
78	00.069.989.4		channelId	stdECU	Wheel Speed RL Inc	0 unit	0
78	00.069.989.4		channelId	stdECU	Wheel Speed FR Inc	0 unit	0
78	00.069.989.4		channelId	stdECU	Wheel Speed FL Inc	0 unit	0

Accelerator Pedal ID: 100 Data: 00 00 00 00 00 00 00 00

Running Signals Received: 1.092 Overruns: 0

Figure 5.48: Signal module Overwrite View

Column	Meaning
No	Consecutive number of the received object
Time (abs/rel)	Time stamp of reception, either absolute (related to the start time of the controller) or relative to the message previously received
State	If messages have been lost, this is indicated here by the symbol . If signal values are outside their defined range, this is indicated here by the symbols ▼ and ▲
Channel	Name of the physical bus etc. according to project database. It is being chosen together with the project database in the Control Panel. ()
Source ECU	Origin of the frame ()
Frame (long)	Frame name in the database ()
Signal (long)	Signal name in the database ()
Phys.Value	Value of the signal and physical unit where applicable
Raw Value	Uninterpreted value extracted from frame
Lower Limit	Signal value's lower limit
Upper Limit	Signal value's upper limit

5.6.4 Overwrite View

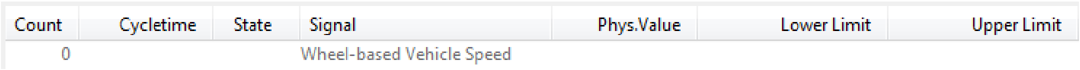
The most recent values of selected signals are displayed in each case on the page **Overwrite** (Fig. 5.48). For this, the signals to be displayed have to be selected first in Signal Selection Tree by double-clicking or via menu command. In addition the overwrite view can be configured to a user-defined display order of the signals.

The buttons of the toolbar for display configuration ✕ ✕ALL ↑ ↓ are used to add the relevant marked entry to the view, to delete it, to move it up or down in the list or to remove all entries from the list.

The data display of the Overwrite View has the following columns:

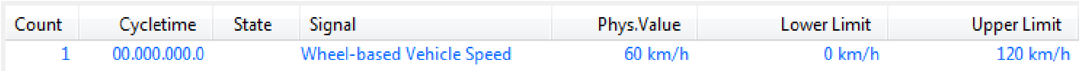
Column	Meaning
Count	Number of received messages
Cycletime / Time (abs)	Either last cycle time of the signal or absolute time stamp of the last reception related to the start time; by right-clicking on the column heading the display of hours and minutes can be switched on or off
State	Receive faults are displayed with icons: • Messages have been lost and displayed data may therefore be false • Messages have been lost. The number displayed in the column **Number** may therefore be too small • A timeout of the cycle time specified in the database has occurred • ▼ A received value has been below its allowed value range limit • ▲ A received value has been above its allowed value range limit
Channel	Name of the physical bus etc. according to project database. It is being chosen together with the project database in the Control Panel. ()
Source ECU	Origin of the frame ()
Frame (long)	Frame name in the database ()
Signal (long)	Signal name in the database ()
Phys.Value	Value of the signal and physical unit where applicable
Raw Value (hex)	Uninterpreted value extracted from frame, optionally shown in hexadecimal representation
Lower Limit	Signal value's lower limit
Upper Limit	Signal value's upper limit
Min Cycletime	Smallest measured cycle time of the Signal
Max Cycletime	Biggest measured cycle time of the Signal

For every signal entry in the overwrite view there is a signal value history. When you select an entry it's value history is shown in the overwrite graph window as a line graph. The overwrite graph window is used in the same way as the **Graph View**.
Aside from that, it is possible to display a second signal value history graph for comparison. For that to happen, left click onto the signal to compare while holding down the Ctrl key (Ctrl+click).
In the graph displays of the signal value history the valid value range of a signal is highlighted in color. Ideally, the signal curve will never leave that area, so the highlight coloring might not be noted by the user at all.



Count	Cycletime	State	Signal	Phys.Value	Lower Limit	Upper Limit
0			Wheel-based Vehicle Speed			

Figure 5.49: Signal added to the Overwrite view



Count	Cycletime	State	Signal	Phys.Value	Lower Limit	Upper Limit
1	00.000.000.0		Wheel-based Vehicle Speed	60 km/h	0 km/h	120 km/h

Figure 5.50: First reception of a signal in the Overwrite view

5.6.5 Change Detection

The change detection takes place in the **Overwrite View** of the signal module and can be activated and deactivated via the menu command **Options | Value Change Detection**. There are change detectors implemented that check against value changes and value range violations. Another check monitors the expected signal cycle time. For the latter checks, too, there is a corresponding menu command for enabling or disabling the feature, at the same place.

Once a deviation is detected, a corresponding icon appears in the state column, staying present until it is manually reset or the view is cleared by the user.

After configuration of a signal in the Overwrite View of the Signal module, the signal appears in gray font color before first reception of the frame (Fig. 5.49).
On first reception, the signal is displayed in blue font in the overwrite view and with the received value. No **Cycletime** is displayed yet, as for the calculation of this the signal must have been received at least twice (Fig. 5.50).
After the second reception of the signals, the changed value compared with the first reception is highlighted in color in the column **Phys.Value** if the data change detection is switched on (Fig. 5.51).

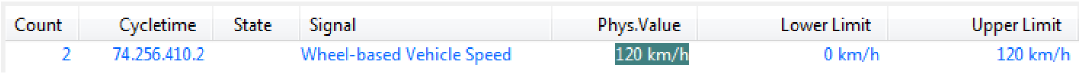
The change detections and the signalling of timeouts can be reset in the context menu of the display area. For this, click with the right mouse button on the signal of the change detection that is to be reset. In the context menu that appears, the display for the selected signal or for all signals can be reset. (Fig. 5.52).

The data change detection always occurs in relation to the signal value of the first reception of a signal or in relation to the signal value at the time of resetting of the data change detection. Therefore, if a different value was already received for a signal than the reference value (first received value or signal value when resetting), then the signal value remains highlighted, even if a signal value identical to the reference value is received again.

5.6.6 Logging View

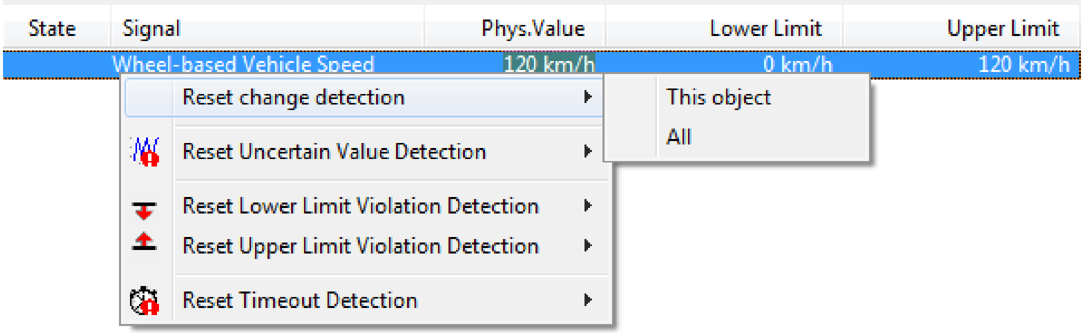
The **Logging** page (Fig. 5.53) displays signal value range violation events in the order in which they occured. This includes the times when the value exceeded its allowed range upwards (received value too high) resp. downwards (received value too low), and the times when the value returned to its normal range.

The individual columns in Logging View have the following meaning:



Count	Cycletime	State	Signal	Phys.Value	Lower Limit	Upper Limit
2	74.256.410.2		Wheel-based Vehicle Speed	120 km/h	0 km/h	120 km/h

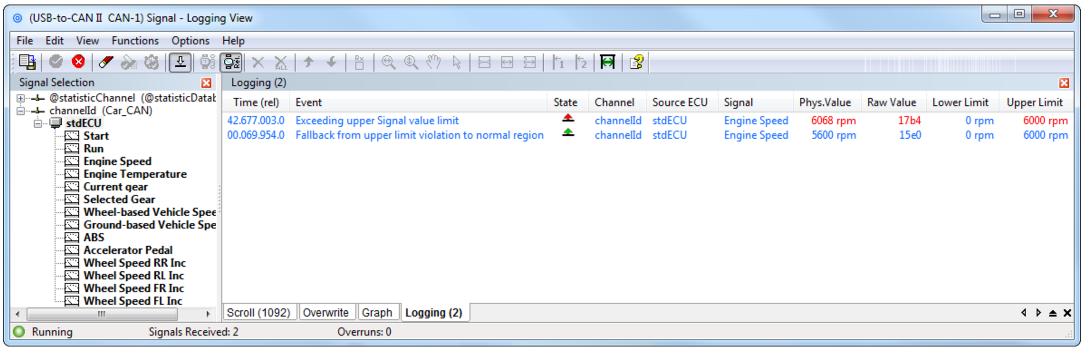
Figure 5.51: Second reception of the signal with changed value



State	Signal	Phys.Value	Lower Limit	Upper Limit
	Wheel-based Vehicle Speed	120 km/h	0 km/h	120 km/h

Reset change detection
Reset Uncertain Value Detection
Reset Lower Limit Violation Detection
Reset Upper Limit Violation Detection
Reset Timeout Detection
This object
All

Figure 5.52: Context menu of Overwrite View



Time (rel)	Event	State	Channel	Source ECU	Signal	Phys.Value	Raw Value	Lower Limit	Upper Limit
42.677.003.0	Exceeding upper Signal value limit	▲	channelId	stdECU	Engine Speed	6068 rpm	17b4	0 rpm	6000 rpm
00.069.954.0	Fallback from upper limit violation to normal region	▲	channelId	stdECU	Engine Speed	5600 rpm	15e0	0 rpm	6000 rpm

Figure 5.53: Signal module Logging View

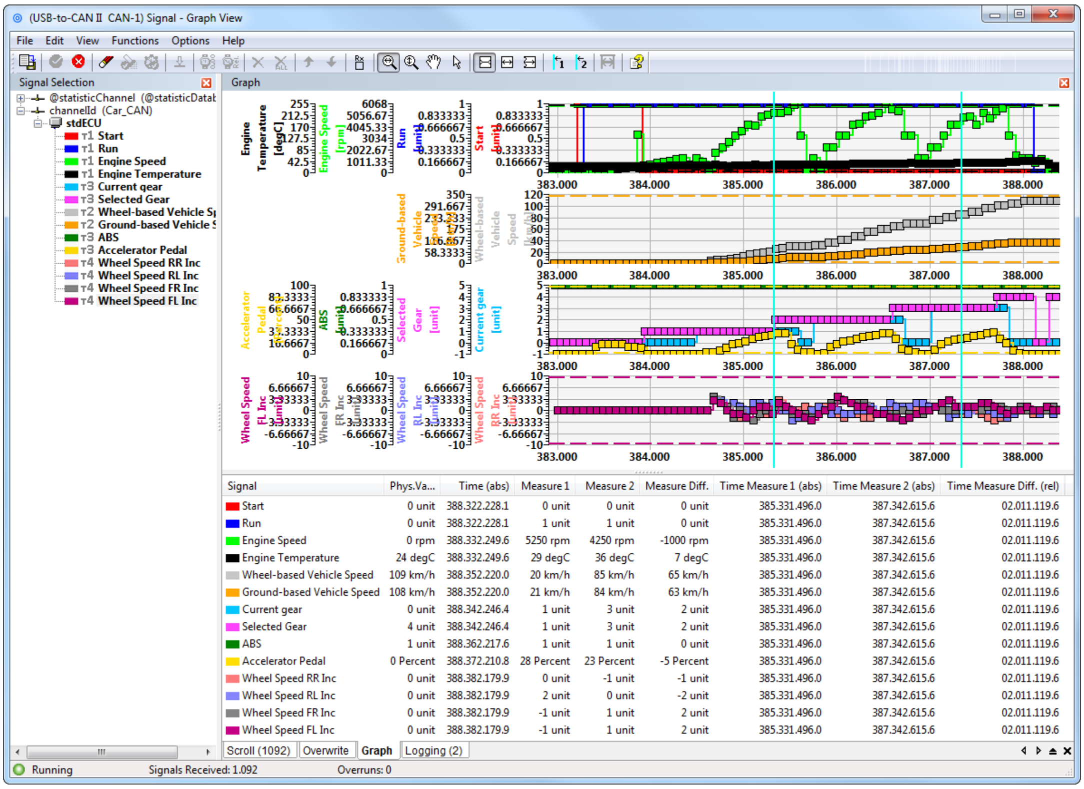
Column	Meaning
Time (abs/rel)	Time stamp of occurence, either absolute (related to the start time of the controller) or relative to the log event previously occured
Event	Description of the event
State	If signal values turned outside their defined range, this is indicated here by the symbols ▼ and ▲. If signal values fell back to their defined range, this is indicated here by the symbols ▼ and ▲
Channel	Name of the physical bus etc. according to project database. It is being chosen together with the project database in the Control Panel. (⊥)
Source ECU	Origin of the frame (🖥)
Frame (long)	Frame name in the database (▭)
Signal (long)	Signal name in the database (▨)
Phys.Value	Value of the signal and physical unit where applicable
Raw Value	Uninterpreted value extracted from frame
Lower Limit	Signal value's lower limit
Upper Limit	Signal value's upper limit

5.6.7 Graph View

The Graph View can display up to 15 signals simultaneously in up to 4 coordinate systems (Abb. 5.54).

Different operating modes are available to adjust the time axis while receiving signal values:

- Off - the time axis will not be adjusted. After manual zooming with the mouse this mode is automatically selected.



Signal	Phys.Va...	Time (abs)	Measure 1	Measure 2	Measure Diff.	Time Measure 1 (abs)	Time Measure 2 (abs)	Time Measure Diff. (rel)
🟥 Start	0 unit	388.322.228.1	0 unit	0 unit	0 unit	385.331.496.0	387.342.615.6	02.011.119.6
🟦 Run	0 unit	388.322.228.1	1 unit	1 unit	0 unit	385.331.496.0	387.342.615.6	02.011.119.6
🟩 Engine Speed	0 rpm	388.332.249.6	5250 rpm	4250 rpm	-1000 rpm	385.331.496.0	387.342.615.6	02.011.119.6
⬛ Engine Temperature	24 degC	388.332.249.6	29 degC	36 degC	7 degC	385.331.496.0	387.342.615.6	02.011.119.6
⬜ Wheel-based Vehicle Speed	109 km/h	388.352.220.0	20 km/h	85 km/h	65 km/h	385.331.496.0	387.342.615.6	02.011.119.6
🟧 Ground-based Vehicle Speed	108 km/h	388.352.220.0	21 km/h	84 km/h	63 km/h	385.331.496.0	387.342.615.6	02.011.119.6
🟦 Current gear	0 unit	388.342.246.4	1 unit	3 unit	2 unit	385.331.496.0	387.342.615.6	02.011.119.6
🟪 Selected Gear	4 unit	388.342.246.4	1 unit	3 unit	2 unit	385.331.496.0	387.342.615.6	02.011.119.6
🟩 ABS	1 unit	388.362.217.6	1 unit	1 unit	0 unit	385.331.496.0	387.342.615.6	02.011.119.6
🟨 Accelerator Pedal	0 Percent	388.372.210.8	28 Percent	23 Percent	-5 Percent	385.331.496.0	387.342.615.6	02.011.119.6
🟥 Wheel Speed RR Inc	0 unit	388.382.179.9	0 unit	-1 unit	-1 unit	385.331.496.0	387.342.615.6	02.011.119.6
🟦 Wheel Speed RL Inc	0 unit	388.382.179.9	2 unit	0 unit	-2 unit	385.331.496.0	387.342.615.6	02.011.119.6
⬛ Wheel Speed FR Inc	0 unit	388.382.179.9	-1 unit	1 unit	2 unit	385.331.496.0	387.342.615.6	02.011.119.6
🟪 Wheel Speed FL Inc	0 unit	388.382.179.9	-1 unit	1 unit	2 unit	385.331.496.0	387.342.615.6	02.011.119.6

Running Signals Received: 1.092 Overruns: 0

Figure 5.54: Signal module showing Graph View

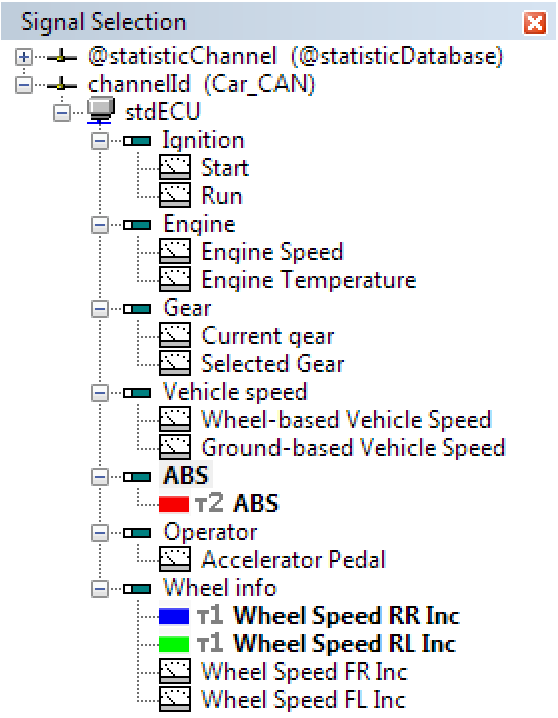
- Range - the visible time axis is automatically set to the range between the minimum and maximum available time stamp. After start this mode is selected.
- Align - The maximum available time stamp is aligned on the right side. The zoom factor stays constant.

Upon signal selection, the graphical display of the received signal values begins. Each coordinate system can be scrolled and zoomed inline.
In the coordinate systems there are additional horizontal dashed lines for visualisation of the valid value range. Ideally, the signal curve always stays in between them.

The current signal values and time stamps are displayed in tabular form and plaintext in the lower half of the window. If no value has yet been received for a signal, this is apparent from the missing time stamp.
After stopping data recording via **Functions** | **Stop**, the recorded data can also be analysed more exactly.
By scrolling and zooming, the time interval to be analysed can be selected. By positioning the cursor with a mouse click on the graphic display, exact signal values can be determined. These values are displayed in the bottom half of the window under "Phys.Value".

The mouse behaviour could be altered by selecting different operating modes:

- Zoom time axis - The mouse could be used to zoom the time axis. By click and click & drag you could zoom in (left mouse button) or zoom out (right mouse button).
- Zoom value axis - The mouse could be used to zoom the value axis. By click and click & drag you could zoom in (left mouse button) or zoom out (right mouse button). This function works on all signal axes of the zoomed track.



Signal Selection

- @statisticChannel (@statisticDatabase)
- channelId (Car_CAN)
  - stdECU
    - Ignition
      - Start
      - Run
    - Engine
      - Engine Speed
      - Engine Temperature
    - Gear
      - Current gear
      - Selected Gear
    - Vehicle speed
      - Wheel-based Vehicle Speed
      - Ground-based Vehicle Speed
    - **ABS**
      - 🟥 T2 **ABS**
    - Operator
      - Accelerator Pedal
    - Wheel info
      - 🟦 T1 **Wheel Speed RR Inc**
      - 🟩 T1 **Wheel Speed RL Inc**
      - Wheel Speed FR Inc
      - Wheel Speed FL Inc

Figure 5.55: Signal Selection Tree of the signal module

- Pan - The mouse could be used to move the visible range of a track. This function works on all signal axes of the moved track and on the common time axis.

If you move the mouse cursor over a signal or time axis a zoom bar will be displayed. This zoom bar allows to alter the visible zoom range. With them it is possible to change the offsets and zoom factors of the signal axes independantly.
Note: Data recording can be resumed with **Functions | Start**. All events between stop and re-start are ignored.
Note: The graphic value variation of a signal marked in the bottom half of the window is displayed in bold.

5.6.8 Signal Selection Tree

The Signal Selection Tree shows all the channels and databases which are assigned to the Virtual Bus as well as the internal statistics database (Abb. 5.55).
The operation is done via keyboard or mouse.
With the Signal Selection Tree, you select the signals to be displayed on each of the four views (Scroll, Overwrite, Logging, Graph) individually. Selected ones are shown **bold**. Upon switching of the active view the Signal Selection Tree adjusts accordingly.

At least three hierarchical levels constitute the Signal Selection Tree:

- **Channel** Topmost level of the tree, symbolised by ⊥. This node represents a **channel from a database** as selected in the corresponding dialog of the Control Panel, showing the channel name and the database name in brackets. Below it the ECUs are arranged.

- **ECU** Second level of the tree, symbolised by 🖥. Each node represents an ECU (electronic control unit) of the superordinate channel, and shows its name. Below it the signals

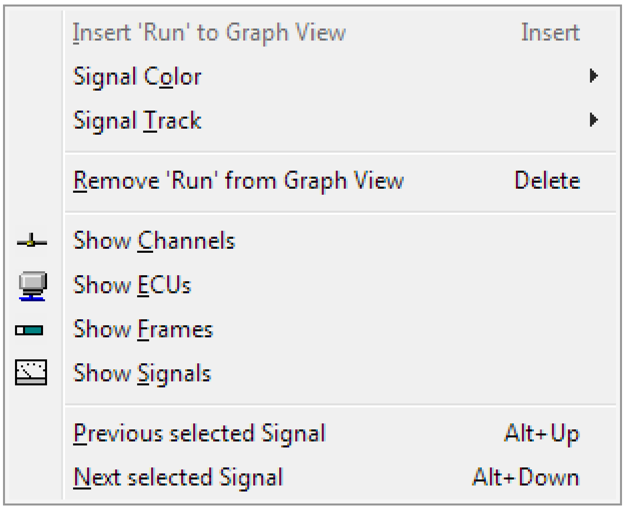

Insert 'Run' to Graph View Insert
Signal Color
Signal Track
Remove 'Run' from Graph View Delete
Show Channels
Show ECUs
Show Frames
Show Signals
Previous selected Signal Alt+Up
Next selected Signal Alt+Down

Figure 5.56: Context menu of the Signal Selection Tree

transmitted by this ECUs can be found. In order to select all signals of the ECU to be shown in the active view, either double click it, or press the "Insert" key, or execute context menu command **Insert to View**. The same applies to the opposite case, to unselect all ECU's signals. Either click it double, or press the "Delete" key, or execute context menu command **Remove from View**. The tooltip of the node shows all the signal names it provides, for a quick overview.

- **Signal** Lowest level of the tree, symbolised by 🗠. This node represents a single signal. In order to select it into the respective view, either double click it, or press the "Insert" key, or execute context menu command **Insert to View**. The same applies to the opposite case, to unselect it. Either double click it, or press the "Delete" key, or execute context menu command **Remove from View**. In case the Graph View is currently active, additional icons visualise the signal color and the signal track. Using the sub menus **Signal Color** and **Signal Track** respectively these properties can be adjusted.

5.6.9 Context menu of the Signal Selection Tree

The menu (Abb. 5.56) consists of the following items:

Menu item	Function
Insert to View	Inserts the signal or all signals belonging to the selected node to the currently active View.
Signal Color	Configures the color of a signal in the graphic display. The current color is highlighted. Select a menu item to change the color.
Signal Track	Configures the track on which the signal is shown. The current track is highlighted. Select a menu item to move the signal to a different track.
Remove from View	Removes the signal or all signals belonging to the selected node from the currently active View.
Show Channels	Collapses the Signal Selection Tree, so that only the channels are visible.
Show ECUs	Collapses or expands the nodes of the tree, so that all ECUs are visible.
Show Frame	Collapses or expands the nodes of the tree, so that all Frames are visible.
Show Signals	Fully expands the Signal Selection Tree, so that all signals are visible.
Previous selected Signal	Locates the previous signal of the tree that is selected in the currently active View. This can be helpful for removing it, or inserting sibling signals.
Next selected Signal	Locates the next signal of the tree that is selected in the currently active View. This can be helpful for removing it, or inserting sibling signals.

5.6.10 Hotkeys of the Signal Selection Tree

Insert	Inserts the signal or all signals of the Channel/ECU to the currently active View.
Delete	Removes the signal or all signals of the Channel/ECU from the currently active View.
Space	Toggles if the signal is part of the currently active View.
Ctrl+T, n	Moves the signal to track 1 to 4. Press Ctrl+T, release, then press numeric key 1 to 4 within 600ms.
Alt+Up	Locates the previous signal of the tree that is selected in the currently active View. This is a global Hotkey that works even if the Signal Selection Tree does not have the input focus.
Alt+Down	Locates the next signal of the tree that is selected in the currently active View. This is a global Hotkey that works even if the Signal Selection Tree does not have the input focus.

5.6.11 Menu reference

File menu

Menu item	Function
Import Options...	Loads module settings from a file (see section 1.4)
Export Options...	Saves module settings to a file (see section 1.4)
Export Signals...	Exports the current View's contents CSV-formatted to a file
Export Graph image...	Saves the graphic display to an image file
Convert Tracefile...	Interpret a binary trace generated via the canAnalyser's Trace-module and save it to a CSV file. This will not display its contents in the Views
Exit	Exits the Signal module

Edit menu

Menu item	Function
Copy CSV	Copies marked lines CSV formatted to clipboard
Toggle Marker *	Sets or Removes a Marker for selected signal
Previous Marker *	Jumps to previous Marker (no wraparound)
Next Marker *	Jumps to next Marker (no wraparound)
Set/Release Time Reference *	Sets Timestamp Zero for selected Signal / Releases previously set Timestamp Zero
Jump to Time Reference *	Jumps to previously set Timestamp Zero Signal
Find Signal in Signal Selection Tree...	Opens a text retrieval dialog to look for a signal name in the Signal Selection Tree

* Not available in Overwrite View

View menu

Menu item	Function
Time relative	Displays the time stamp absolute or relative to the previously displayed signal
Hexadecimal raw values	Displays the uninterpreted values, which are shown in the Raw Value column, in hexadecimal representation rather than in decimal form
Frame representation	Submenu to switch the Frame column representation of the currently active View between short and long framename display. Long representation means, the full path from the frame name down to the signal via all multiplexors, separated by dots. The same can be achieved by a right-click on the Frame column header of the Views
Signal representation	Submenu to switch the Signal column representation of the currently active View between short and long signalname display The same can be achieved by a right-click on the Signal column header of the Views
Draw Guides	Draws additional horizontal guides between the lines in grey in the currently active View
Show recent Frames	Always the most recent telegrams are displayed in the currently active View
Signal Selection Tree	Shows resp activates the Signal Selection Tree and brings it to the front
Scroll View	Shows resp activates the Scroll View and brings it to the front
Overwrite View	Shows resp activates the Overwrite View and brings it to the front
Graph View	Shows resp activates the Graph View and brings it to the front
Logging View	Shows resp activates the Logging View and brings it to the front
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the toolbar

Functions menu

Menu item	Function
Start	Starts the interpretation of messages
Stop	Stops the interpretation of messages
Reset Change Detection All	Resets the value change detection of all signals
Reset Lower Limit Violation Detection All	Resets the value lower limit exceeded flag of all signals
Reset Upper Limit Violation Detection All	Resets the value upper limit exceeced flag of all signals
Reset Timeout Detection All	Resets timeout detection or all signals
Mark Signal Receptions	Marks the received signals in the Graph View by a box each
Clear All	Deletes the contents of all Views
Autosize Columns	Regulates ideal column widths

Options menu

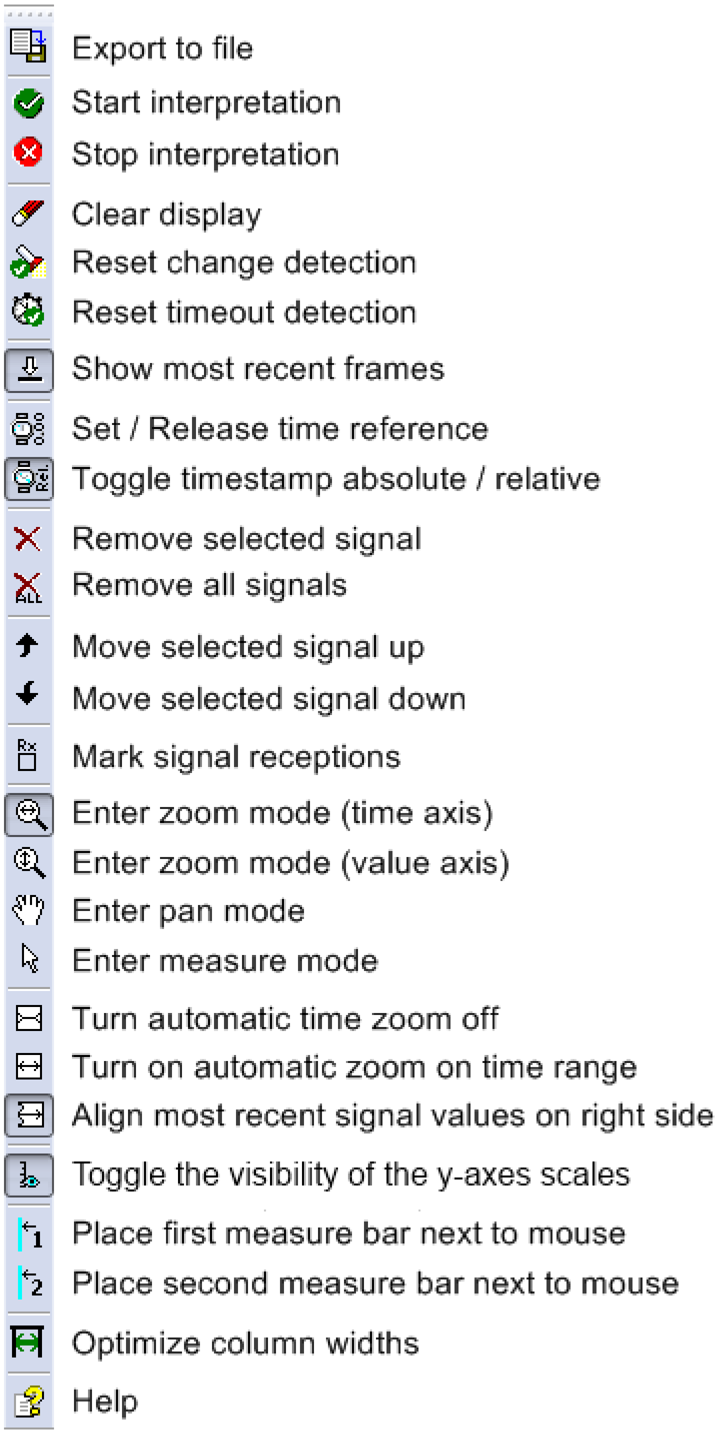
Menu item	Function
Value Change Detection	Switches the value change detection On or Off
Value Range Check	Switches the value range check On or Off
Timeout Detection	Switches the timeout detection On or Off
Value Change Detection Color...	Opens a dialog to select the color with which changed value data are highlighted
Value Range Violation Font Color...	Opens a dialog to select the color with which values outside the defined range are displayed in all views
Graph View Panning Interval	Sub menu to set a fixed width of the visible section as time span. New signal values will be added at the right, while the visible section pans to the left, similar to a physical plotter with firmly installed styluses. Supported time intervals are 1, 5, 10 or 30 seconds, and 1 or 5 minutes.
Graph View Scales Shown	Toggles the visibility of the y-axes scales
Fonts...	Opens a dialog to select the font type in which the data are displayed in all Views

Help menu

Menu item	Function
Help Topics	Opens the online help of the Signal module
About...	Opens the display of the version information of the Signal module

5.6.12 Toolbar

The most important functions of the Signal module can be called via the toolbar (Fig. 5.57).



Export to file
Start interpretation
Stop interpretation
Clear display
Reset change detection
Reset timeout detection
Show most recent frames
Set / Release time reference
Toggle timestamp absolute / relative
Remove selected signal
Remove all signals
Move selected signal up
Move selected signal down
Mark signal receptions
Enter zoom mode (time axis)
Enter zoom mode (value axis)
Enter pan mode
Enter measure mode
Turn automatic time zoom off
Turn on automatic zoom on time range
Align most recent signal values on right side
Toggle the visibility of the y-axes scales
Place first measure bar next to mouse
Place second measure bar next to mouse
Optimize column widths
Help

Figure 5.57: Toolbar of the Signal module

5.6.13 Hotkeys

TAB	Switch between Signal Selection Tree and signal views
Ctrl+TAB	Switch between the different Views
F1	Online-Help
F2	Go to Next Marker
Ctrl+F2	Toggle Marker
Shift+F2	Go to Previous Marker
F5	Start message reception
Shift+F5	Stop message reception
F8	Clear all Views
F11	Hide or Show Signal Selection Tree
Ctrl+C	Copy marked lines CSV formatted to clipboard
Ctrl+E	Export current View contents to a file
Ctrl+F	Find signal
Ctrl+M	Mark signal values in Graph View by a box
Ctrl+O	Load module settings from a file
Ctrl+S	Save module settings to a file
Ctrl+1	Place first measure bar on mouse cursor of graphic display
Ctrl+2	Place second measure bar on mouse cursor of graphic display
Alt+Up	Go to previous selected signal in Signal Selection Tree
Alt+Down	Go to next selected signal in Signal Selection Tree
PageUp	Scroll one page backward in current View
PageDown	Scroll one page ahead in current View
Ctrl+PageUp	Scroll 1000 signals backward in current View
Ctrl+PageDown	Scroll 1000 signals ahead in current View
Ctrl+0	Jump to Time Reference message
Ctrl+1..9	Jump to 10%..90% of current View
Ctrl+W	Close the Signal module

5.7 Signal transmit module

5.7.1 Overview

The signal transmit module is used to change signal values and transmit the affected frames. These signals come from signal databases which have been previously assigned to the configuration of the virtual bus in the control panel.

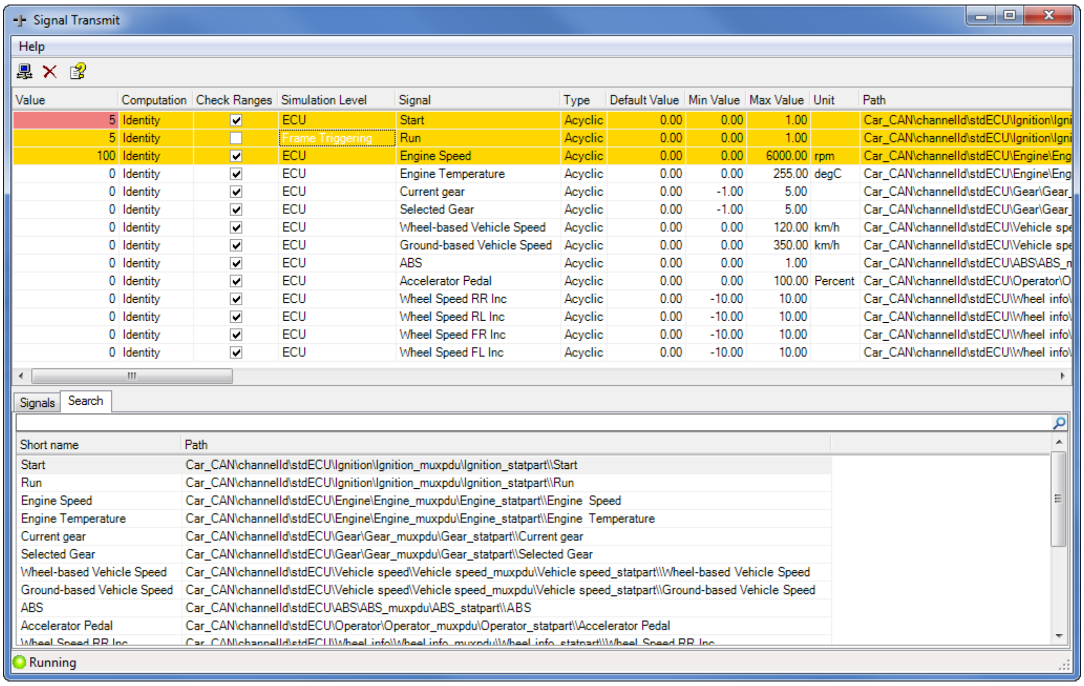
5.7.2 Module window

The available signals are displayed within the lists at the bottom of the module window (Abb. 5.58). To select signals you can use a hierarchical tree combined with a signal search list. The selected signals are displayed in the signal view on the top of the module window.

5.7.3 Signal selection

Signal tree

Selecting signals is done via lists in the lower half of the module window. Within the **signals** tab you find a tree with the database structure and the signals as leave nodes. Every node of the tree can be selected. Via Drag & Drop, the shortcut key **Ctrl+I** or by using the context menu you



Value	Computation	Check Ranges	Simulation Level	Signal	Type	Default Value	Min Value	Max Value	Unit	Path
5	Identity	☑	ECU	Start	Acyclic	0.00	0.00	1.00		Car_CAN\channelId\stdECU\Ignition\Igni
5	Identity	□	Frame Triggering	Run	Acyclic	0.00	0.00	1.00		Car_CAN\channelId\stdECU\Ignition\Igni
100	Identity	☑	ECU	Engine Speed	Acyclic	0.00	0.00	6000.00	rpm	Car_CAN\channelId\stdECU\Engine\Eng
0	Identity	☑	ECU	Engine Temperature	Acyclic	0.00	0.00	255.00	degC	Car_CAN\channelId\stdECU\Engine\Eng
0	Identity	☑	ECU	Current gear	Acyclic	0.00	-1.00	5.00		Car_CAN\channelId\stdECU\Gear\Gear_
0	Identity	☑	ECU	Selected Gear	Acyclic	0.00	-1.00	5.00		Car_CAN\channelId\stdECU\Gear\Gear_
0	Identity	☑	ECU	Wheel-based Vehicle Speed	Acyclic	0.00	0.00	120.00	km/h	Car_CAN\channelId\stdECU\Vehicle spe
0	Identity	☑	ECU	Ground-based Vehicle Speed	Acyclic	0.00	0.00	350.00	km/h	Car_CAN\channelId\stdECU\Vehicle spe
0	Identity	☑	ECU	ABS	Acyclic	0.00	0.00	1.00		Car_CAN\channelId\stdECU\ABS\ABS_n
0	Identity	☑	ECU	Accelerator Pedal	Acyclic	0.00	0.00	100.00	Percent	Car_CAN\channelId\stdECU\Operator\O
0	Identity	☑	ECU	Wheel Speed RR Inc	Acyclic	0.00	-10.00	10.00		Car_CAN\channelId\stdECU\Wheel info\
0	Identity	☑	ECU	Wheel Speed RL Inc	Acyclic	0.00	-10.00	10.00		Car_CAN\channelId\stdECU\Wheel info\
0	Identity	☑	ECU	Wheel Speed FR Inc	Acyclic	0.00	-10.00	10.00		Car_CAN\channelId\stdECU\Wheel info\
0	Identity	☑	ECU	Wheel Speed FL Inc	Acyclic	0.00	-10.00	10.00		Car_CAN\channelId\stdECU\Wheel info\

Short name	Path
Start	Car_CAN\channelId\stdECU\Ignition\Ignition_muxpdu\Ignition_statpart\\Start
Run	Car_CAN\channelId\stdECU\Ignition\Ignition_muxpdu\Ignition_statpart\\Run
Engine Speed	Car_CAN\channelId\stdECU\Engine\Engine_muxpdu\Engine_statpart\\Engine Speed
Engine Temperature	Car_CAN\channelId\stdECU\Engine\Engine_muxpdu\Engine_statpart\\Engine Temperature
Current gear	Car_CAN\channelId\stdECU\Gear\Gear_muxpdu\Gear_statpart\\Current gear
Selected Gear	Car_CAN\channelId\stdECU\Gear\Gear_muxpdu\Gear_statpart\\Selected Gear
Wheel-based Vehicle Speed	Car_CAN\channelId\stdECU\Vehicle speed\Vehicle speed_muxpdu\Vehicle speed_statpart\\Wheel-based Vehicle Speed
Ground-based Vehicle Speed	Car_CAN\channelId\stdECU\Vehicle speed\Vehicle speed_muxpdu\Vehicle speed_statpart\\Ground-based Vehicle Speed
ABS	Car_CAN\channelId\stdECU\ABS\ABS_muxpdu\ABS_statpart\\ABS
Accelerator Pedal	Car_CAN\channelId\stdECU\Operator\Operator_muxpdu\Operator_statpart\\Accelerator Pedal
Wheel Speed RR Inc	Car_CAN\channelId\stdECU\Wheel info\Wheel info_muxpdu\Wheel info_statpart\\Wheel Speed RR Inc

Figure 5.58: Signal-Transmit-Modul

can add the current node to the transmission table. After that all signals which are descendants of the inserted node are added to the signal view.
The context menu provides functions to expand and collapse all nodes in the signal tree and to open a dialog with the signal properties of the selected signal. This option is only available if the selected node represents a signal.

Signal searching

Picking signals within the tree can be tedious in case of big signal databases. For this scenario the search list provides an elegant way to select signals. By default the list shows all signals by their full qualified name which encodes the tree location as a string. Above the list there is a search edit field where search expressions can be typed in.
After a new database has been added to the control panel configuration the search list creates a new search index. On databases with many signal entries you may notice a short delay: While the new index is processed a rotating symbol (⟳) is displayed on the right side of the search edit field. As soon as indexing took place you can use the signal search edit field.
By simply enter a text in the signal search edit filed you can look up signal names. The search is not case sensitive and uses substring search on the qualified signal name. If the search string contains space characters you have to enclose the search string in single or double hyphens. A hypen or a colon must be escaped by a backslash character, otherwise a syntax error is detected and the search edit field turns red. Error descriptions are available in the search fields tooltip.
Signal lookup is started by pressing the start button 🔍 or by using the **Enter** key. If the lookup takes longer a wait symbol ⟳ appears on the right side of the signal search edit field. In this case you can exit the lookup procedure by pressing the symbol with the mouse or by using the **Esc** key. The list shows all signals found till the search operation has been terminated.
If a search procedure took place the remove result symbol ✕ appears on the right side of the signal search edit field. By pressing the symbol with the mouse or using the **Esc** key you can reset the current search result and the search list shows all available signals. Using the **Esc** key a second time also resets the current search expression.

Additional to the search for signal name parts you can lookup a signal by other properties a signal provides. For this you have to use special keywords and the syntax "keyword:search expression" in the signal search edit field. Keywords can be combined but using the same keyword multiple times is not supported. Keywords are not case sensitive.

Keyword	search expression
channel	Search all signals belonging to channels which have the search expression as a substring in their element id
frame	Search all signals belonging to frames which have the search expression as a substring in their element id
pdu	Search all signals belonging to PDUs which have the search expression as a substring in their element id
ecu	Search all signals belonging to ECUs which have the search expression as a substring in their element id. By the reserved search expression "(none)" you can locate all signals, that are not assigned to any ECU.

To add a signal to the signal view you have to select it by using the mouse or the keyboard. Multiple selection is supported and can be accomplished in a standard way by using the **Ctrl** or the **Shift** key. By Drag & Drop or by pressing **Ctrl+I** you can add the selected signals to the signal view.

5.7.4 Sending signals

The signal view contains all manually added signals and displays the following columns:

column	description
Value	The value that should be transmitted. Cells turn red when range checking is active and the value is out of the valid range.
Computation	A list of computations which are defined in the database for the given signal. They describe how the entered signal value is turned into the transmitted value.
Check Ranges	Is checked if range checking should be active for this row. In this case transmitting of invalid signal values is prohibited.
Simulation Level	Determines on which level signal transmission is simulated. In mode "frame triggering" only the frame that has been used to add the signal is transmitted. In mode "ECU" further frames could be transmitted if the signal is mapped to multiple frames.
Signal	The signal name.
Type	Determines if the signal is sent cyclic or acyclic. Note that a cyclic signal could be sent acyclic on further frames.
Default Value	The default value from the database.
Min Value	The minimum value from the database. The value changes depending on the selected computation.
Max Value	The maximum value from the database. The value changes depending on the selected computation.
Unit	The unit from the database.
Path	The qualified name as defined in the search tree.

Sorting rows can be achieved by single clicks on the column headers. By pressing the **Shift** key sorting by multiple columns is possible.

To transmit a signal you first have to enter a new value. If the value is valid the whole row turns yellow which means that the value has been set, but not yet transmitted. This behaviour allows to change multiple signals before transmitting them simultaneously. You start transmission of changed signals by pressing the send button 🖥 in the button bar. The signals are sent and the value cells turn white to show that the value is the current value in the system.
Sending a signal value leads internally to updates of PDUs and frames which contain the changed signal. But which message frames are really sent depends on the content of the assigned databases, which describes cyclic triggered and event triggered PDUs/frames.
Via the remove button ✕ you are able to remove signals from the signal view. Alternatively you can use the context menu or the **Del** key.
When loading project files in the control panel it is possible that referenced database files are temporarily or permanently not available. In this case the signal transmit module can load the list of selected signals only partially. Signals which suffer from this problem are marked grey in the signal view and and they can not be sent nor all their properties are available. You can remove them from the signal view but that is not necessary. If the database gets available on the next project load the signals are displayed as usual.
Both, the search list and the signal view have the context menu entry **Signal properties**. With this entry you open a dialog which shows further available signal attributes.

5.7.5 Menu reference

Help menu

Menu point	Function
Help topics	Opens the online help of the Signal transmit module
About...	Opens the display of the version information of the Signal transmit module

5.7.6 Tool bar

Symbol	Function
🖥	Transmit all signals
✕	Remove selected signal from the signal view
❓	Open help file

5.7.7 Hotkeys

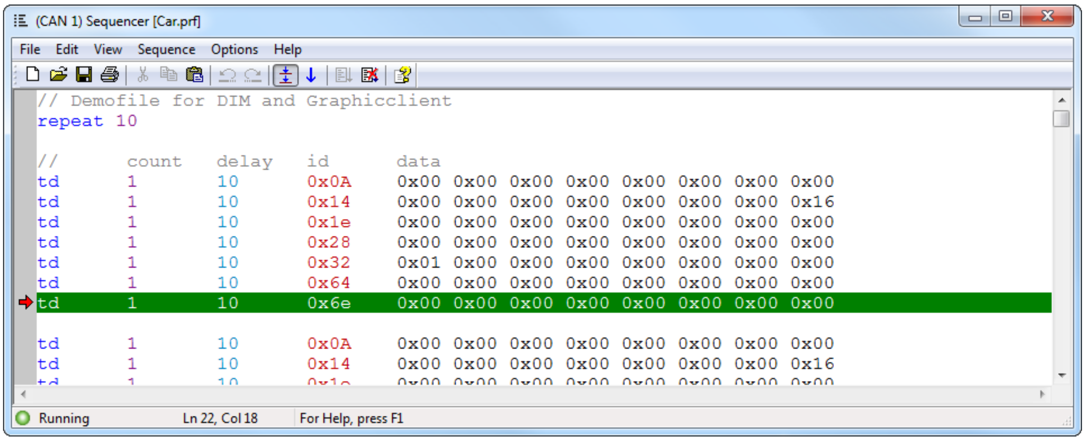
Ctrl+I	Add the current node to the transmission table
Del	Remove selected signals
F1	Online-Help

5.8 Sequencer-module

5.8.1 Overview

The Sequencer-module (Fig. 5.59) provides processing of command-controlled message sequences and can therefore be used to simulate nodes or protocol sequences or to generate a certain bus load. It has commands for:

- Transmitting data and remote messages


(CAN 1) Sequencer [Car.prf]
File Edit View Sequence Options Help
Running
Ln 22, Col 18
For Help, press F1

```
// Demofile for DIM and Graphicclient
repeat 10

//      count   delay   id      data
td      1       10      0x0A    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x14    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x16
td      1       10      0x1e    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x28    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x32    0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x64    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x6e    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

td      1       10      0x0A    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
td      1       10      0x14    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x16
td      1       10      0x1e    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

Figure 5.59: Sequencer-module

- Waiting for data or remote messages
- Adding delay times
- Waiting for user input
- Repeating command blocks

After every command it is possible to add a delay time.
The user interface of the Sequencer-module consists of a menu bar, a toolbar, a status bar and the editor window. When processing a sequence, the currently transmitted message of the sequence is visualized via a scroll bar.

5.8.2 Command syntax

For the syntax of the commands, the following is to be noted:

- Upper/lower case of the commands is not relevant
- The parameters of the commands must be separated by at least one space or tab character
- A command must be only one line long
- The parameters can be entered in decimal, hexadecimal or octal form. Differentiation is made by means of the prefix

Prefixes for parameters:

Prefix	Meaning	Example
0x or 0X	hexadecimal notation	0xF8
0	octal notation	080
	decimal notation	100

Comments

Comments can be entered at the end of a command line separated by ; or //
Lines that do not contain valid commands are automatically interpreted as comments.

Example:

```
td   20   2   100  0x01 0x02 0x03     ; Comment at the end
                                      ; of a command line
// Comment line
This is also a comment line
```

5.8.3 Command overview

Sequencer-module supports the following commands:

Command	Meaning	Action	Bustype
td	transmit data	Transmission of a message	CAN, LIN
tds	transmit data (std)	Transmission of a message (std)	CAN, LIN
tde	transmit data (ext)	Transmission of a message (ext)	CAN, LIN
tdsl	transmit data (std) long	Transmission of a message (std) with extended data length	CAN-FD
tdel	transmit data (ext) long	Transmission of a message (ext) with extended data length	CAN-FD
tdsfl	transmit data (std) fast long	Transmission of a fast data message (std) with extended data length	CAN-FD
tdefl	transmit data (ext) fast long	Transmission of a fast data message (ext) with extended data length	CAN-FD
tr	transmit remote	Transmission of a message request	CAN, LIN
trs	transmit remote (std)	Transmission of a message request (std)	CAN, LIN
tre	transmit remote (ext)	Transmission of a message request (ext)	CAN, LIN
wd	wait for data	Waiting for a message	CAN, LIN
wr	wait for remote	Waiting for a message request	CAN, LIN
delay	delay	Adding a delay time in ms	CAN, LIN
pause	pause	Waiting for user action	CAN, LIN
repeat	repeat	Repetition of a block	CAN, LIN
endrep	end repeat	End of a repetition block	CAN, LIN

For LIN the postfix *e* resp *(ext)* means enhanced CRC message format acc. to LIN 2.0++. The Postfix *r* resp *remote* analogously serves as message request by ID only.

Important: The parameter <Delay_time> indicates how long the sequencer is delayed after a command. This time period can be longer than the specified value depending on the operating system and the load of the processor.

Transmission of a message

```
td[s|e] <Number_of_repeats> <Delay_time> <Message-ID> <Data_field>
```

A message with the identifier `<Message-ID>` and the data field `<Data_field>` is transmitted.
After transmission of the message, `<Delay_time>` ms is awaited.
This process is repeated `<Number_of_repeats>` times.
While `td/tds` transmits standard frames (11 bit identifier), the `tde` command is used to transmit an extended frame (29 bit identifier).
With a transmit command (`td`, `tr`), a telegram is placed in the Transmit queue. There is no delay time until the CAN telegram has been transmitted on the bus. With low baud rates, telegrams may therefore not be transmitted fast enough and the Transmit queue may fill up. In this case the delay times between the telegrams also no longer match. On the other hand it is thus possible to generate high bus loads.

Example:

```
//  Transmission of 20 messages with ID = 0x10A,
//  Data field = 11 22 33 44 55 66 77 88
//  there is a delay time between messages of 100 ms in each case
//
td  20  100  0x10A  11 22 33 44 55 66 77 88
```

Transmission of a message request

```
tr[s|e] <Number_of_repeats> <Delay_time> <Message-ID> <Length_of_data_field
```

A remote message with the identifier `<Message-ID>` and `<Length_of_data_field>` data bytes is transmitted with a delay time `<Delay_time>` `<Number_of_repeats>` times in sequence. While `tr/trs` transmits standard frames (11 bit identifier) the `tre` command is used to transmit an extended frame (29 bit identifier).

Example:

```
//  Transmission of 10 remote messages with ID = 33
//  and Data Length Code = 4
//  There is a delay time of 200 ms between each message
//
tr  10  200  33  4
```

Waiting for a message

```
wd <Delay_time> <Message-ID>
```

Sequencer-module stops until a message with identifier `<Message-ID>` has been received. There is then a delay time of `<Delay_time>` ms, before the next command is processed.

Example:

```
//  Waiting for a message with ID = 0x54
//  then delay 500 ms
//
wd  500  0x54
```

Waiting for a message request

```
wr <Delay_time> <Message-ID>
```

The Module is stopped until a remote message with identifier `<Message-ID>` has been received. There is then a delay time of `<Delay_time>` ms before the next command is processed.

Example:

```
//  Waiting for a remote message with ID = 0x54
//  then wait 500 ms
//
wr  500  0x54
```

Adding a delay time

```
delay <Delay_time>
```

Wait `<Delay_time>` ms before the next command is processed.

Example:

```
//  wait for one second
//
delay 1000
```

Repetition of a block

```
repeat <Number_of_repeats>
  <Further commands>
endrep
```

The command `repeat` indicates the number of times a subsequent block is repeated.
A block is ended with the command `endrep`.

Example:

```
// A block of four messages should be transmitted 5
// times.
// After sending the messages, a message with
// identifier 60 should be waited for.
// Then the sequence is repeated 10 times.
// The indentations are not necessary
// but they show the structure of the sequence clearly.
//
repeat 10
  repeat 5
    ;   Count Delay ID    Data
    td  1     1     41    12 13 14 15
    td  1     1     0x3f  33 44 55
    td  1     10    0x0a  22 33
    td  1     100   32    76 65 43 26
  endrep
  wd  1  60
endrep
```

Waiting for user input

```
pause <Output_text>
```

A dialog is opened and the text `<Output_text>` is displayed. Then the sequence is stopped until the **OK** button in the dialog window is clicked. With the **Abort** button it is possible to end processing.

Example:

```
//  Waiting for user input
//
pause  Press OK to resume the sequence!
```

5.8.4 Menu reference

File menu

Menu item	Function
New	Opens a new editor window. Where applicable it is possible to save changes of a previously opened editor window
Open...	Opens a previously saved Sequencer file
Save	Saves a Sequencer file
Save As...	Saves a Sequencer file under a new file name
Import Trace...	Imports an available canAnalyser trace file (ending .tr0, .tr1, ...) to the Sequencer-module. The individual CAN telegrams in the trace file are edited and inserted as td-commands in the editor at the current cursor position. The delay parameter is calculated from the time stamps of the CAN telegrams and rounded to 1 ms. Important: the number of imported telegrams is limited. Only the first 50000 telegrams of a trace file at most are imported.
Print...	Prints the current Sequencer file
Print Preview	Shows a preview of the print-out
Page Setup...	Opens a window to select the page edges for the print-out
Print Setup...	Opens the Windows printer dialog
Exit	Exits Sequencer-module

Edit menu

Menu item	Function
Undo	Cancels the last change
Redo	Restores the cancelled change
Cut	Removes the marked area of the editor window and copies it to the Windows clipboard
Copy	Copies the marked area of the editor window to the Windows clipboard
Paste	Inserts the contents of the Windows clipboard at the cursor position
Delete	Deletes the marked area of the editor window
Select All	Selects the complete contents of the editor window
Find...	Opens the dialog to enter a string to be searched for
Find Next	Searches for the next occurrence of the string entered
Find Previous	Searches for the previous occurrence of the string entered
Replace...	Replaces a string to be searched for with another string to be entered

View menu

Menu item	Function
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar

Message sequence menu

Menu item	Function
Start	Starts the transmission of the currently loaded message sequence
Stop	Stops Sequencer-module
Single step	Execute next single program step

Options menu

Menu item	Function
Set Tab Stops...	Opens a dialog to define the tab width of the editor window
Set Screen Font...	Opens a dialog to define the font used by the editor window
Set Printer Font...	Opens a dialog to select the font type for the print-out
Single-step mode	Executes the program sequence step by step rather than in one go
Autoscroll	Makes sure that the currently active program step is always in view

Help menu

Menu item	Function
Help Topics	Opens the online help of Sequencer-module
About...	Opens the display of the version information of Sequencer-module

5.8.5 Toolbar

The most important functions of Sequencer-module can also be called via the toolbar (Fig. 5.60).

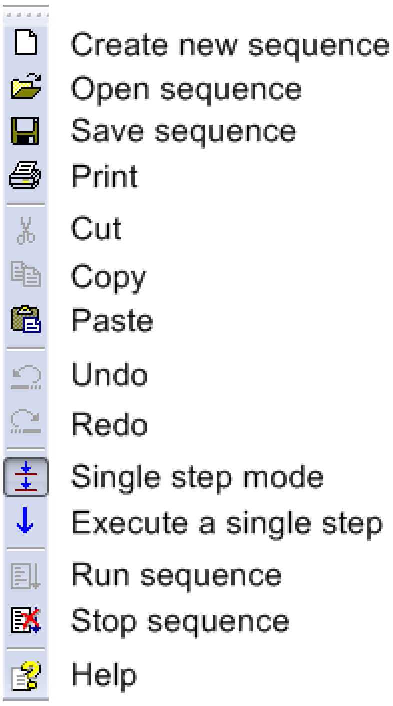


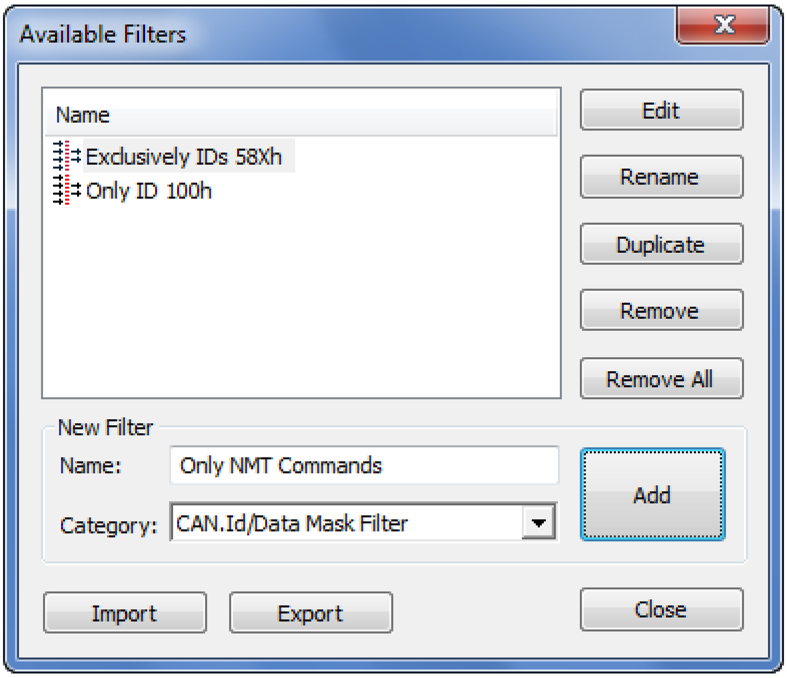
Figure 5.60: Toolbar of Sequencer-module

5.8.6 Hotkeys

Ctrl+N	Create new sequencer-file
Ctrl+O	Open sequencer-file
Ctrl+S	Save sequencer-file
Ctrl+P	Print sequencer-file
Ctrl+Z	Undo changes
Ctrl+Y	Redo changes
Ctrl+X	Cut text
Ctrl+C	Copy text
Ctrl+V	Paste text
Del	Delete text
Ctrl+A	Select all
Ctrl+F	Find text
F3	Go to next search result
Shift+F3	Go to previous search result
Ctrl+H	Replace text
F5	Start message sequence
Shift+F5	Stop message sequence
F7	Toggle Single-step mode
F9	Perform a single step
F1	Online-Help

5.9 Handling of filters

With the aid of a filter, certain messages become visible or invisible to an analysis module. Message filters are available throughout the application. Because the filters are configured centrally they can be activated within an analysis modules by simply selecting it.



Exclusively IDs 58Xh
Only ID 100h

New Filter
Name: Only NMT Commands
Category: CAN.Id/Data Mask Filter

Figure 5.61: Creating and configuring filters

5.9.1 Filter configuration

Creating filters is done with the aid of the **Available Filters** dialog (Fig. 5.61) which can be opened within the Control Panel, the Receive module or the Trace module via the toolbar icon or the menu command **Available Filters...**

Creating new filters

To create a new filter you have to specify a name and a filter category for it. The subsequent click on the button **Add** creates the filter and automatically opens it's configuration dialog. As soon as you closed this dialog with **OK** the new filter will be available within the analysis modules.

Modifying a filter configuration

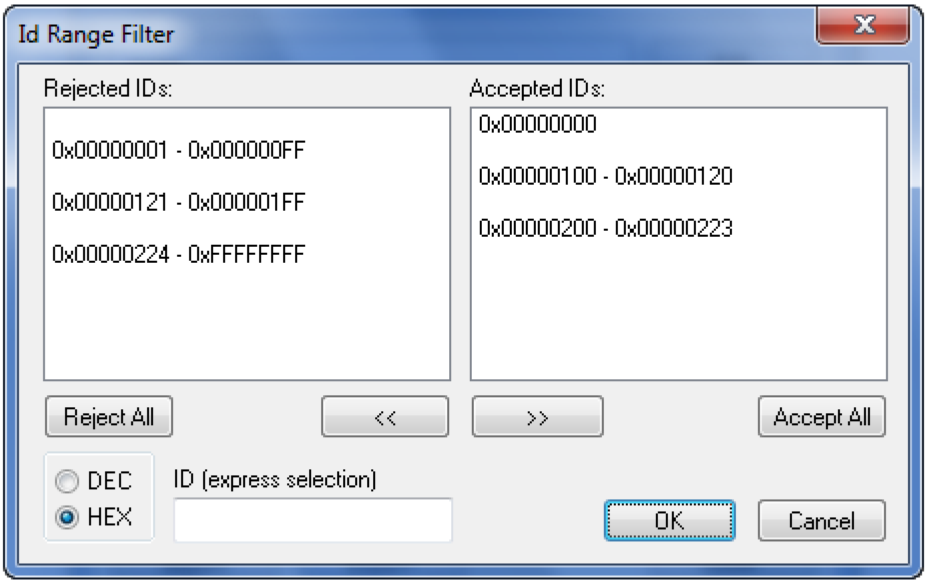
To modify a filter open it's configuration dialog via the **Space** key or the button **Edit**. The new filter configuration is automatically applied by analysis modules that are currently using this filter.

Renaming filters

An existing filter can be renamed by pressing the **F2** key or clicking the button **Rename**. The new name is automatically applied by the analysis modules.

Removing filters

Single filters can be removed from the list of available filters via the key **Del** or the button **Remove**. The button **Remove All** removes all available filters but shows a confirmation dialog before really removing them.
Analysis modules which are currently using a removed filter will react with no longer filtering the incoming message stream. This means: All messages will be accepted and the filter selection within the analysis module will be switched to **<No Filter>**.



Id Range Filter

Rejected IDs:

0x00000001 - 0x000000FF

0x00000121 - 0x000001FF

0x00000224 - 0xFFFFFFFF

Accepted IDs:

0x00000000

0x00000100 - 0x00000120

0x00000200 - 0x00000223

○ DEC

● HEX

ID (express selection)

Figure 5.62: Id Range Filter configuration

Duplicating a filter configuration

If several similar filters with only small differences are required it can be very efficient to duplicate an existing filter configuration via the button **Duplicate** and then modify it's name and settings.

Taking over filters in other analysis configurations

To make the set of filters available to other analysis configurations you have to export the filters to an extra file by using the button **Export**. From there the filters can be imported into other analysis configurations via the button **Import** (see also section 1.4).

5.9.2 Id Range Filter

With the aid of the Id Range filter, certain messages become visible or invisible (Fig. 5.62). This is selected via the identifier.
The filter dialog contains the following elements:

Element	Meaning
Rejected IDs	List of the identifiers whose assigned messages do not pass the filter
Accepted IDs	List of the identifiers which pass the filter
>>	Assignment of the identifier group selected in the list Rejected IDs to the list Accepted IDs
<<	Deletion of the entry selected in the list Accepted IDs
Accept All	When this button is pressed, all messages are received (all identifiers are entered in the list Accepted IDs)
Reject All	When this button is pressed, all messages are blocked (all identifiers are deleted from the list Accepted IDs and entered in the list Rejected IDs)
ID (express selection)	A filter function can be entered alphanumerically via this command line. This enables quick selection of identifiers. Individual identifiers or complete identifier arrays can be blocked or released. Individual filter commands are separated by a space. The command line facilitates selection of identifiers.
DEC/HEX	This checkbox is used to select whether the identifiers are displayed in this dialog window in hexadecimal or decimal form.

Syntax of the command line:

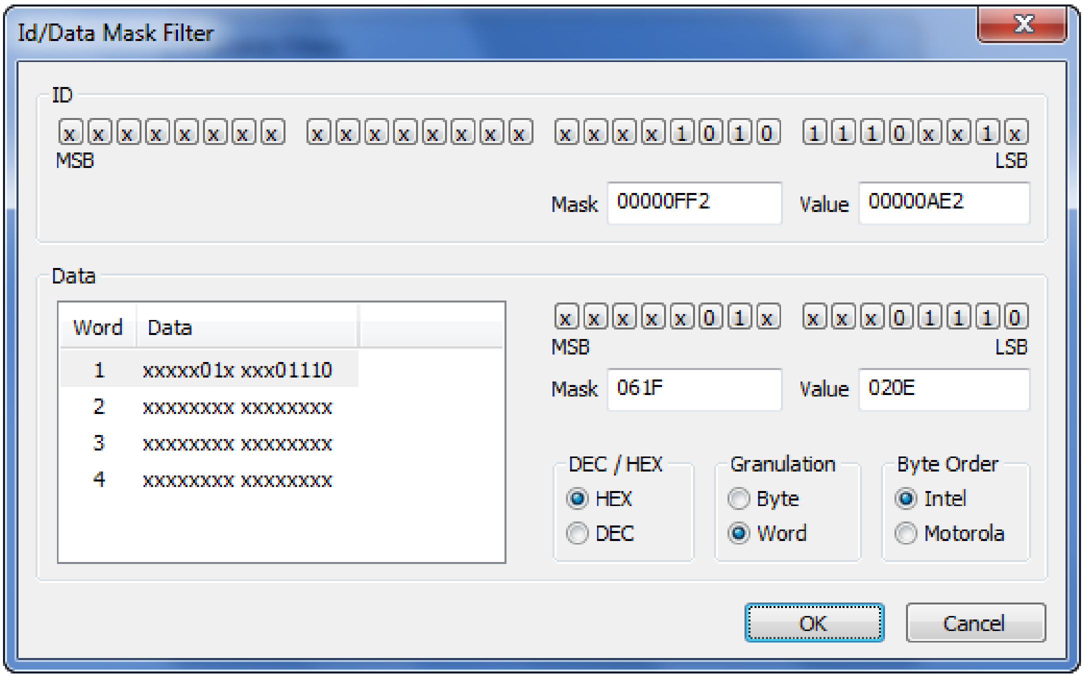
Command	Meaning
-ID	Move identifier ID into the list of rejected IDs
-ID1,ID2	Move identifier array ID1 to ID2 into the list of rejected IDs
+ID	Move identifier ID into the list of accepted IDs
+ID1,ID2	Move identifier array ID1 to ID2 into the list of accepted IDs
z.B.: -3,8	Moves the identifiers 3 to 8 into the list of rejected IDs, i.e. the identifiers 3 to 8 are filtered out

5.9.3 Id/Data Mask Filter

With the Id/data filter (Fig. 5.63), filtering on certain identifiers and/or databytes is possible. The filter condition is defined via bit-masks, which are later compared with each receive message. Only messages that fulfill the filter condition pass the filter. The filter condition of the individual bits can be altered by clicking with the left mouse button.

Bit	Meaning
0	Bits marked with 0 must have the value 0 in order that the filter condition is fulfilled
1	Bits marked with 1 must have the value 1 in order that the filter condition is fulfilled
x	Bits marked with x are not relevant for the filter condition and are not fulfilled

The bit-masks of the filter conditions can also be altered manually via the input fields **Mask** and **Value** . In **Mask** the bits with the value 1 are marked as relevant for the filter condition. In **Value** these relevant bits receive their nominal value (0 or 1). The input/display of the input fields can be made according to the settings in the box **DEC/HEX** in decimal or hexadecimal form.
The buttons **Byte** and **Word** define the granulation of the data field. If **Word** granulation is selected, **Intel** (Little Endian) or **Motorola** (Big Endian) format can be set for the byte order.
The filter condition is fulfilled when all bits marked as relevant have the nominal value defined for them. However, this also means that a filter condition in which all bits are marked as not relevant is fulfilled for every message and therefore all messages pass the filter.



Id/Data Mask Filter

ID

x x x x x x x x x x x x x x x x x x x x 1 0 1 0 1 1 1 0 x x 1 x

MSB LSB

Mask 00000FF2 Value 00000AE2

Data

Word	Data
1	xxxxx01x xxx01110
2	xxxxxxxx xxxxxxxx
3	xxxxxxxx xxxxxxxx
4	xxxxxxxx xxxxxxxx

x x x x x 0 1 x x x x 0 1 1 1 0

MSB LSB

Mask 061F Value 020E

DEC / HEX: ● HEX ○ DEC

Granulation: ○ Byte ● Word

Byte Order: ● Intel ○ Motorola

Figure 5.63: Id/Data Mask Filter configuration

5.10 Integrating own analysis modules

Via the open .NET programming interface the user has the possibility to extend the canAnalyser by own modules and user interfaces. Own, autonomous, on the .NET Framework based modules can be written by using common Windows development environments (e.g. Visual Studio .NET, Delphi) and can then be integrated to the canAnalyser. Consequently it's possible to create user interfaces for own systems respectively for devices and tools with system specific analysis functions.

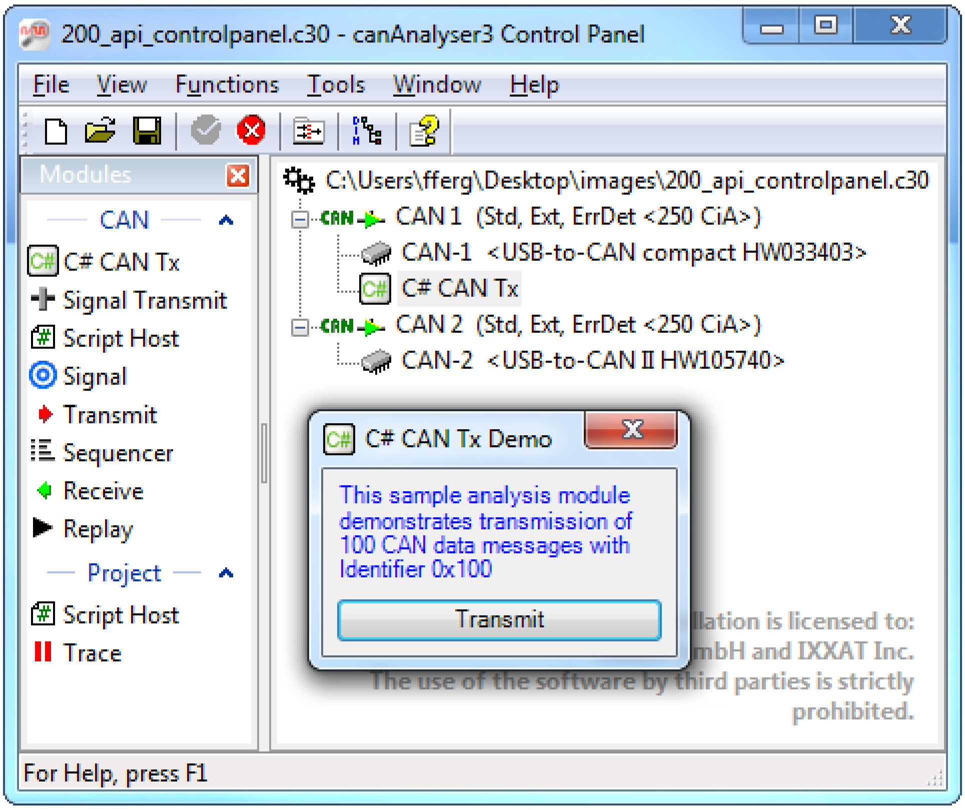
A guidance for developing own analysis modules and a detailed description of the API is installed with the canAnalyser in the form of the online help file IXXAT.MbsAnalyser.chm. To ease off your first steps the canAnalyser setup also installs several programming samples which may be used as base code for user defined modules.

Depending on the extension and the purpose of an analysis module and it's user it can be an advantage to integrate the analysis module into the canAnalyser as compiled assembly:

- The analysis module is provided to someone else and it's source code neither must not be handed down nor be modified.
- The analysis module is very extensive and must be subdivided into several source code files.

User defined analysis modules are integrated into the Control Panel via a plug-in mechanism. Concerning this the assemblies of the user defined modules inclusive their referenced assemblies have to be placed info one of these directories:

- Installationpath
- Public Documents\IXXAT\canAnalyser\3.0\UDModules
- My Documents\IXXAT\canAnalyser\3.0\UDModules



C:\Users\fferg\Desktop\images\200_api_controlpanel.c30
CAN 1 (Std, Ext, ErrDet <250 CiA>)
CAN-1 <USB-to-CAN compact HW033403>
C# CAN Tx
CAN 2 (Std, Ext, ErrDet <250 CiA>)
CAN-2 <USB-to-CAN II HW105740>

This sample analysis module demonstrates transmission of 100 CAN data messages with Identifier 0x100

llation is licensed to:
mbH and IXXAT Inc.
The use of the software by third parties is strictly prohibited.

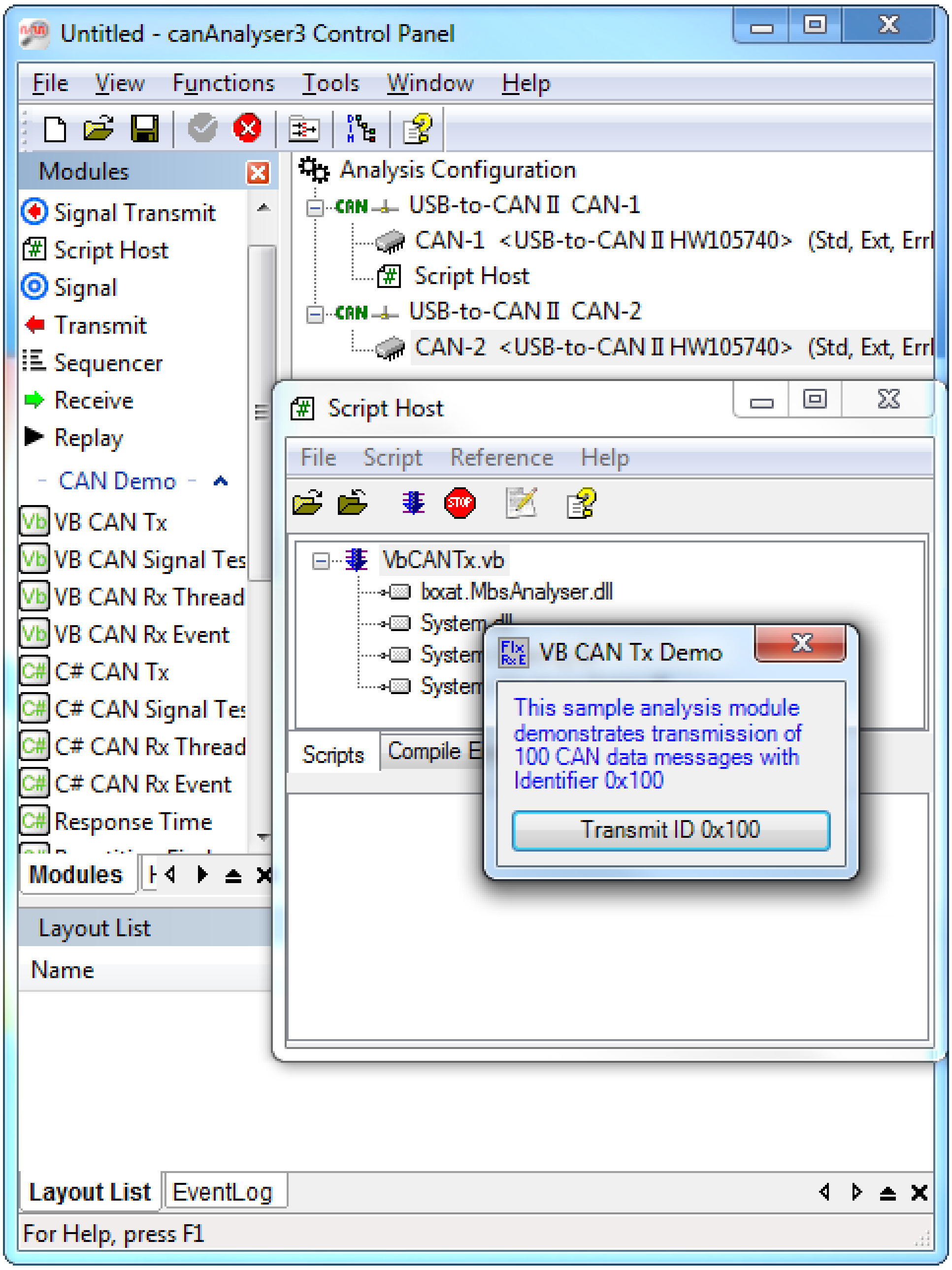
Figure 5.64: Control Panel with "C# CAN Tx" sample analysis module

All User defined modules that are automatically detected at application startup are displayed beside the standard modules within the Modules window of the Control Panel and can be started via Drag-and-Drop (Fig. 5.64).
Any modification at the source code of an analysis module does not go into operation before a restart of the canAnalyser application therefore.
During setup you can install example projects for user defined modules implemented in C# of VB.NET within the "Public Documents\IXXAT\canAnalyser\3.0\Api\Examples" directory. Precompiled assemblies are installed in "Public Dokuments\IXXAT\canAnalyser\3.0\UDModules" if requested. The output path of the example projects is also set to this path, therefore recompiling the example projects overwrites the precompiled assemblies.

5.11 Script Host

5.11.1 Overview

Because creation and modification of scripts is very flexible they ease off the work of a developer during the testing phase as well as searching errors by service engineers on-site. At this it's not mandatory having an installed development environment.
For configuring and executing scripts the canAnalyser Control Panel provides a Script Host as analysis module within the Modules window (Fig. 5.65). In here executable scripts are based on the same .NET programming interface as used for integration of own analysis modules (Fig. 5.66). The Script Host supports scripts with graphical user interface as well as console based scripts. Each script with graphical user interface has it's own window. Console based scripts have an optional, text oriented input/output window which is integrated into the Script Host window and can alternatively stay invisible for execution time.



This sample analysis module demonstrates transmission of 100 CAN data messages with Identifier 0x100

Figure 5.65: GUI and console programming sample executed as script

Against the integration of an analysis module via it’s assembly the Script Host has the benefit not having to recompile an assembly and to restart the canAnalyser after a modification. The user simply has to restart the script’s source code file within the Script Host window.
For the purpose of the Script Host you have to consider the following restrictions:

- The script has to consist of one single source code file.
- The script has to be coded in C# or VisualBasic.NET.
- There are no further embedded resources supported beside the Form resources (the related .resx or .resources file).
- The Script Host does not support integrated debugging.

As long as you consider these restrictions for coding it’s possible to execute the identical source code as script as well as to compile it to an assembly and to integrate it as analysis module

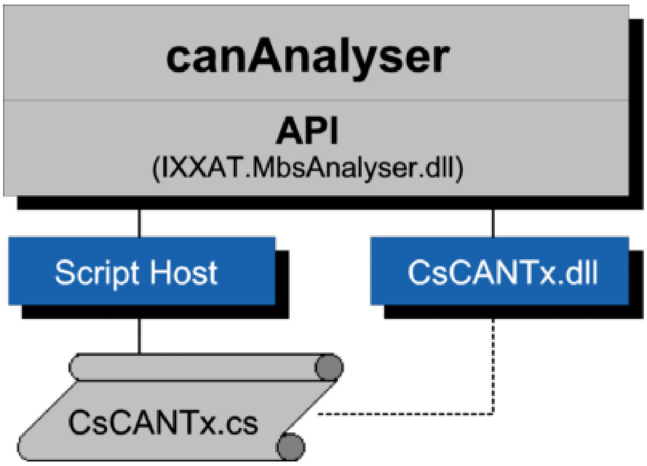


Figure 5.66: Integration of "C# CAN Tx" sample as assembly DLL and via Script Host.

into the Control Panel (Fig. 5.66). This way integrated debugging is possible anyhow.Console scripts cannot be integrated as analysis module. Therefore there is really no way for integrated debugging these.

5.11.2 The Script Host within the analysis configuration

The Control Panel provides the Script Host as analysis module within the Modules window. By using Drag-and-Drop it can be dragged onto the root node of the analysis configuration or onto a single bus (Fig. 5.67). This is dependent on the particular application:

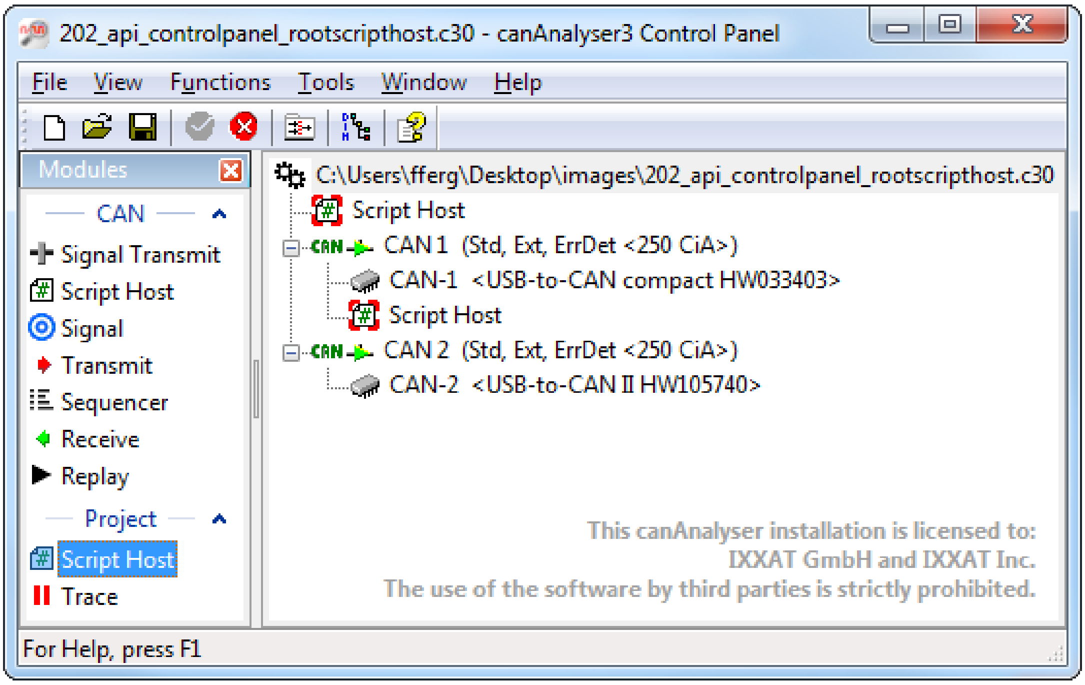
- For a gateway script you hang in the Script Host below the configuration root node because such scripts require simultaneous access to several busses.
- A script for device simulation is performed in a Script Host hanging at a single bus as rule. With the aid of various Script Host instances it's also possible to simulate more than one device of the same type within your system.

5.11.3 The Script Host window

The Script Host window shows two index cards in the upper half:

- **Scripts**: Here you control the Script Host. Scripts can be loaded, unloaded configured, executed and stopped.
- **Compile Errors**: This index card outputs errors while compiling and starting a script.

The lower partial window shows index cards with input/output windows for console scripts. A script's console is not displayed until it's explicitly requested by the script. This way it's possible to execute invisible scripts.
Before a script can be executed you have to load it's source code file before. This can take place via the button 📂, the related context menu entry (Fig. 5.68) or via Drag-and-Drop.
Because each analysis module uses classes and interfaces of the canAnalyser programming interface and at least has one Windows Form (with GUI scripts) the following assemblies are preconfigured as references by default:


202_api_controlpanel_rootscripthost.c30 - canAnalyser3 Control Panel
File View Functions Tools Window Help
Modules
CAN
Signal Transmit
Script Host
Signal
Transmit
Sequencer
Receive
Replay
Project
Script Host
Trace
C:\Users\fferg\Desktop\images\202_api_controlpanel_rootscripthost.c30
Script Host
CAN 1 (Std, Ext, ErrDet <250 CiA>)
CAN-1 <USB-to-CAN compact HW033403>
Script Host
CAN 2 (Std, Ext, ErrDet <250 CiA>)
CAN-2 <USB-to-CAN II HW105740>
For Help, press F1

This canAnalyser installation is licensed to:
IXXAT GmbH and IXXAT Inc.
The use of the software by third parties is strictly prohibited.

Figure 5.67: Script Host may have access to one or all busses

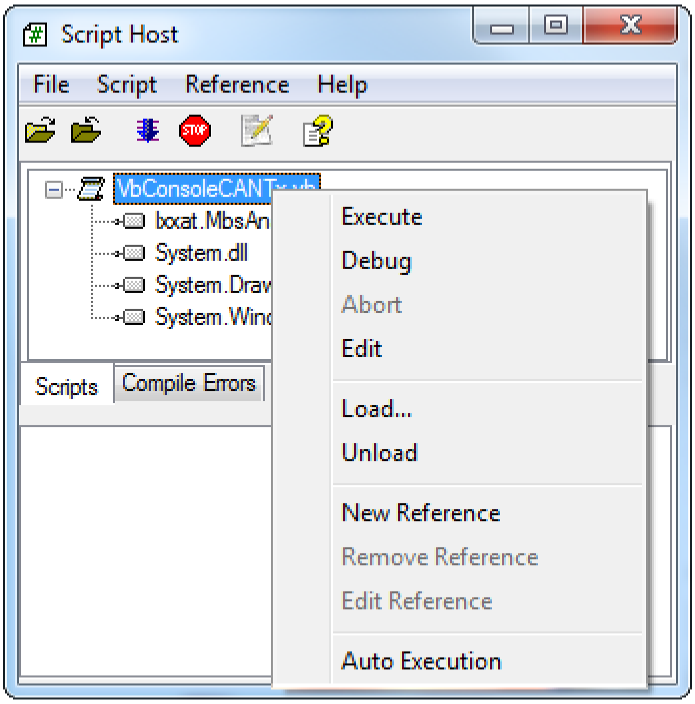


Figure 5.68: Script Host context menu

- IXXAT.MbsAnalyser.dll
- System.dll
- System.Drawing.dll
- System.Windows.Forms.dll

If a script references further external assemblies you will have to add these references with the aid of the command **Add Reference** in the context menu. Adding reference assemblies via Drag-and-Drop is also supported. These referenced assemblies can be located directly via their filenames within the following directories:

- Installation path
- C:\Users\Public\Documents\IXXAT\canAnalyser\3.0\API\UDModules
- Directory of the script
- The .NET Framework

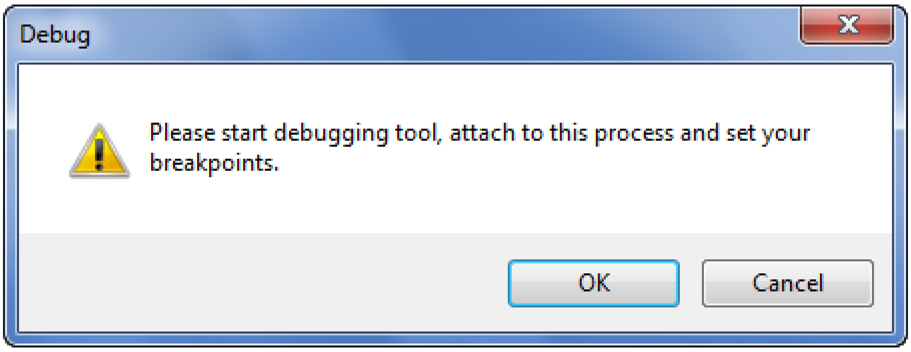
Otherwise references have to be added as absolute filenames. It's not sufficient having the referenced assembly within the GAC (Global Assembly cache). In case of need you have to place a copy into a conventional directory outside the GAC. But in general such an assembly has to be in either in the .NET Framework or within the canAnalyser program directory respectively within one of it's direct or indirect subdirectories.

5.11.4 Editing scripts

Via button or the related context menu entry you may edit a script's source code file. About this the Script Host opens the editor registered at the operating system for the according file extension. The linkage to the editor may be defined manually within the properties of the source code file by using the Windows Explorer. If no specific editor is registered the Script Host opens the Windows standard editor Notepad.
Editing source code is essentially more comfortable by using specialized editors. Alternatively to commercial products like Microsoft Visual Studio .NET there are also free available tools like #Develop (www.sharpdevelop.net). Especially for creation and modification of graphical user interfaces such an environment is recommended.

5.11.5 Executing scripts

A loaded and configured script is executed via button or the corresponding context menu entry. For this the user has to select the affected script before. If the source code contains syntax errors or the script could not have been started the index card **Compile Errors** shows appropriate error messages. However, if the script could have been started with no errors then the script's main window is displayed and the corresponding entry within the Script Host receipts this status with icon .
Scripts which **Auto Execution** option in the context menu is switched on are automatically executed when loading the saved analysis configuration the next time.



Debug

Please start debugging tool, attach to this process and set your breakpoints.

Figure 5.69: Script Host waiting for attaching debugger

5.11.6 Stopping scripts

An executed script can be stopped by clicking the button STOP or the corresponding context menu entry or by manually closing the script window. This is signalled by the Script Host with status icon. Also unloading the script via button or the context menu entry aborts the scripts.

5.11.7 Debugging scripts

If you have problems at the runtime of your script you may possibly want to debug the script. Therefore the Script Host provides the possibility to execute a script with debug information. You may do this via the menu entry **Script | Debug** or the corresponding entry of the context menu. As a result the Scipt Host displays a message box that delays the actual script execution until you attached your debugging environment to the canAnalyser process.
The following steps illustrate how to debug a script by using Microsoft Visual Studio 2008:

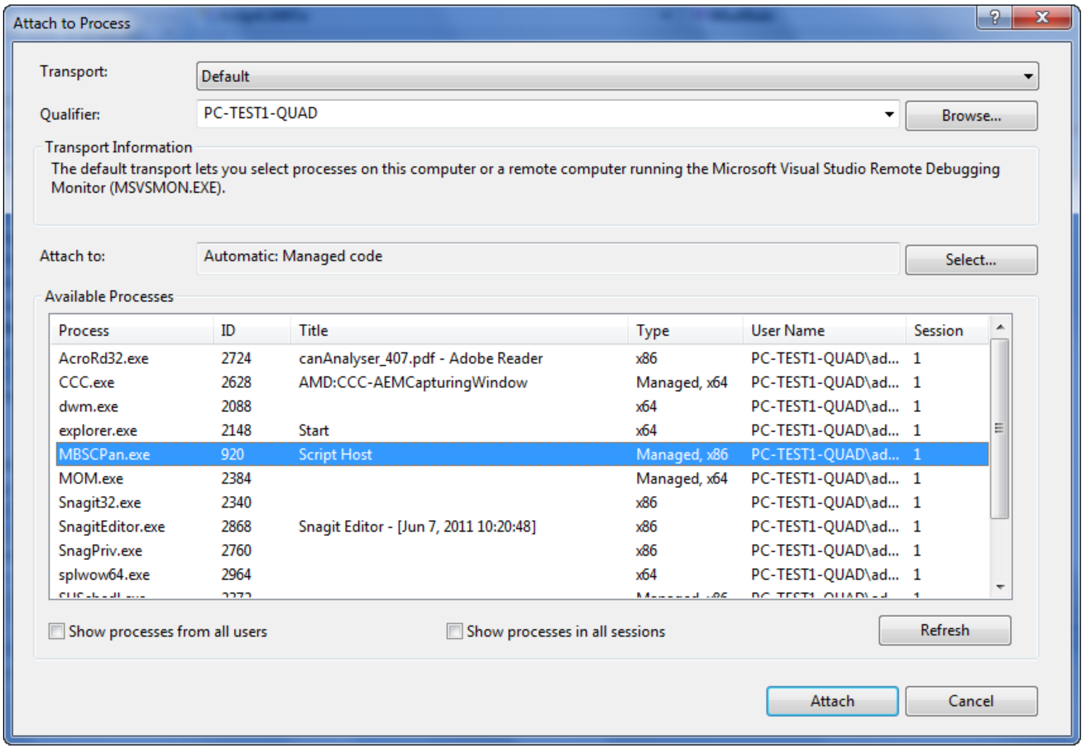
- After the Script Host displayed the dialog shown by Fig. 5.69 start Visual Studio 2008 and execute menu item **Tools | Attach to Process...**.
- In the displayed **Attach to Process** dialog (Fig. 5.70) select the canAnalyser process (MbsCPan.exe), and simply press the **Attach** button.
- Load the script file into Visual Studio and set the debug breakpoints.
- Resume script execution by clicking the OK button in the Script Host dialog (Fig. 5.69).

The script will be executed up to your breakpoint. From there you are able to debug the script code. Modification at the script are not adopted until you restart the script execution within the Script Host.

5.11.8 Menu reference

File menu

Menu point	Function
Load...	Loads a script
Unload	Unloads the selected script
Exit	Exits the Script Host



Attach to Process

Transport: Default

Qualifier: PC-TEST1-QUAD

Transport Information
The default transport lets you select processes on this computer or a remote computer running the Microsoft Visual Studio Remote Debugging Monitor (MSVSMON.EXE).

Attach to: Automatic: Managed code

Available Processes

Process	ID	Title	Type	User Name	Session
AcroRd32.exe	2724	canAnalyser_407.pdf - Adobe Reader	x86	PC-TEST1-QUAD\ad...	1
CCC.exe	2628	AMD:CCC-AEMCapturingWindow	Managed, x64	PC-TEST1-QUAD\ad...	1
dwm.exe	2088		x64	PC-TEST1-QUAD\ad...	1
explorer.exe	2148	Start	x64	PC-TEST1-QUAD\ad...	1
MBSCPan.exe	920	Script Host	Managed, x86	PC-TEST1-QUAD\ad...	1
MOM.exe	2384		Managed, x64	PC-TEST1-QUAD\ad...	1
Snagit32.exe	2340		x86	PC-TEST1-QUAD\ad...	1
SnagitEditor.exe	2868	Snagit Editor - [Jun 7, 2011 10:20:48]	x86	PC-TEST1-QUAD\ad...	1
SnagPriv.exe	2760		x86	PC-TEST1-QUAD\ad...	1
splwow64.exe	2964		x64	PC-TEST1-QUAD\ad...	1

□ Show processes from all users

□ Show processes in all sessions

Figure 5.70: Attaching debugger to the canAnalyser

Script menu

Menu point	Function
Execute	Executes the selected script
Debug	Executes the selected script with debug information (Chapter 5.11.7)
Abort	Aborts execution of the selected script
Auto Execution	Automatically execute the selected script when loading the analysis configuration the next time
Edit	Edit the selected script. Therefore the editor registered at the operating system is opened.

Reference menu

Menu point	Function
New	Add a reference to the selected script
Remove	Remove selected reference
Edit	Edit selected reference

Help menu

Menu item	Function
Help Topics	Opens the online help
About...	Opens the display of the version information

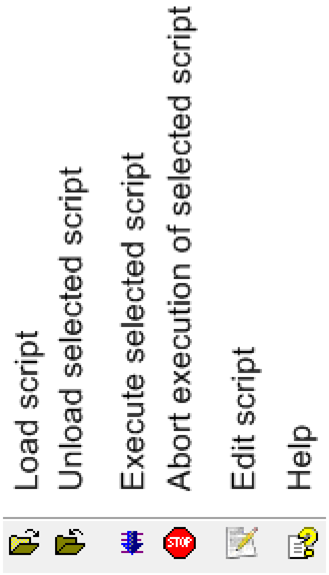


Figure 5.71: Toolbar of the Script Host

5.11.9 Toolbar

The most important functions of the Script Host module can also be called via the toolbar (Fig. 5.71).

5.11.10 Hotkeys

F5	Run script
Ctrl+F5	Debug script
Shift+F5	Stop script
Del	Remove reference
F2	Edit reference
F1	Online-Help

Appendix A

Export

A.1 Export of CSV files

Many export opportunities within canAnalyser create CSV files (comma separated value). This text based format is suitable to export tabular data and could be read by most spreadsheet applications. Nevertheless there are some differences which are subject of this chapter.

A.1.1 CSV format used by canAnalyser

The list separator character, which is language dependant and could be altered in the Windows ® control panel (via language settings), is used in all exports to separate columns. Lines are delimited by carriage return/line feed. Cell data is surrounded by quotation marks ("). Quotation marks within cell data are replaced by an escape sequence ("").

A.1.2 Import in Microsoft ® Excel

CSV files could be imported into excel by selecting the file type "Text files" within the "File open" dialog. Depending on the file extension (.csv or .txt) of the selected file Excel uses different import filters.
Files with the extension ".csv" will be imported by Excel without further interaction with the user. Excel is trying to determine the format of the cell data automatically. This behaviour could lead to undesirable results. One small example:
Enter "3e0" in a Excel table and export it as CSV file. After you reimport the CSV the cell contains the value "3,00E+00". This is because Excel interprets "3e0" as a floating point number on import.
The Excel CSV import uses the language dependant list separator character, from the system settings to determine column boundaries.
While importing files with extension ".txt" Excel opens the Text import dialog. Within this dialog you can fine tune the import settings. You could use other column separator or field separator characters or set the data type per column manually. The following parameters could be used to import files exported by canAnalyser:

- Separated - characters separate fields
- Separator - semicolon (;), comma (,) or other, depends on the system language setting during export
- If columns contains hexadecimal numbers you should set the column type to "Text" or else specific hexadecimal numbers will be interpreted as floating point numbers.

Another characteristic with Excel is the Drag&Drop behaviour: If you Drag a CSV file onto an Excel instance, files with ".csv" extension are treated as if opened via file open. But if the file has the extension ".txt" the content of the file is copied line by line into the first column of the Excel sheet without opening the text import dialog.

A.1.3 Import in OpenOffice/LibreOffice

When importing files with extension ".csv" into OpenOffice the text import dialog is displayed automatically. Within this dialog you could set all necessary parameters:

- Separated - characters separate fields
- Separator - semicolon (;), comma (,) or other, depends on the system language setting during export
- If columns contains hexadecimal numbers you should set the column type to "Text" or else specific hexadecimal numbers will be interpreted as floating point numbers.

Files with extension ".txt" will be treated as text files and opened via OpenOffice Writer, if you have not selected the CSV import filter explicitely. Because of this Drag&Drop works only for files with extension ".csv".

Appendix B

Definitions

B.1 Definitions, acronyms, abbreviations

Bitrate Transmission rate in bits/sec. with which a bus is operated.

CAN Controller Area Network

CAN status In order not to block a CAN network with a defective node, CAN controllers have internal error counters. If these error counters exceed a certain limit, the status of the CAN controller changes to the warning level. If a further level is exceeded, the node is switched off by the bus (Bus off).

Data Frame Standard data telegram of the CAN bus. A data frame consists of an 11 or 29 bit wide identifier (COBID), a data field of between 0 and 8 bytes and protocol information such as RTR flag and DLC (data length code).

Database editor Application to create and alter databases on which the interpretation of layer-2 messages is based.

Error frame Special telegram for error signalling on the CAN bus

FIBEX Field Bus Exchange Format - Fibex is an XML exchange format proposed for data exchange between tools that deal with message-oriented bus communication systems.The FIBEX specification document is downloadable from the web page of ASAM e.V. (Association for Standardisation of Automation- and Measuring Systems) on `http://www.asam.net`.

Filter Module to select or exclude messages according to certain criteria for display or trace.

FlexRay FlexRay is a fast, deterministic and fault-tolerant bus system, developed for automotive use.

FlexRay CCM IXXAT PC-Interface for FlexRay and CAN

Online mode Recording or display of messages immediately after reception without further processing.

Remote frame CAN request telegram. Special telegram format without data field to request a data telegram

RTR RemoteTransmitRequest: The RTR-bit within a CAN message distinguishes between data telegrams and data request telegrams

Standard/Extended The CAN bus supports two message formats, which differ in the length of the identifier. Standard with 11-bit identifier and extended with 29-bit identifier.

Trace Recording of messages in a file

Trace file A recording carried out of layer-2 messages, which can be saved as a binary or text file, and which can then be evaluated

Trigger Event used to start/stop a recording (Trace).

TX-echo Mode in which the canAnalyser also receives messages which it has transmitted itself.

TX-passive Mode in which active access to the bus is prevented by hardware. Neither acknowledge nor errors can be terminated. The canAnalyser is only a listener.

VCI Universal CAN driver for all PC/CAN boards of IXXAT

Appendix C

Copyrights

C.1 Copyrights

C.1.1 Copyright

C.1.2 Additional Copyrights

Dundas software

This software contains material that is © 1994-2000 DUNDAS SOFTWARE LTD., all rights reserved.

Microsoft

This software installs or updates Microsoft OS components (MSXML3 SP5) which are copyrighted by © Microsoft Corp.

Apache Software Foundation

This product includes software developed by The Apache Software Foundation (`http://www.apache.org/`). Portions of this software was originally based on the following:

- software copyright (c) 1999, IBM Corporation., `http://www.ibm.com`.

Castle windsor

Apache License, Version 2.0, January 2004

`http://www.apache.org/licenses/`

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

- "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

- "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.
- "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.
- "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.
- "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.
- "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.
- "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).
- "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.
- "Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."
- "Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and

2. You must cause any modified files to carry prominent notices stating that You changed the files; and

3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any partof the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Lua.org, PUC-Rio

License for Lua 5.0 and later versions
Copyright © 1994-2010 Lua.org, PUC-Rio.
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

OxyPlot

OpenTK

The Open Toolkit library license

Third parties

OpenTK.Platform.Windows and OpenTK.Platform.X11 include portions of the Mono class library. These portions are covered by the following license:

Copyright (c) 2003-2007 Tao Framework Team OpenTK.Compatibility includes portions of the Tao Framework library (Tao.OpenGl, Tao.OpenAl and Tao.Platform.Windows.SimpleOpenGlControl). These portions are covered by the following license:

OpenTK.Half offers Half-to-Single and Single-to-Half conversions based on OpenEXR source code, which is covered by the following license: