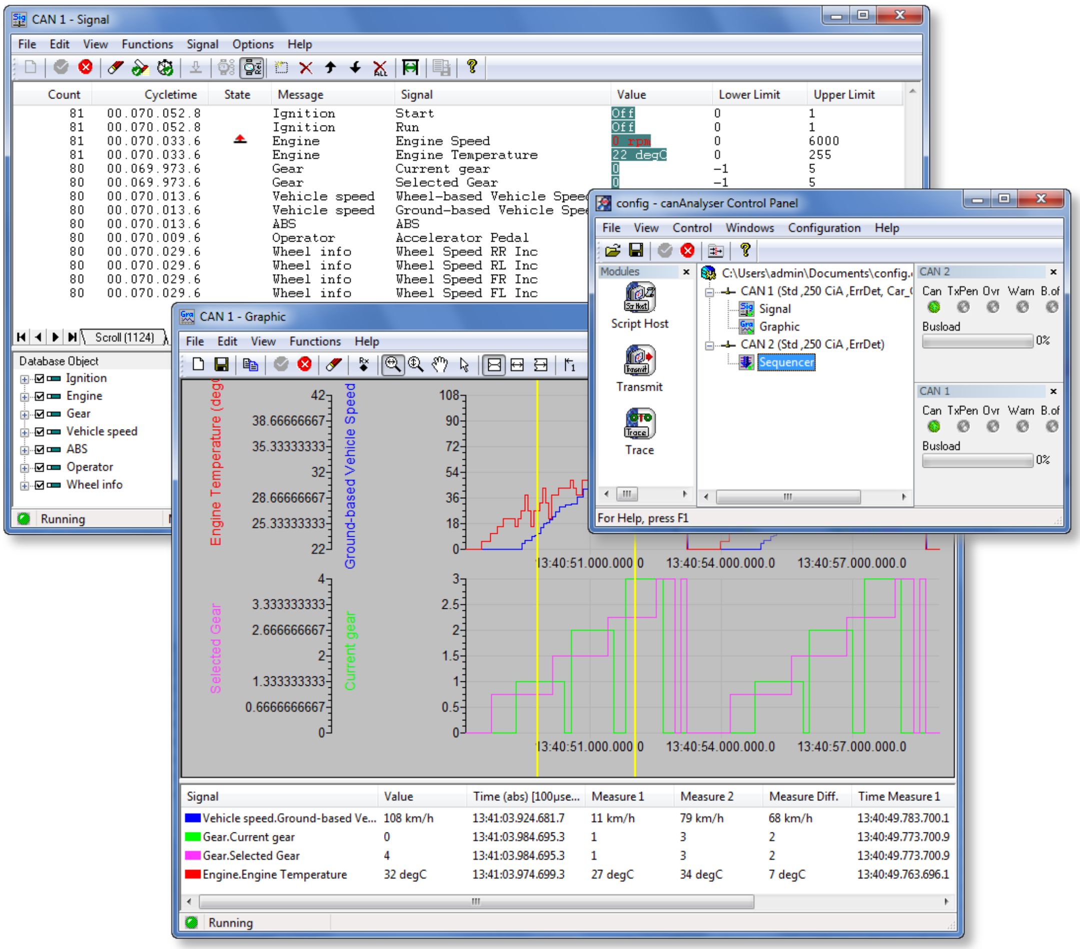
Manual

canAnalyser

Version 2



Count	Cycletime	State	Message	Signal	Value	Lower Limit	Upper Limit
81	00.070.052.8		Ignition	Start	Off	0	1
81	00.070.052.8		Ignition	Run	Off	0	1
81	00.070.033.6	▲	Engine	Engine Speed	0 rpm	0	6000
81	00.070.033.6		Engine	Engine Temperature	22 degC	0	255
80	00.069.973.6		Gear	Current gear	0	-1	5
80	00.069.973.6		Gear	Selected Gear	0	-1	5
80	00.070.013.6		Vehicle speed	Wheel-based Vehicle Speed			
80	00.070.013.6		Vehicle speed	Ground-based Vehicle Spee			
80	00.070.013.6		ABS	ABS			
80	00.070.009.6		Operator	Accelerator Pedal			
80	00.070.029.6		Wheel info	Wheel Speed RR Inc			
80	00.070.029.6		Wheel info	Wheel Speed RL Inc			
80	00.070.029.6		Wheel info	Wheel Speed FR Inc			
80	00.070.029.6		Wheel info	Wheel Speed FL Inc			

Signal	Value	Time (abs) [100µse...	Measure 1	Measure 2	Measure Diff.	Time Measure 1
🟦 Vehicle speed.Ground-based Ve...	108 km/h	13:41:03.924.681.7	11 km/h	79 km/h	68 km/h	13:40:49.783.700.1
🟩 Gear.Current gear	0	13:41:03.984.695.3	1	3	2	13:40:49.773.700.9
🟪 Gear.Selected Gear	4	13:41:03.984.695.3	1	3	2	13:40:49.773.700.9
🟥 Engine.Engine Temperature	32 degC	13:41:03.974.699.3	27 degC	34 degC	7 degC	13:40:49.763.696.1

IXXAT

The expert for industrial and automotive communication

IXXAT

Headquarter	**US Sales Office**
IXXAT Automation GmbH Leibnizstr. 15 D-88250 Weingarten	IXXAT Inc. 120 Bedford Center Road USA-Bedford, NH 03110
Tel.: +49 (0)7 51 / 5 61 46-0 Fax: +49 (0)7 51 / 5 61 46-29 Internet: www.ixxat.de e-Mail: info@ixxat.de	Phone: +1-603-471-0800 Fax: +1-603-471-0880 Internet: www.ixxat.com e-Mail: sales@ixxat.com

Support

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

Fax: +49 (0)7 51 / 5 61 46-29
e-Mail: support@ixxat.de

Copyright

Document No. 1.01.0133-166-Manual-E-V2.80

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 (only possible with canAnalyser standard). 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 process data (Signal module)
- Time-synchronous analysis of several buses
- Display of bus load, number of Error Frames and identifier-related message distribution (Statistic module, CAN)

- Emulation of nodes and protocol sequences by processing command-controlled message sequences (Sequencer module, CAN)
- Display of interpreted message contents in plaintext and in diagrams (Graphic module, CAN)
- Data modification and cycle time monitoring

The hardware and the analysis environment are configured via the control panel. The configuration data can be saved and loaded. In addition to the hardware parameters, the configuration also contains the type, number and specific settings of the analysis modules.

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 database to be used. The project database contains, for example, 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
- Error protocol window

The following information is shown in the configuration tree:

- Name of the loaded configuration
- Available controllers of the selected card with its settings
- Assignment of the analysis modules to the individual controllers

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.

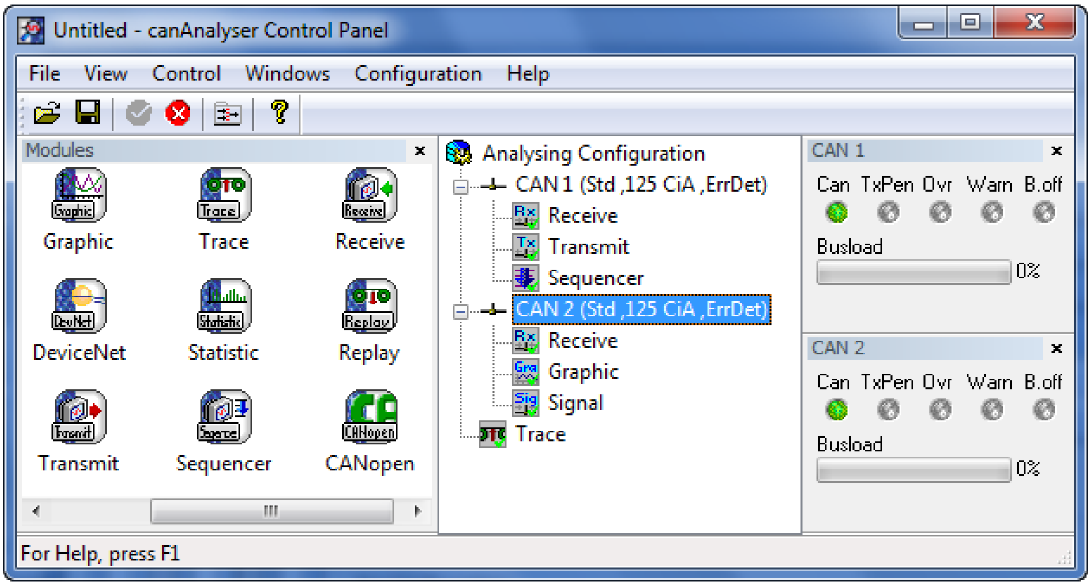


Figure 1.1: Control panel of the canAnalyser

Receiving messages

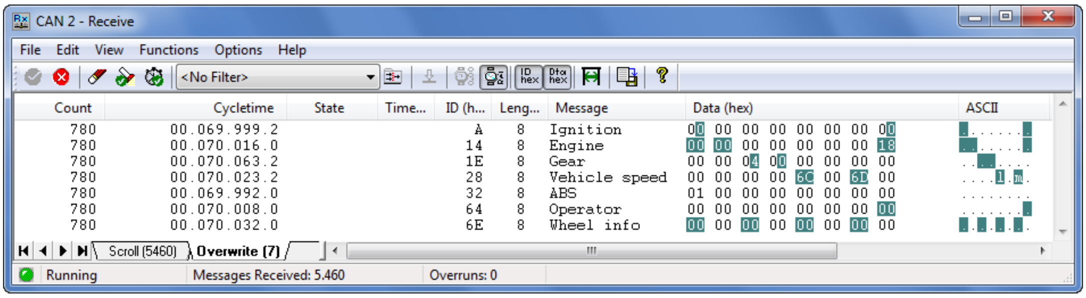
The Receive module provides the following analysis functions:

- 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 database is displayed in each analysis module. Fig. 1.2 shows a Receive module in overwrite mode.

Transmitting messages

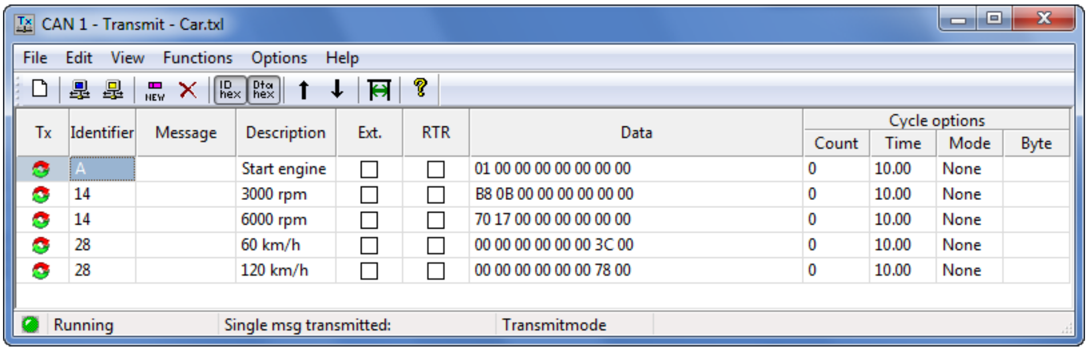
With the Transmit module (Fig. 1.3) transmit messages can be specified and transmitted individually or cyclically. With cyclic transmission, the number of messages, their cycle time and an increment can be defined.



Count	Cycletime	State	Time...	ID (h...	Leng...	Message	Data (hex)	ASCII
780	00.069.999.2			A	8	Ignition	00 00 00 00 00 00 00 00	....
780	00.070.016.0			14	8	Engine	00 00 00 00 00 00 00 18	....
780	00.070.063.2			1E	8	Gear	00 00 04 00 00 00 00 00	....
780	00.070.023.2			28	8	Vehicle speed	00 00 00 00 6C 00 6D 00	....l.m.
780	00.069.992.0			32	8	ABS	01 00 00 00 00 00 00 00	....
780	00.070.008.0			64	8	Operator	00 00 00 00 00 00 00 00	....
780	00.070.032.0			6E	8	Wheel info	00 00 00 00 00 00 00 00	....

Running | Messages Received: 5.460 | Overruns: 0

Figure 1.2: Receive module in overwrite mode



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	10.00	None	
	14		3000 rpm	□	□	B8 0B 00 00 00 00 00 00	0	10.00	None	
	14		6000 rpm	□	□	70 17 00 00 00 00 00 00	0	10.00	None	
	28		60 km/h	□	□	00 00 00 00 00 00 3C 00	0	10.00	None	
	28		120 km/h	□	□	00 00 00 00 00 00 78 00	0	10.00	None	

Running | Single msg transmitted: | Transmitmode

Figure 1.3: Transmit module

Message recording (Trace) from several buses

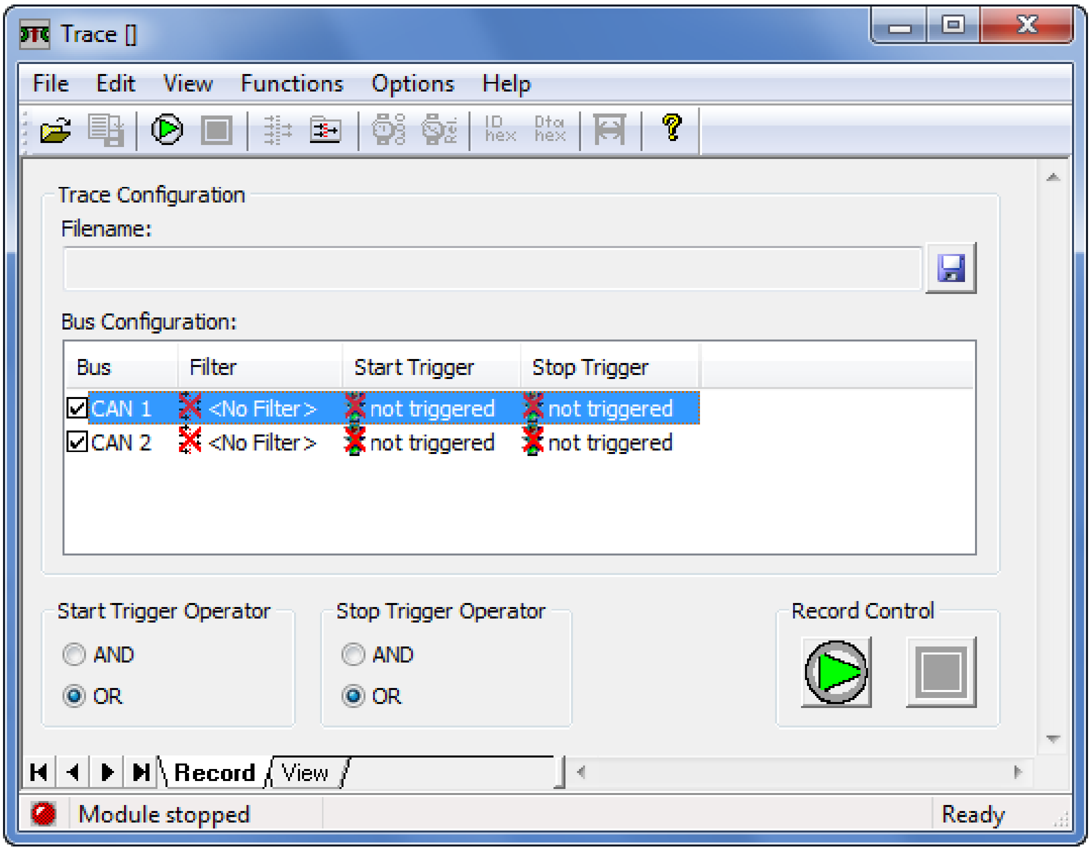
The Trace module enables parallel message recording from several buses onto the hard disk. Message recording (Trace) is carried out for the configured bus systems in separate files. After recording, the messages are displayed in relation to one another in terms of time and color-coded.
Control and configuration of the trace (Fig. 1.4) and the display (Fig. 1.5) 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 a database, which must be specified when the Analysis environment is configured. A database can be created with the Database editor. 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. 1.6). In overwrite mode, the display of signal value modifications and monitoring of the cycle time are supported.



Trace Configuration

Filename:

Bus Configuration:

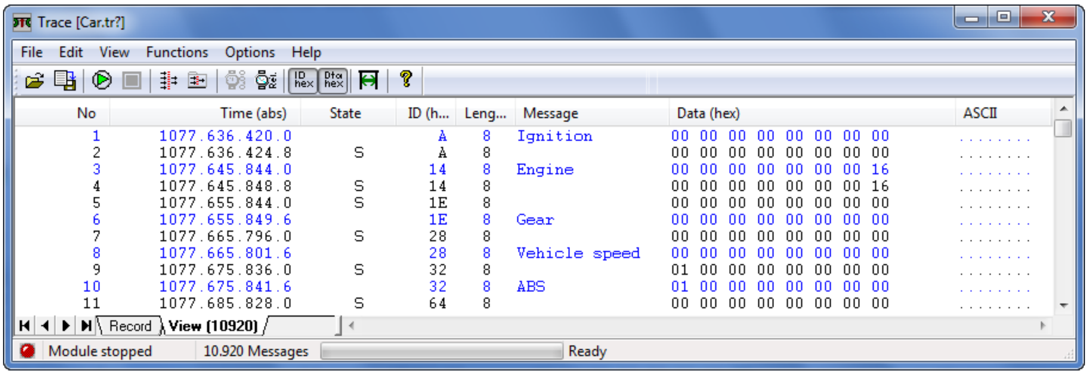
Bus	Filter	Start Trigger	Stop Trigger
☑ CAN 1	<No Filter>	not triggered	not triggered
☑ CAN 2	<No Filter>	not triggered	not triggered

Start Trigger Operator: ○ AND ● OR

Stop Trigger Operator: ○ AND ● OR

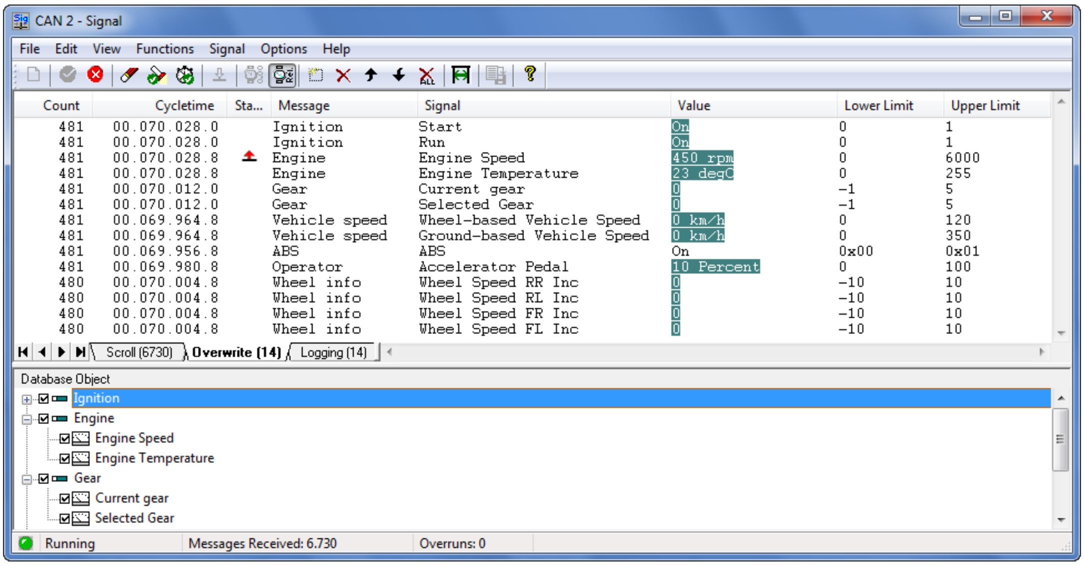
Record Control

Figure 1.4: Trace module, index card record



No	Time (abs)	State	ID (h...	Leng...	Message	Data (hex)	ASCII
1	1077.636.420.0		A	8	Ignition	00 00 00 00 00 00 00 00	....
2	1077.636.424.8	S	A	8		00 00 00 00 00 00 00 00	....
3	1077.645.844.0		14	8	Engine	00 00 00 00 00 00 00 16	....
4	1077.645.848.8	S	14	8		00 00 00 00 00 00 00 16	....
5	1077.655.844.0	S	1E	8		00 00 00 00 00 00 00 00	....
6	1077.655.849.6		1E	8	Gear	00 00 00 00 00 00 00 00	....
7	1077.665.796.0	S	28	8		00 00 00 00 00 00 00 00	....
8	1077.665.801.6		28	8	Vehicle speed	00 00 00 00 00 00 00 00	....
9	1077.675.836.0	S	32	8		01 00 00 00 00 00 00 00	....
10	1077.675.841.6		32	8	ABS	01 00 00 00 00 00 00 00	....
11	1077.685.828.0	S	64	8		00 00 00 00 00 00 00 00	....

Figure 1.5: Trace module, index card view



Count	Cycletime	Sta...	Message	Signal	Value	Lower Limit	Upper Limit
481	00.070.028.0		Ignition	Start	On	0	1
481	00.070.028.0		Ignition	Run	On	0	1
481	00.070.028.8	▲	Engine	Engine Speed	450 rpm	0	6000
481	00.070.028.8		Engine	Engine Temperature	23 degC	0	255
481	00.070.012.0		Gear	Current gear	0	-1	5
481	00.070.012.0		Gear	Selected Gear	0	-1	5
481	00.069.964.8		Vehicle speed	Wheel-based Vehicle Speed	0 km/h	0	120
481	00.069.964.8		Vehicle speed	Ground-based Vehicle Speed	0 km/h	0	350
481	00.069.956.8		ABS	ABS	On	0x00	0x01
481	00.069.980.8		Operator	Accelerator Pedal	10 Percent	0	100
480	00.070.004.8		Wheel info	Wheel Speed RR Inc	0	-10	10
480	00.070.004.8		Wheel info	Wheel Speed RL Inc	0	-10	10
480	00.070.004.8		Wheel info	Wheel Speed FR Inc	0	-10	10
480	00.070.004.8		Wheel info	Wheel Speed FL Inc	0	-10	10

Scroll (6730) | **Overwrite (14)** | Logging (14)

Database Object

- ☑ Ignition
- ☑ Engine
  - ☑ Engine Speed
  - ☑ Engine Temperature
- ☑ Gear
  - ☑ Current gear
  - ☑ Selected Gear

● Running | Messages Received: 6.730 | Overruns: 0

Figure 1.6: Signal module, overwrite mode

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.

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.

Graphic display of the bus load and identifier distribution

With the Statistic module (Fig. 1.7) it possible to display the variation over time of the bus load, the message distribution and the total number of objects graphically.

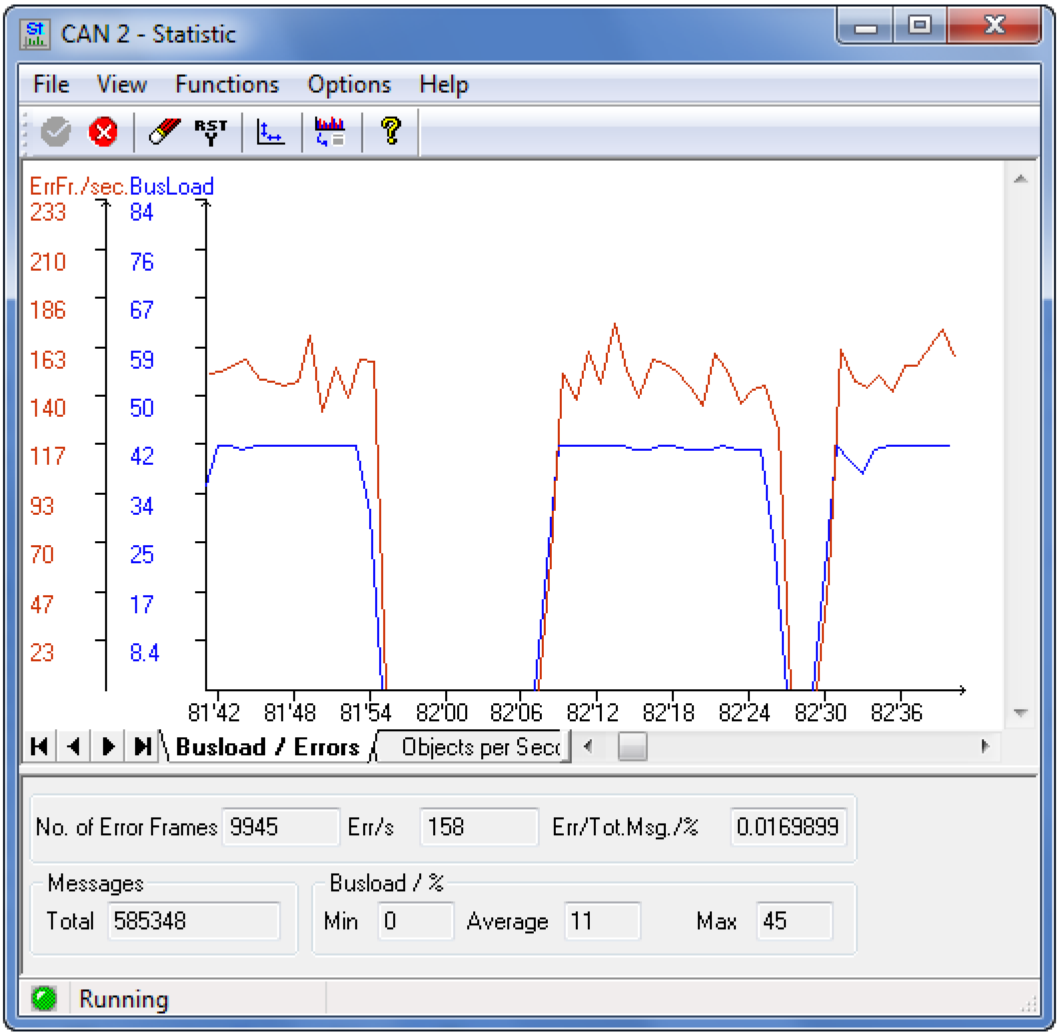
Emulation of nodes or protocols by running sequences

The Sequencer module (Fig. 1.8) 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 Graphic module (Fig. 1.9) supports the display of signals in the form of y-t diagrams (line writers). Interpretation is carried out as in the Signal module based on the project database specified in the control panel.



No. of Error Frames 9945 Err/s 158 Err/Tot.Msg./% 0.0169899

Messages

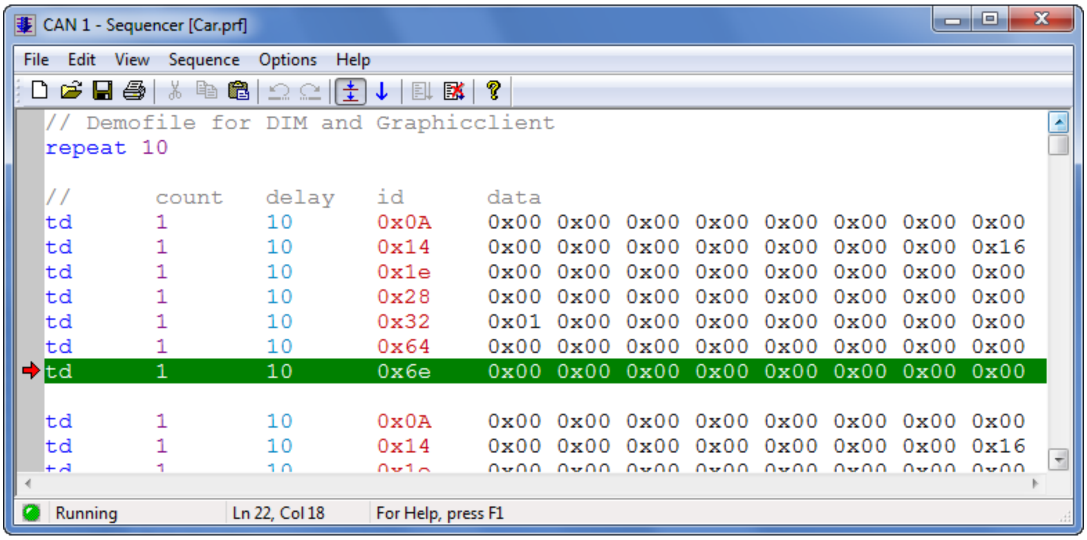
Total 585348

Busload / %

Min 0 Average 11 Max 45

Running

Figure 1.7: Statistic module


CAN 1 - Sequencer [Car.prf]
File Edit View Sequence Options Help

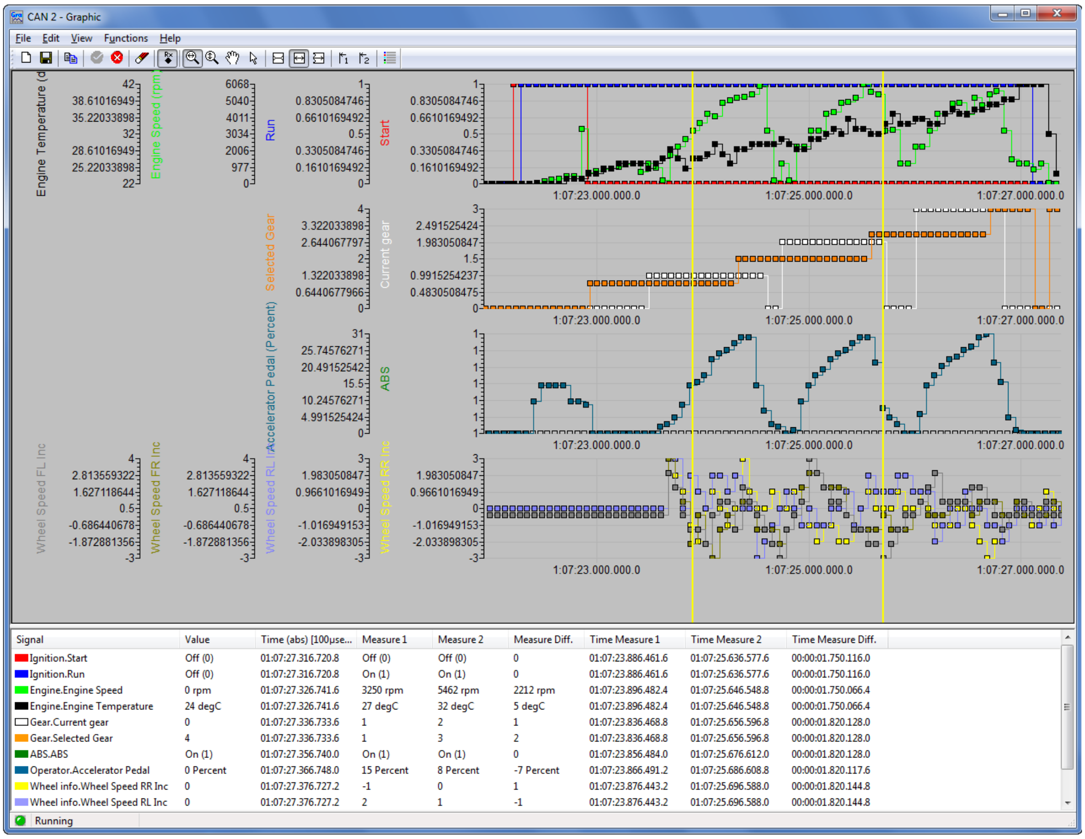
```
// Demofile for DIM and Graphicclient
repeat 10

//      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
```

Running Ln 22, Col 18 For Help, press F1

Figure 1.8: Sequencer module



Signal	Value	Time (abs) [100µse...	Measure 1	Measure 2	Measure Diff.	Time Measure 1	Time Measure 2	Time Measure Diff.
🟥 Ignition.Start	Off (0)	01:07:27.316.720.8	Off (0)	Off (0)	0	01:07:23.886.461.6	01:07:25.636.577.6	00:00:01.750.116.0
🟦 Ignition.Run	Off (0)	01:07:27.316.720.8	On (1)	On (1)	0	01:07:23.886.461.6	01:07:25.636.577.6	00:00:01.750.116.0
🟩 Engine.Engine Speed	0 rpm	01:07:27.326.741.6	3250 rpm	5462 rpm	2212 rpm	01:07:23.896.482.4	01:07:25.646.548.8	00:00:01.750.066.4
⬛ Engine.Engine Temperature	24 degC	01:07:27.326.741.6	27 degC	32 degC	5 degC	01:07:23.896.482.4	01:07:25.646.548.8	00:00:01.750.066.4
□ Gear.Current gear	0	01:07:27.336.733.6	1	2	1	01:07:23.836.468.8	01:07:25.656.596.8	00:00:01.820.128.0
🟧 Gear.Selected Gear	4	01:07:27.336.733.6	1	3	2	01:07:23.836.468.8	01:07:25.656.596.8	00:00:01.820.128.0
🟩 ABS.ABS	On (1)	01:07:27.356.740.0	On (1)	On (1)	0	01:07:23.856.484.0	01:07:25.676.612.0	00:00:01.820.128.0
🟦 Operator.Accelerator Pedal	0 Percent	01:07:27.366.748.0	15 Percent	8 Percent	-7 Percent	01:07:23.866.491.2	01:07:25.686.608.8	00:00:01.820.117.6
🟨 Wheel info.Wheel Speed RR Inc	0	01:07:27.376.727.2	-1	0	1	01:07:23.876.443.2	01:07:25.696.588.0	00:00:01.820.144.8
🟪 Wheel info.Wheel Speed RL Inc	0	01:07:27.376.727.2	2	1	-1	01:07:23.876.443.2	01:07:25.696.588.0	00:00:01.820.144.8

Figure 1.9: Graphic module

Integrating own analysis modules

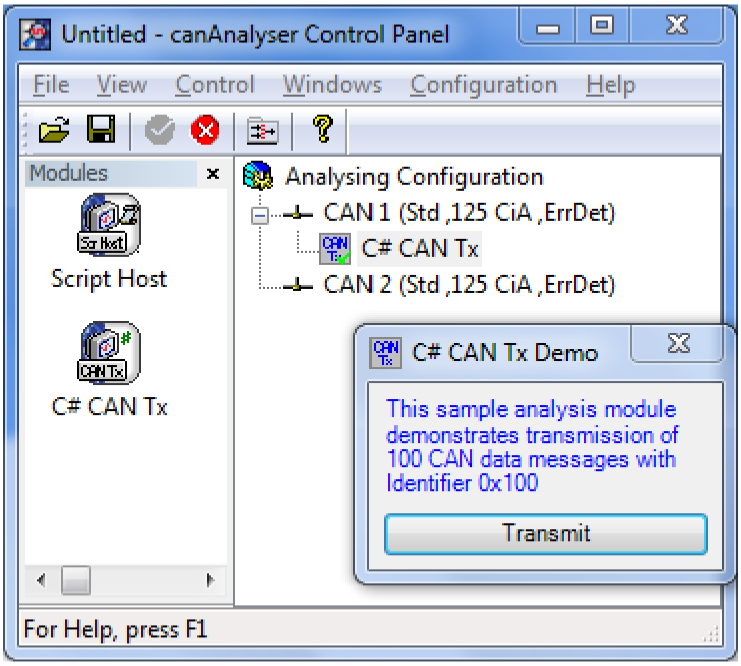
Via the open .NET programming interface the user has the possibility to extend the canAnalyser by own modules and user interfaces. Own, autonomous, on .NET Framework V1.1 based modules can be written by using common Windows development environments (e.g. Visual Studio .NET, Delphi) and can then be integrated to the canAnalyser. Consequently it's possible to create user interfaces for own systems respectively for devices and tools with system specific analysis functions.

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. 1.10).

Executing scripts

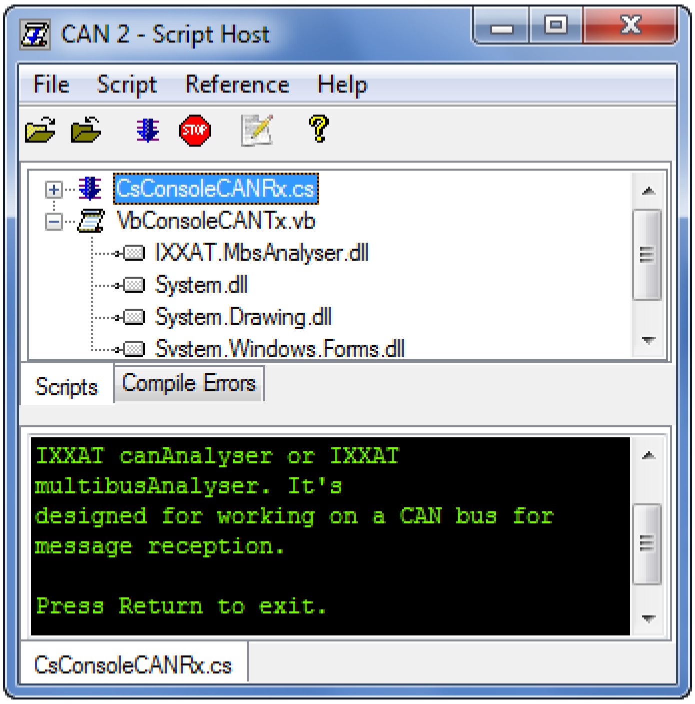
Because creation and modification of scripts is very flexible and cost effective they ease off the work of developers during the testing phase as well as searching errors by service engineers on-site. At this it's not mandatory having an installed development environment and each modification can be tested immediately.

For configuring and executing scripts the Control Panel provides a Script Host as analysis module within the Modules window (Fig. 1.11). 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).


Untitled - canAnalyser Control Panel
File View Control Windows Configuration Help
Modules
Script Host
C# CAN Tx
Analysing Configuration
CAN 1 (Std ,125 CiA ,ErrDet)
C# CAN Tx
CAN 2 (Std ,125 CiA ,ErrDet)
C# CAN Tx Demo
Transmit
For Help, press F1

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

Figure 1.10: Control Panel with "C# CAN Tx" sample analysis module



```
IXXAT canAnalyser or IXXAT
multibusAnalyser. It's
designed for working on a CAN bus for
message reception.

Press Return to exit.
```

Figure 1.11: 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 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
Two-channel capability	Only one channel can be analysed	Simultaneous analysis of two channels possible (this must be provided by the hardware driver)
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, Statistic, Replay	Receive, Transmit, Trace, Sequencer, Statistic, Replay, Signal, Graphic
Optional modules	CANopen, DeviceNet, J1939	CANopen, DeviceNet, J1939

Chapter 3

Installation and start-up

3.1 System requirements

To operate the canAnalyser, the following system requirements must be fulfilled:

- x86 compatible processor with minimum 800 MHz or higher
- Windows 2000, Windows XP, Windows Vista, Windows 7
- At least 256 MB RAM, preferably 512 MB
- Installed IXXAT VCI-driver, version 2.16 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. canAna28_std_3240.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

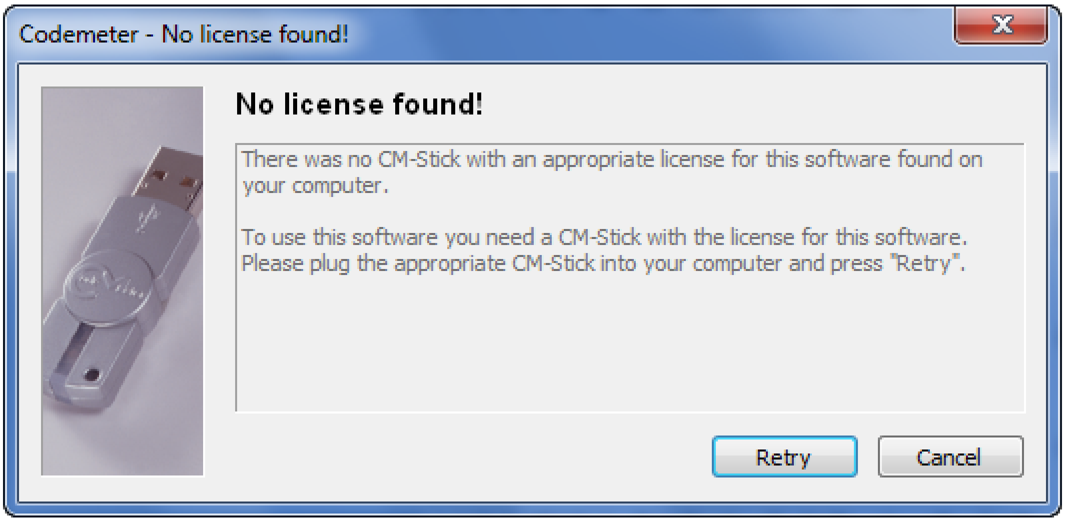
The installation of the CodeMeter user runtime is integrated into the canAnalyser installation.
However, if you want to uninstall the canAnalyser you have to uninstall the CodeMeter user runtime separately.
The CodeMeter Runtime Kit can be uninstalled easily on Microsoft Windows operating systems. Just 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

During operation of the canAnalyser the CodeMeter stick with the appropriate license has to be plugged into the USB port. Otherwise a dialog with the appropriate error message (Fig. 4.1) will pop up and the canAnalyser will be locked until the license is accessible.

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.



Codemeter - No license found!

No license found!

There was no CM-Stick with an appropriate license for this software found on your computer.

To use this software you need a CM-Stick with the license for this software. Please plug the appropriate CM-Stick into your computer and press "Retry".

Figure 4.1: CodeMeter error dialog

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.

5.1 Control panel

5.1.1 Overview

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 the configured analysis modules.

5.1.2 Configuration selection dialog

When the control panel is started, a configuration selection dialog appears (Fig. 5.1). Here it is possible either to select and load an existing configuration or create a new configuration.

5.1.3 Board selection dialog

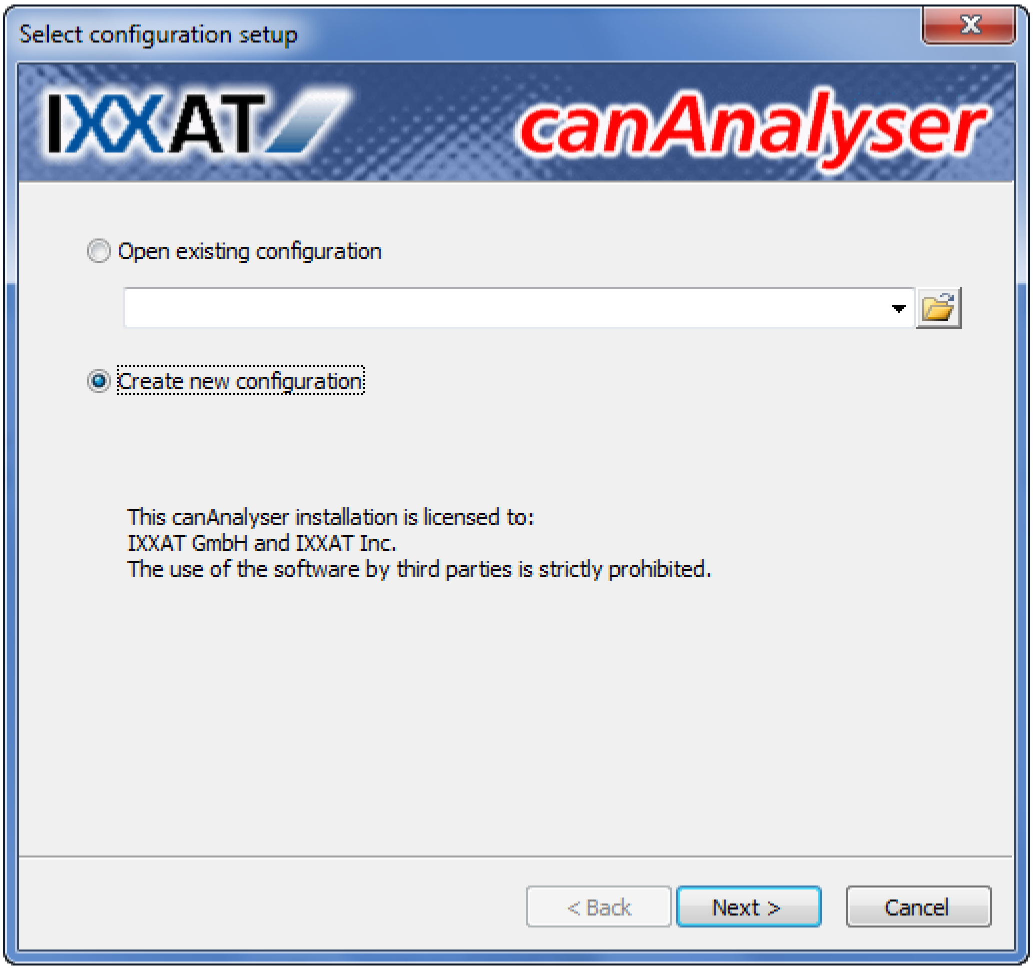
The board selection dialog (Fig. 5.2) appears when a new configuration is created or the configured hardware is no longer present or is not available. In this case the user can select an alternative hardware.
In the drop-down box all installed IXXAT boards can be selected. The selection has to be acknowledged with the "Finalize" button.

5.1.4 Start/Stop of the hardware

The analysis can be started via the toolbar. By clicking on the **Start** 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** 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 points **Control | Start Communication** and **Control | Stop Communication**.

5.1.5 Control panel main window

The program window of the control panel (Fig. 5.3) is sub-divided into the following areas:

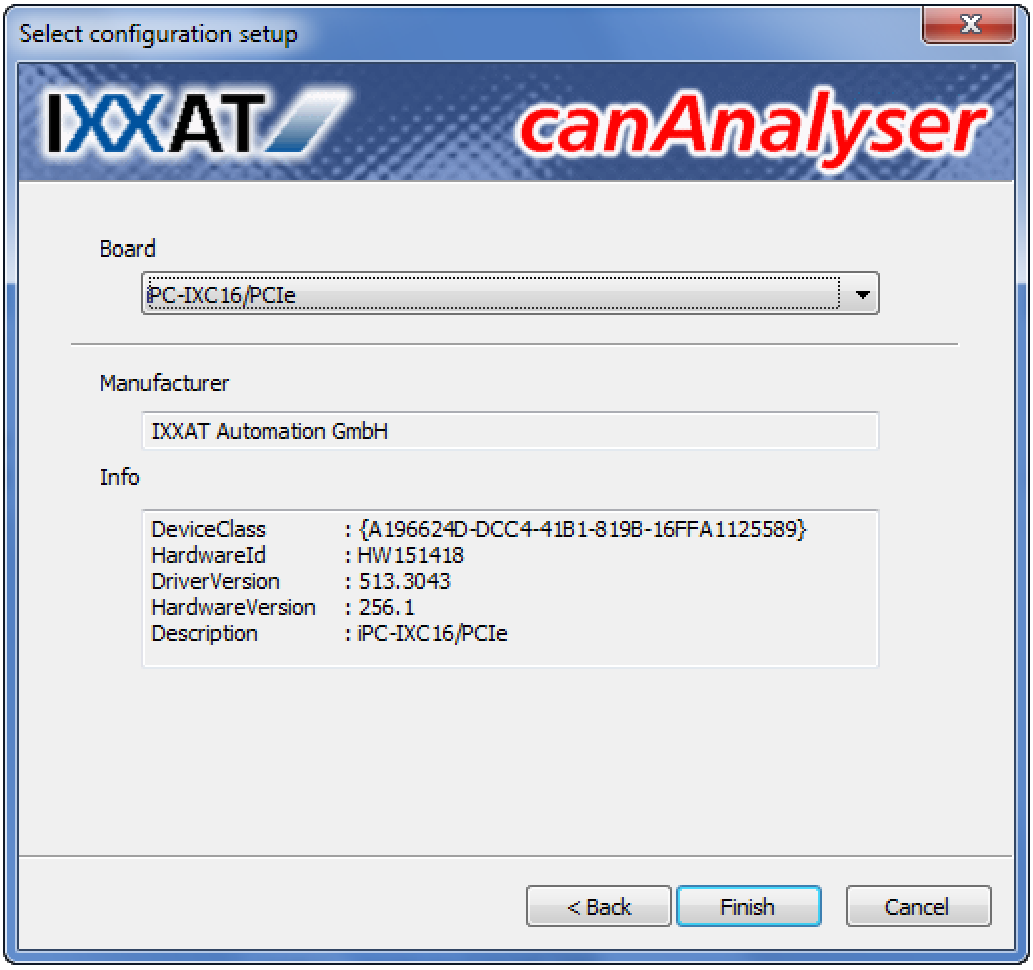


○ Open existing configuration

● Create new configuration

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

Figure 5.1: The configuration selection dialog of the control panel



Board

iPC-IXC16/PCIe

Manufacturer

IXXAT Automation GmbH

Info

```
DeviceClass      : {A196624D-DCC4-41B1-819B-16FFA1125589}
HardwareId       : HW151418
DriverVersion    : 513.3043
HardwareVersion  : 256.1
Description      : iPC-IXC16/PCIe
```

Figure 5.2: The board selection dialog of the control panel

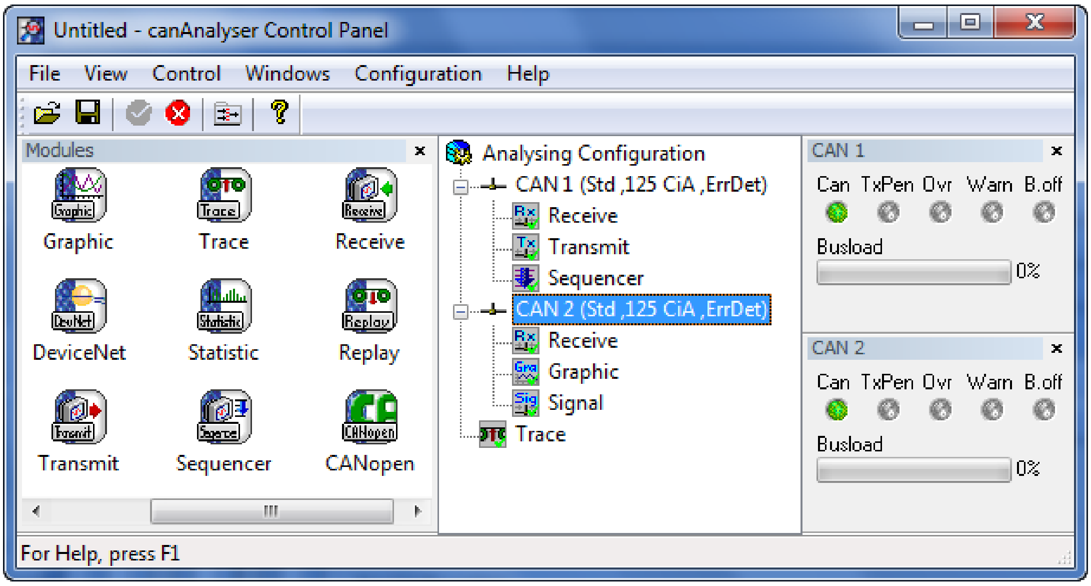


Figure 5.3: The visible fields of the control panel

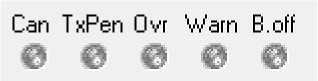


Figure 5.4: CAN status window

- 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

5.1.6 Status window

The control panel provides a status window for each available controller (Fig. 5.4). By right-clicking on a controller in the configuration tree, a pop-up menu (Fig. 5.10) appears, which enables the corresponding status window to be shown or hidden.

CAN status window

The CAN status window contains the following information:

Meaning	Display off	Display on
CAN	CAN controller is started	CAN controller is stopped
TxPen (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

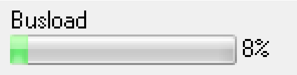


Figure 5.5: Display of bus load

Online analysis is only possible if controller is started.
After the first occurrence of a data overrun, the data overrun LED remains activated. It only goes out when the controller is stopped and restarted.
If the Bus Off LED is lit, 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.
The bus load display (Fig. 5.5) shows the load of the bus in per cent. The bus load measurement is a hardware function which is not supported by all interface boards. The bus load is only displayed if this is supported by the interface.

5.1.7 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 `Documents and Settings\UserName\AppData\IXXAT\canAnalyser\2.8\Log\ *\canAnalyse`
Use **View** menu to configure which event kinds should be shown in the Event Log window. Menu command **View | Clear Eventlog** empties the Event Log.

5.1.8 Adding of modules to the configuration

Insertion of modules using drag and drop

The module list (Fig. 5.6) 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/controller 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 module list

By hitting the **Insert** key, the **Menu** key or by right-clicking the module in the module list a menu (Fig. 5.7) is displayed which allows to add the selected module to a controller. The menu contains a list of all supported controllers.

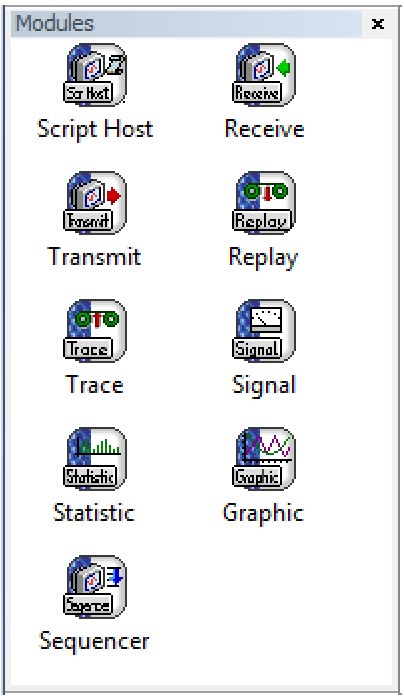


Figure 5.6: Control panel, module list

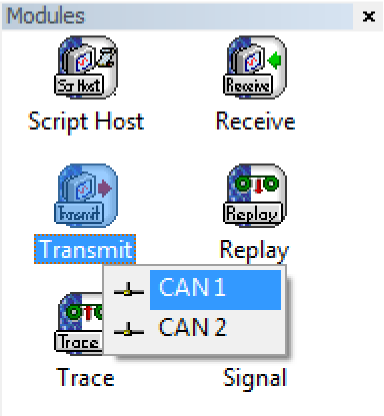


Figure 5.7: Pop-up menu of the module list to add modules to the configuration

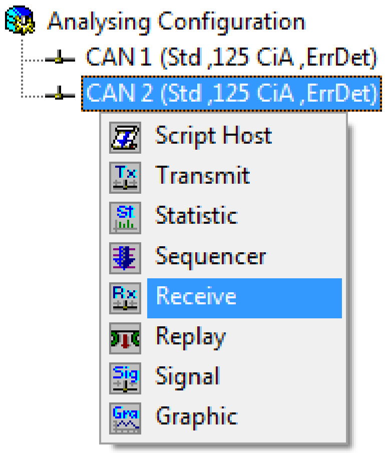


Figure 5.8: Pop-up menu of the configuration tree to add modules

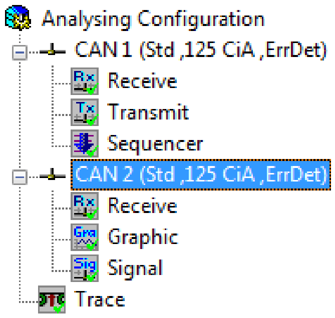


Figure 5.9: Configuration tree of the control panel

Insertion of modules using the menu of the configuration tree

By hitting the **Insert** key in the configuration tree a menu (Fig. 5.8) is displayed which allows to add a module to the selected controller. The menu contains a list of all supported modules.

5.1.9 Configuration tree

In the configuration tree (Fig. 5.9) the current analysis configuration is displayed hierarchically. An analysis configuration defines the following objects:

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

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. 5.10), which can be activated using the right mouse button. The following settings can be made:

- Setting the properties of the bus

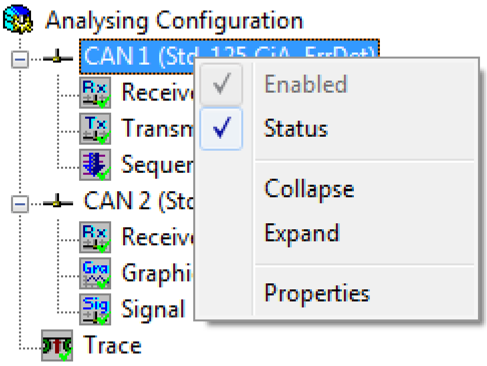
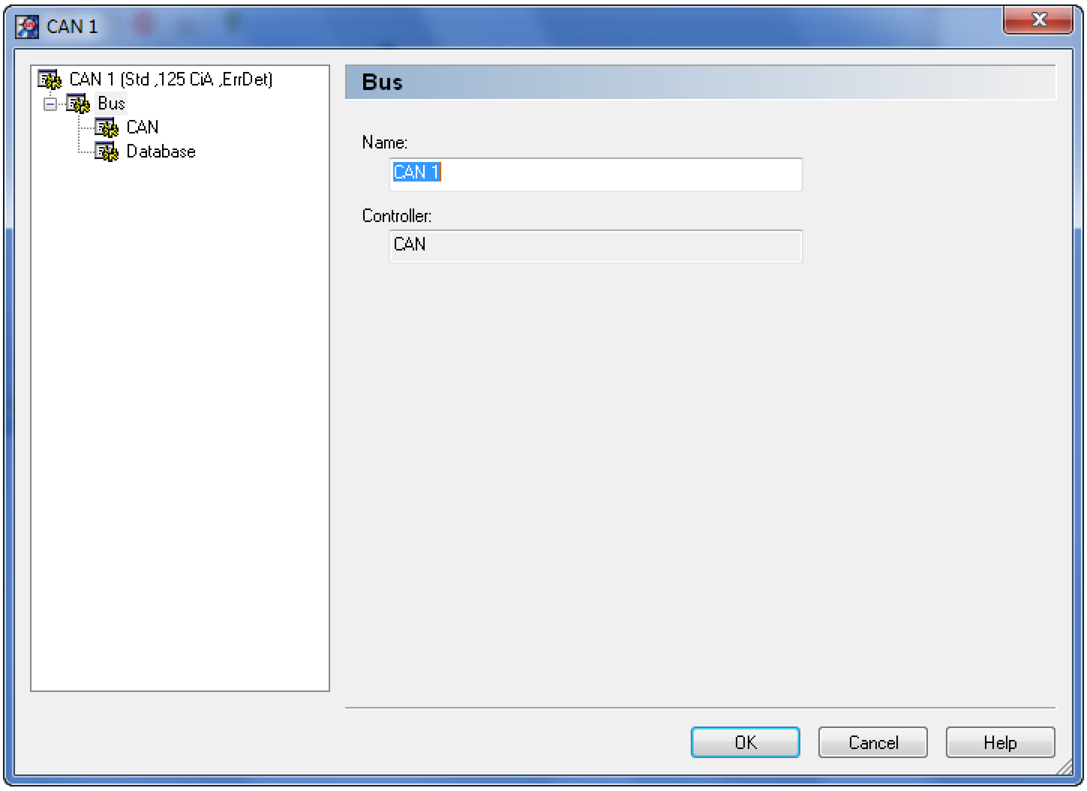


Figure 5.10: Pop-up menu of a CAN bus



Name: CAN 1

Controller: CAN

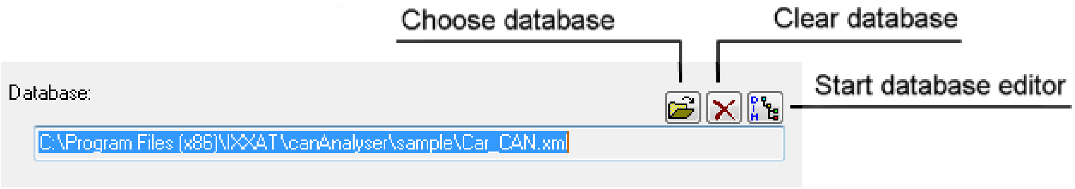
Figure 5.11: Setting a symbolic bus name

- 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.

Setting a symbolic bus name

In the entry field **Name** of the section **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.



Database:

C:\Program Files (x86)\IXXAT\canAnalyser\sample\Car_CAN.xml

Figure 5.12: Select/create interpretation database

Selection of an interpretation database

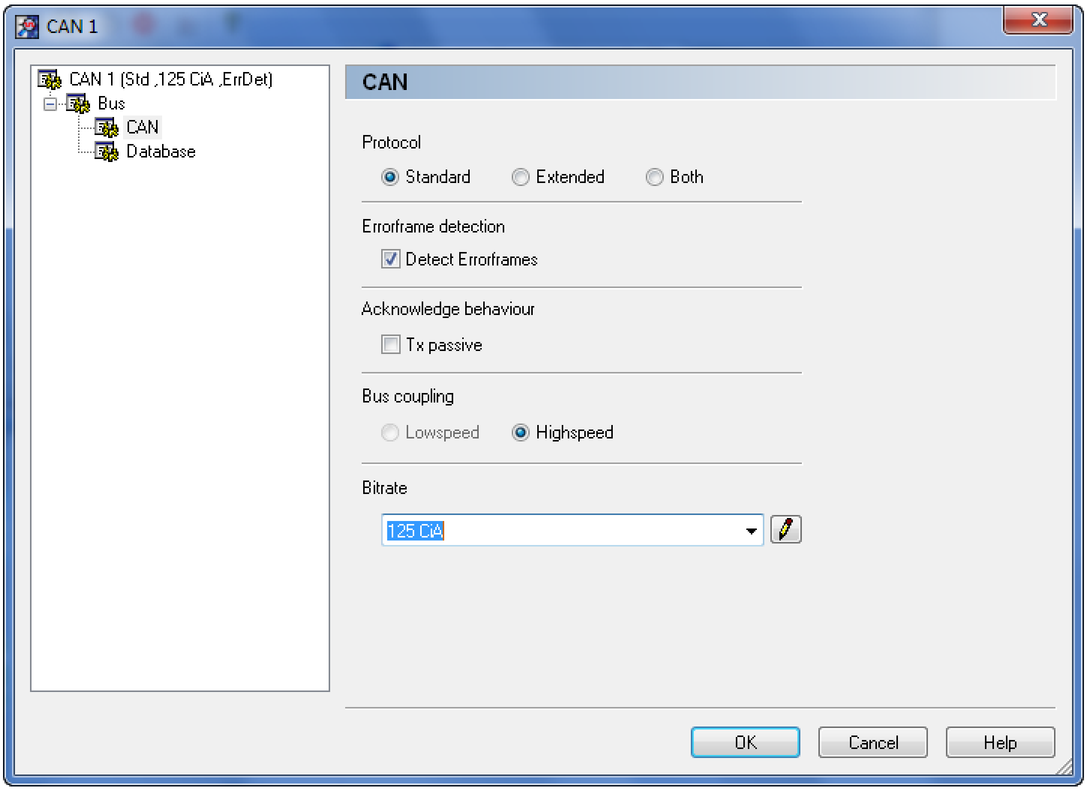
With the Database editor (see database editor manual), it is possible to assign a symbolic name to individual identifiers and to interpret the data transmitted with their identifier.
If no database exists yet, the Database editor can be started directly from the section **Database** of the bus properties dialog (Fig. 5.12). If a database has already been selected, this is automatically loaded when the Database editor is started. 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) 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 section **CAN** of the properties Dialog of a CAN bus. These are:

- Message format
- Error frame detection
- Acknowledge behavior
- Timing parameters
- Buscoupling (High or low speed)

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 and new entries can be added.
The meaning of the parameters in the **CAN** section:



Protocol

● Standard ○ Extended ○ Both

Errorframe detection

☑ Detect Errorframes

Acknowledge behaviour

□ Tx passive

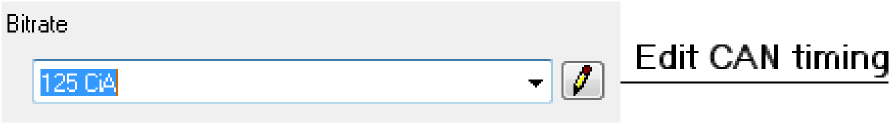
Bus coupling

○ Lowspeed ● Highspeed

Bitrate

125 CiA

Figure 5.13: CAN Settings



Bitrate

125 CiA

Figure 5.14: Open the Timings or Filter dialog

Setting	Operation mode 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 also 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.

Setting bitrate

The bitrate is selected via the symbolic name of the timing. The timing parameters assigned to the name can be altered and new parameter sets can be added. For this, the **Pencil** button next to the symbolic name (Fig. 5.14) is pressed in order to open the timing/time setting dialog (Fig. 5.15).

The meaning of the parameters in the Timings dialog:

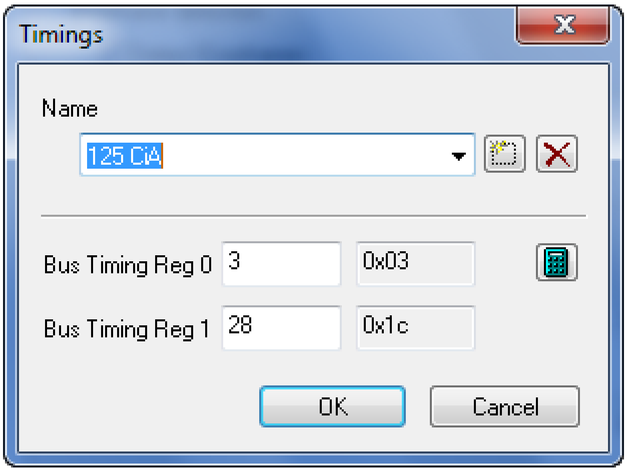


Figure 5.15: CAN Timings dialog

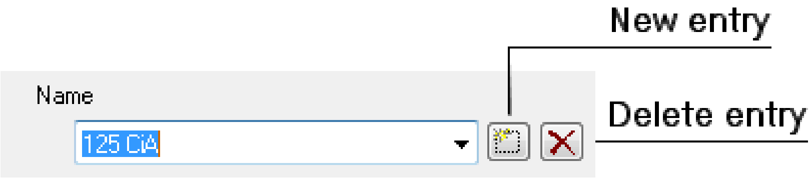


Figure 5.16: Create new entry in the Timings dialog or delete entry

Time settings	Function
Name	Symbolic name of the parameter set
Bus Timing Reg 0	Specific value for bus timing register 0
Bus Timing Reg 1	Specific value for bus timing register 1

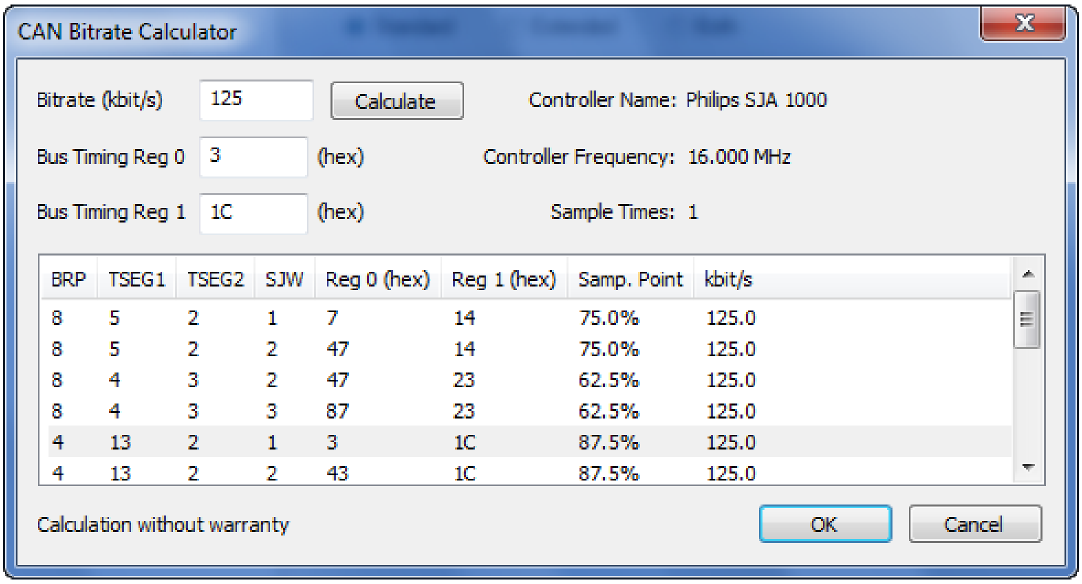
Using the two buttons next to the symbolic name (Fig. 5.16) it is possible to generate new parameter sets or delete the selected parameter set.

Bitrate calculator

The bitrate calculator (Fig. 5.17) can be startet via the **Calculator** button the in the CAN Timings dialog. Here you can calculate the timing parameters by entering the desired bitrate.
Description of the bitrate calculator input fields:

Field	Description
Bitrate (kbit/s)	Bitrate to be calculated in kBit per second
Bus Timing Reg 0	Value of the bus timing register 0
Bus Timing Reg 1	Value of the bus timing register 1

Description of the columns in the list of calculated values:



CAN Bitrate Calculator

Bitrate (kbit/s) 125 Calculate

Controller Name: Philips SJA 1000

Bus Timing Reg 0 3 (hex)

Controller Frequency: 16.000 MHz

Bus Timing Reg 1 1C (hex)

Sample Times: 1

BRP	TSEG1	TSEG2	SJW	Reg 0 (hex)	Reg 1 (hex)	Samp. Point	kbit/s
8	5	2	1	7	14	75.0%	125.0
8	5	2	2	47	14	75.0%	125.0
8	4	3	2	47	23	62.5%	125.0
8	4	3	3	87	23	62.5%	125.0
4	13	2	1	3	1C	87.5%	125.0
4	13	2	2	43	1C	87.5%	125.0

Calculation without warranty

OK Cancel

Figure 5.17: The bitrate calculator

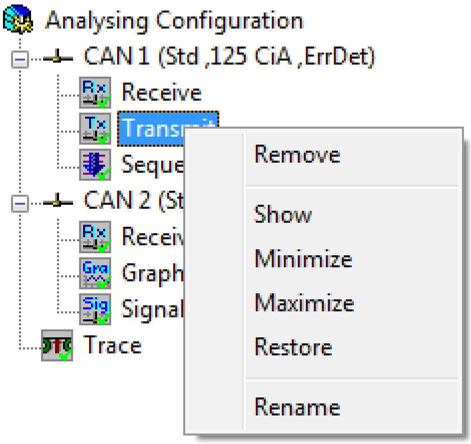
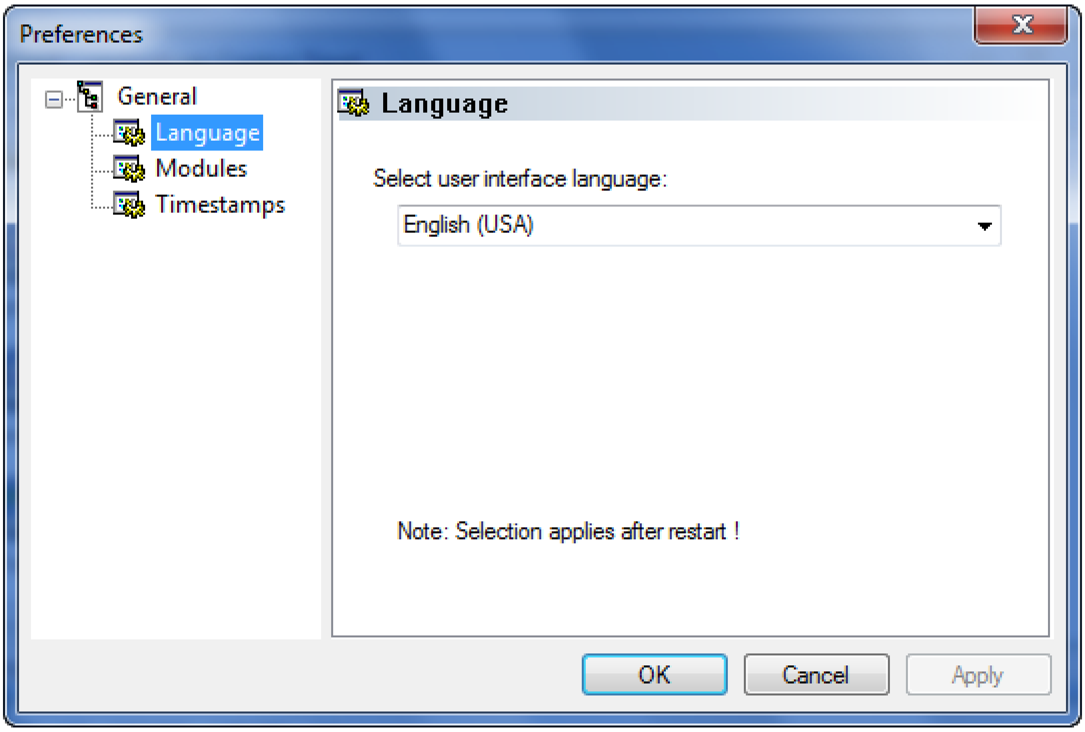


Figure 5.18: Pop-up menu of the analysis modules

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)
Samp. Point	Sample location
kbit/s	Calculated bitrate with the values of the marked line

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. 5.18). 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 **Show** command in the pop-up menu and moves the window of the corresponding module into the foreground.



Select user interface language:

English (USA)

Note: Selection applies after restart !

Figure 5.19: Language Selection dialog

5.1.10 Language Selection

The language setting of the canAnalyser can be altered with the **Language Selection** dialog (Fig. 5.19). 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.

5.1.11 Module settings

It is possible to change module specific settings with the **Module settings** dialog (Fig. 5.20). The two options are:

- visibility of the module in the taskbar
- number of entries in the scroll view of receive modules (default 10000, max. 50000 entries)

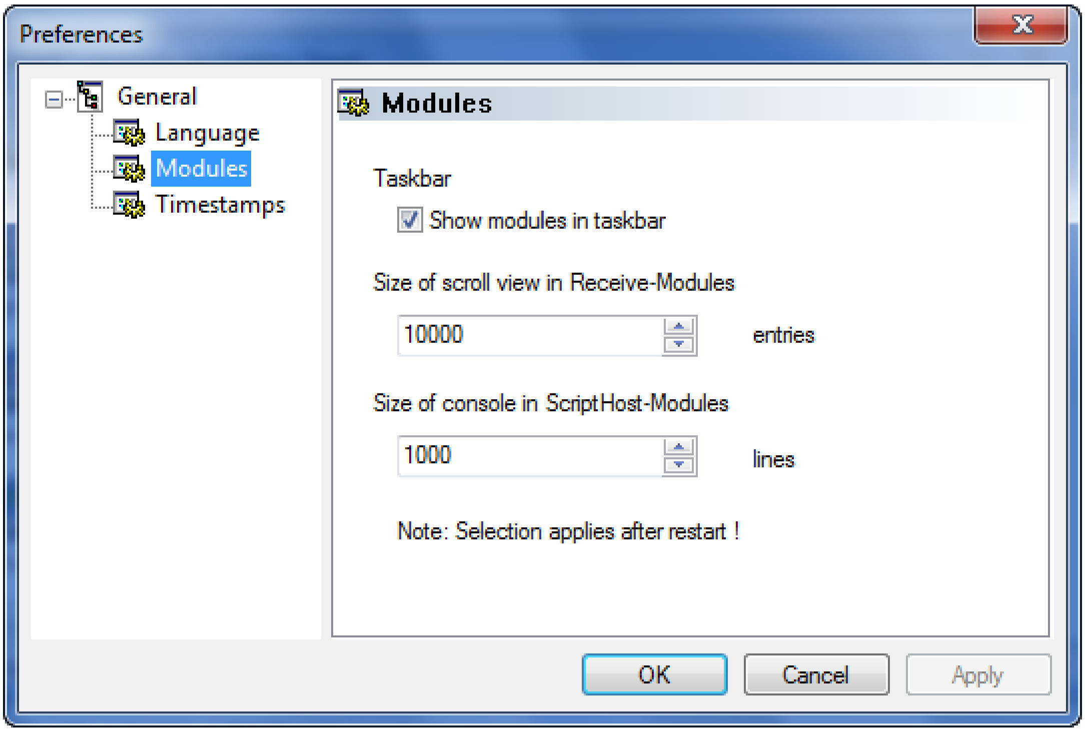
Note that canAnalyser has to be restarted for these options to become effective.

5.1.12 Timestamp settings

You can change the timestamp base time within the **Timestamp settings** dialog (Fig. 5.21). The two options are:

- Reset on Start
- Get Systemtime on Start

With the first option the timestamp is set to zero on controller start and subsequent timestamps are relative this reference time. When using the second option on controller start the system time is taken as reference for the following timestamps.
Note that canAnalyser has to be restarted for these options to become effective.



Modules

Taskbar

☑ Show modules in taskbar

Size of scroll view in Receive-Modules

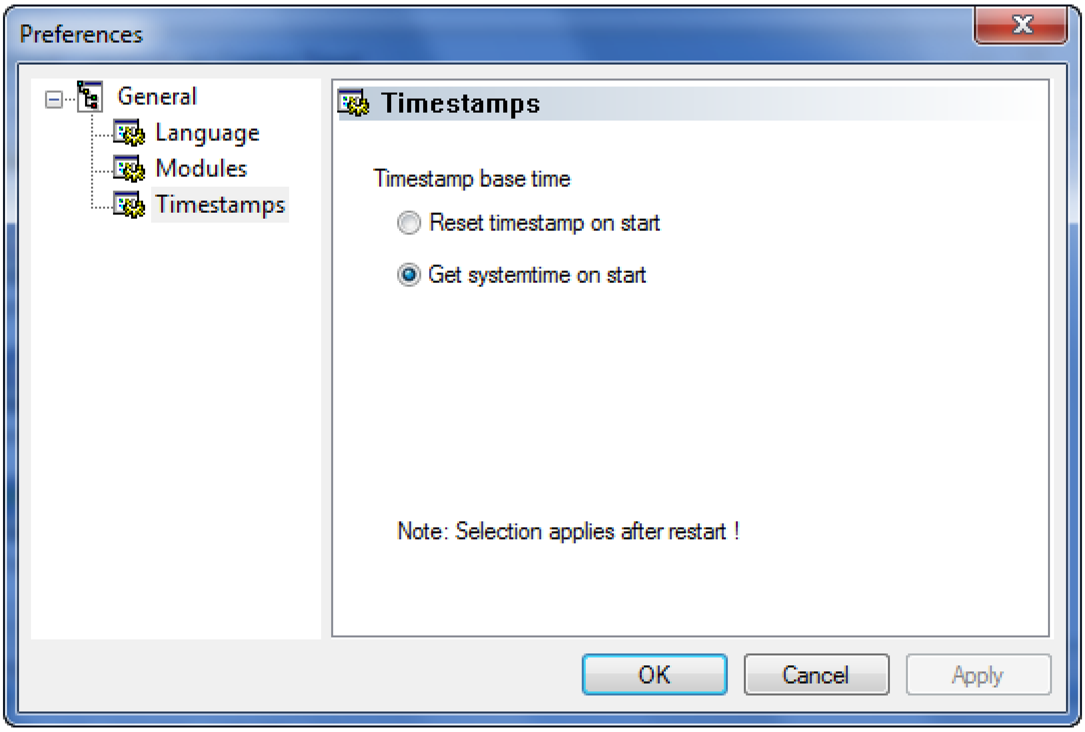
10000 entries

Size of console in ScriptHost-Modules

1000 lines

Note: Selection applies after restart !

Figure 5.20: Module settings dialog



Timestamps

Timestamp base time

○ Reset timestamp on start

● Get systemtime on start

Note: Selection applies after restart !

Figure 5.21: Timestamp settings dialog

5.1.13 Menu reference

File menu

Menu point	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 point	Function
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar
Modules	Shows/hides the module list
Event Log	Shows/hides the error protocol window
Clear Eventlog	Deletes the contents of the error protocol file and of the window

Control menu

Menu point	Function
Start Communication	Starts the online analysis
Stop Communication	Stops the online analysis

Windows menu

Menu point	Function
Show	Opens/activates the module marked in the configuration tree
Remove	Removes the module marked in the configuration tree from the configuration
Minimize	Minimizes the module marked in the configuration tree
Maximimize	Maximizes the module marked in the configuration tree
Restore	Restores the module marked in the configuration tree
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)

Configuration menu

Menu point	Function
Options...	Opens the dialog to select the common canAnalyser settings
Available Filters...	Adjust application wide available message filters see section 5.10.1)

Help menu

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

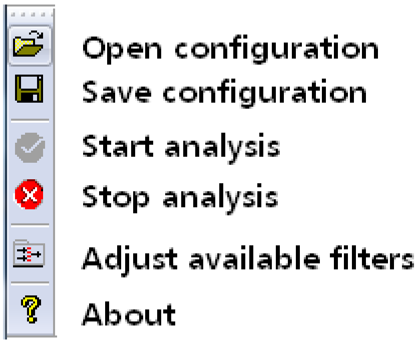


Figure 5.22: Toolbar of the control panel

5.1.14 Toolbar

The most important functions of the control panel can also be called via the toolbar (Fig. 5.22).

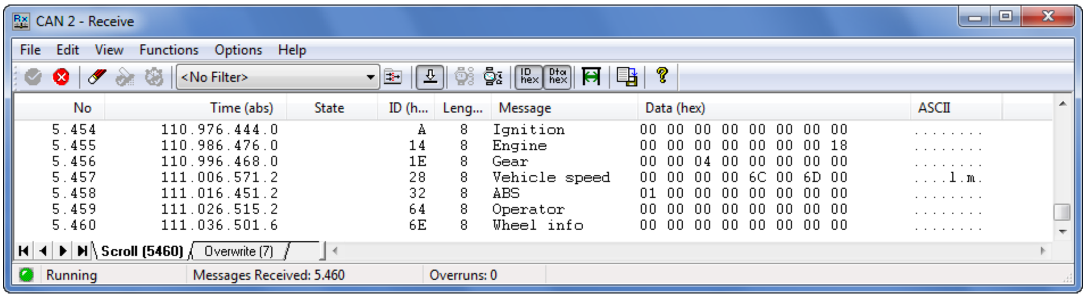
5.2 Receive module

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)
- Display of the received messages sorted according to identifiers (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 are updated simultaneously. It is therefore possible to switch between view modes during reception of messages. 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 (only possible with canAnalyser standard).



No	Time (abs)	State	ID (h...	Leng...	Message	Data (hex)	ASCII
5.454	110.976.444.0		A	8	Ignition	00 00 00 00 00 00 00 00	....
5.455	110.986.476.0		14	8	Engine	00 00 00 00 00 00 00 18	....
5.456	110.996.468.0		1E	8	Gear	00 00 04 00 00 00 00 00	....
5.457	111.006.571.2		28	8	Vehicle speed	00 00 00 00 6C 00 6D 00	....l.m.
5.458	111.016.451.2		32	8	ABS	01 00 00 00 00 00 00 00	....
5.459	111.026.515.2		64	8	Operator	00 00 00 00 00 00 00 00	....
5.460	111.036.501.6		6E	8	Wheel info	00 00 00 00 00 00 00 00	....

Running Messages Received: 5.460 Overruns: 0

Figure 5.23: Receive module in Scroll View

5.2.2 Scroll View

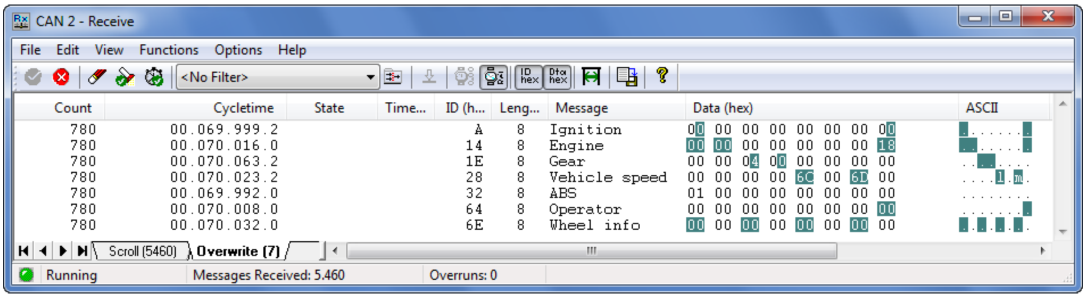
Messages are listed on the **Scroll** index card (Fig. 5.23) in the order of reception with the following information:

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
ID (hex/dec)	Identifier of the received message
Length	Data length code, 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

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:

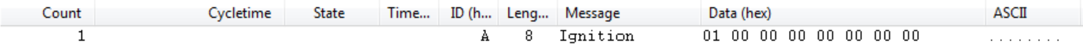
Status	Type	Meaning
C	-	Controller overrun: Messages were lost.
D	-	Driver queue overrun: The PC could not read out the driver queue fast enough. Messages were lost.
Q	-	Software queue overrun: The PC could not read out the internal software queue fast enough. Messages were lost.
S	-	Self-reception: Transmit and receive module use the same controller
E	CAN	Extended CAN frame: If E is not displayed, a standard CAN frame was received.
Y	FlexRay	Sync message
U	FlexRay	Startup message
I	FlexRay	Dynamic message
A	FlexRay	Static message
P	FlexRay	Payload preamble
N	FlexRay	Nullframe
R	FlexRay	Asynchronous mode



Count	Cycletime	State	Time...	ID (h...	Leng...	Message	Data (hex)	ASCII
780	00.069.999.2			A	8	Ignition	00 00 00 00 00 00 00 00	....
780	00.070.016.0			14	8	Engine	00 00 00 00 00 00 00 18	....
780	00.070.063.2			1E	8	Gear	00 00 04 00 00 00 00 00	....
780	00.070.023.2			28	8	Vehicle speed	00 00 00 00 6C 00 6D 00	....l.m.
780	00.069.992.0			32	8	ABS	01 00 00 00 00 00 00 00	....
780	00.070.008.0			64	8	Operator	00 00 00 00 00 00 00 00	....
780	00.070.032.0			6E	8	Wheel info	00 00 00 00 00 00 00 00	....

Running | Messages Received: 5.460 | Overruns: 0

Figure 5.24: Receive module in Overwrite View



Count	Cycletime	State	Time...	ID (h...	Leng...	Message	Data (hex)	ASCII
1				A	8	Ignition	01 00 00 00 00 00 00 00	....

Figure 5.25: First reception of an identifier in the Overwrite View

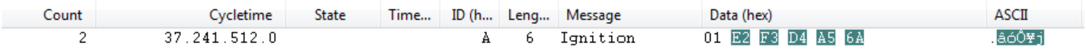
5.2.3 Overwrite View

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. 5.24), the messages are sorted according to identifiers. The information of the last message received is always shown in each case. When change monitoring is switched on, changed data are highlighted in color. If a cycle time is defined in the database, timeouts of the database are displayed by the icon ⏱.
The index card **Overwrite** 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
State	Display of the receive status (such as Scroll View)
Timeout	If a cycle time is defined in the database, timeouts of the database are displayed by a ⏱ icon
ID (hex/dec)	Identifier of the received message
Length	Data length code, 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

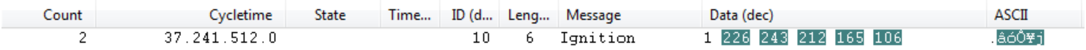
5.2.4 Data change detection

The data change detection occurs in the Overwrite View of the Receive module and can be enabled and disabled via the menu point **Options**.
With the first reception of an identifier, a new message is added in the Overwrite View (Fig. 5.25). 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.



Count	Cycletime	State	Time...	ID (h...	Leng...	Message	Data (hex)	ASCII
2	37.241.512.0			A	6	Ignition	01 E2 F3 D4 A5 6A	.âóÔ¥j

Figure 5.26: Hexadecimal data change display



Count	Cycletime	State	Time...	ID (d...	Leng...	Message	Data (dec)	ASCII
2	37.241.512.0			10	6	Ignition	1 226 243 212 165 106	.âóÔ¥j

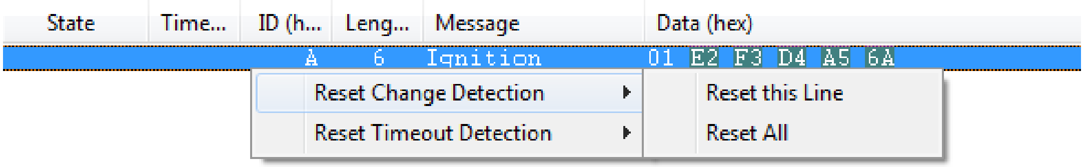
Figure 5.27: Decimal data change display

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. 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. 5.26).
With decimal display of the data field, changes to the received data bytes can only be highlighted as a whole. (Fig. 5.27).
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. 5.28).
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



State	Time...	ID (h...	Leng...	Message	Data (hex)
		A	6	Ignition	01 E2 F3 D4 A5 6A

Figure 5.28: Context menu for monitoring display

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 point **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 point **Functions | Select Filter**.

5.2.7 Menu reference

File menu

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

Edit menu

Menu point	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 point	Function
ID hex	Display of the message identifiers in hexadecimal or decimal notation
Data hex	Display of the data of layer-2 messages in hexadecimal or decimal notation
Time rel.	Display of the time stamp absolute or relative to the previously displayed telegram
Show recent Frames	Always the most recent telegrams are displayed in Scroll View
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar
Autosize Columns	Regulates ideal column widths

Functions menu

Menu point	Function
Start	Starts message reception of the Receive module
Stop	Stops message reception
Available Filters...	Adjust application wide available message filters see section 5.10.1)
Select Filter	Selects a message filter
Clear All	Deletes the display and resets the receive counter
Reset Change Detection	Resets the data change detection
Reset Timeout Detection	Resets cycle time detection

Options menu

Menu point	Function
Data Change Detection	Switches the data change display on or off
Change Detection Color	Opens a dialog to select the color with which changed data are highlighted
Timeout Detection	Switches cycle time monitoring on or off
Fonts...	Opens a dialog to select the font type in which the data are displayed in all Views
Data with Line Break	Displays the data over several lines if the specified column width is not sufficient for the display of the complete data field.
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).

Help menu

Menu point	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. 5.29).

5.2.9 Status bar

The status bar contains an LED icon which displays the status of the control panel or of the Receive module:

LED color	Meaning
Green	Control panel and Receive module are started
Red flashing	Control panel is stopped
Red	Receive module is stopped

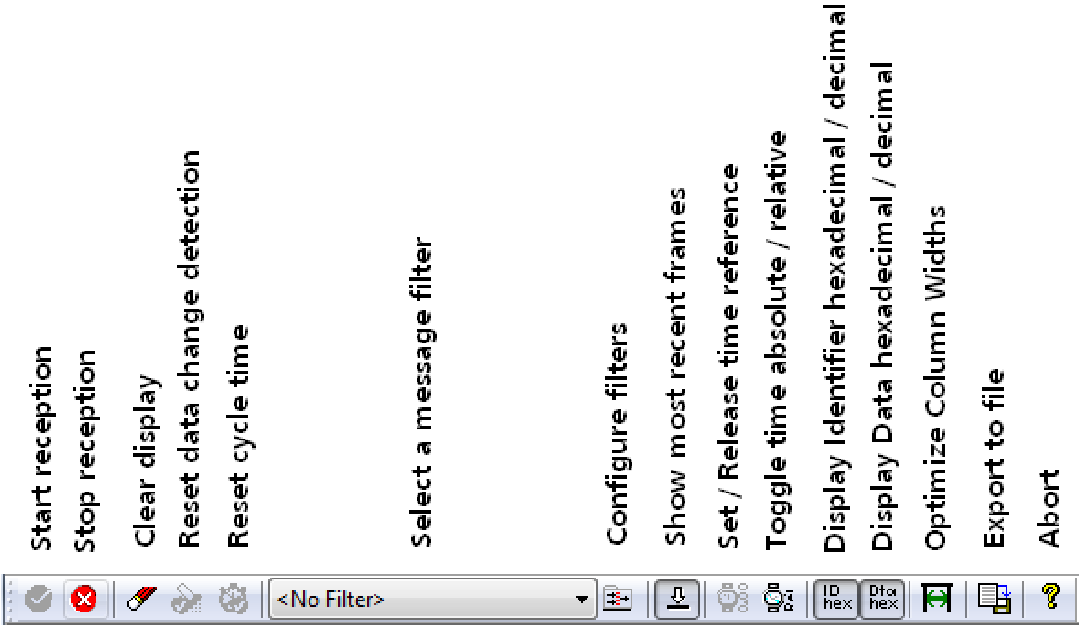


Figure 5.29: Toolbar of the Receive module

5.2.10 Hotkeys

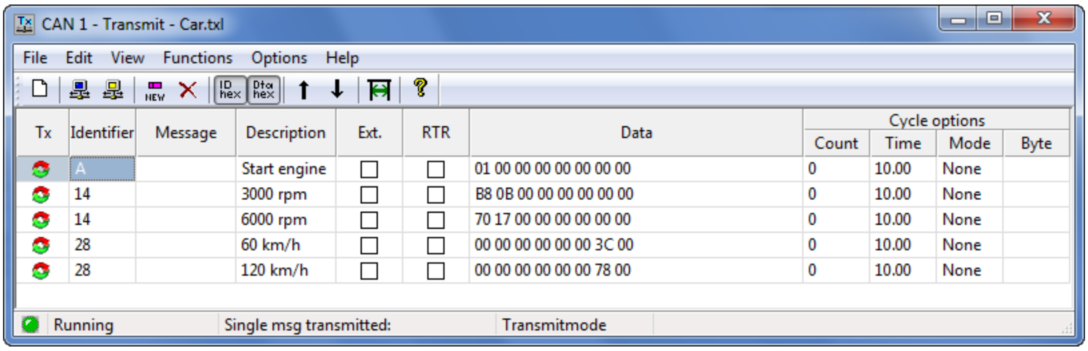
Ctrl+TAB	Switch between Scroll View and Overwrite View
F1	Online-Help
F2	Go to Next Marker in Scroll View
Shift+F2	Go to Previous Marker in Scroll View
Ctrl+F2	Toggle Marker in Scroll View
F5	Start message reception
Shift+F5	Stop message reception
F8	Clear all Views
Ctrl+C	Copy marked lines CSV formatted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



CAN 1 - Transmit - Car.txl

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	10.00	None	
⟳	14		3000 rpm	□	□	B8 0B 00 00 00 00 00 00	0	10.00	None	
⟳	14		6000 rpm	□	□	70 17 00 00 00 00 00 00	0	10.00	None	
⟳	28		60 km/h	□	□	00 00 00 00 00 00 3C 00	0	10.00	None	
⟳	28		120 km/h	□	□	00 00 00 00 00 00 78 00	0	10.00	None	

● Running | Single msg transmitted: | Transmitmode

Figure 5.30: Transmit module

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

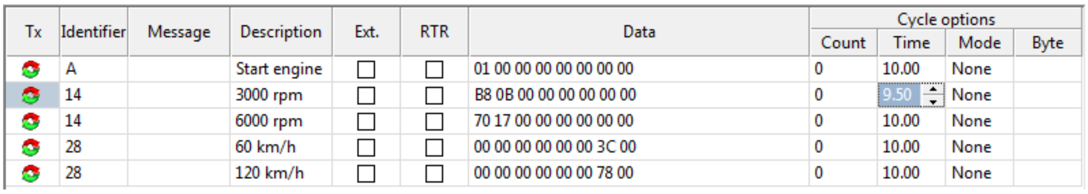
5.3.2 Program window

In the Transmit module (Fig. 5.30), the objects to be transmitted are entered in a table, which is displayed centrally in the Transmit module. The entries are transmitted by selecting the message and command **Transmit Single Message** or **Transmit Cyclic Message**.
The transmit table has the following columns:

Column	Meaning
Tx	Symbol ⟳ for transmission state visualization. It's rotating as long as the message's cyclic transmission is active.
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 used 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
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 Creation of transmit objects

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 message**. A new entry



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	10.00	None	
⟳	14		3000 rpm	□	□	B8 0B 00 00 00 00 00 00	0	9.50	None	
⟳	14		6000 rpm	□	□	70 17 00 00 00 00 00 00	0	10.00	None	
⟳	28		60 km/h	□	□	00 00 00 00 00 00 3C 00	0	10.00	None	
⟳	28		120 km/h	□	□	00 00 00 00 00 00 78 00	0	10.00	None	

Figure 5.31: Edit mode of the cells

is also generated when the cursor key is pressed in the last line of the transmit table ↓. The new line is then marked with the selection bar. The individual columns can be selected with the mouse or with the cursor keys ← and →. Editable fields are identified by a different color in the selection bar. A transmit object is defined by entering the identifier, the description and the data to be transmitted. Here the edit fields change automatically to edit mode as soon as a numerical or alphanumerical key or the F2 key is pressed (Fig. 5.31).

The name of a transmit object is taken from the project database specified in the control panel and automatically added.

Editing of the edit fields or columns of the transmit table is completed by pressing the enter key, the cursor key ← or → or by clicking on another cell of the transmit table.

5.3.4 Editing the transmit table

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

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

5.3.5 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 ↓

Once a message is selected, it is transmitted by:

- Pressing the key F5
- Activating the menu point **Functions | Transmit Single Message**
- Activating the **Transmit single message** button in the toolbar
- Clicking with the left mouse button on the transmit symbol ⟳ in the first column

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

5.3.6 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). Before transmission, the cyclic message must be selected.
Cyclic transmission is carried out by:

- The key combination F6
- Activating the menu entry **Functions | Transmit Cyclic Message**
- Activating the **Transmit cyclic message** button in the toolbar
- Pressing the Ctrl-key and at the same time clicking with the left mouse button on the transmit symbol in the first column.

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

5.3.7 Drag-and-Drop

Received CAN messages might be dragged from the Scroll View of a 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.

5.3.8 Menu reference

File menu

Menu point	Function
New	Creates a new, empty transmit table
Import Transmit Data...	Load transmit table from a file (see section 1.4)
Export Transmit Data...	Save transmit table to a file (see section 1.4)
Recent files	Displays the last transmit tables opened
Exit	Exits the Transmit module

Edit menu

Menu point	Function
Insert 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 point	Function
ID Hex	Displays the identifier column in hexadecimal or decimal notation
Data Hex	Displays the data column in hexadecimal or decimal notation
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar
Columns	Shows/hides the columns: Message, Description, Cycle options
Autosize Columns	Regulates ideal column widths

Functions menu

Menu point	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

Options menu

Menu point	Function
Font...	Opens a dialog to select the font used to display data
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).

Help menu

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

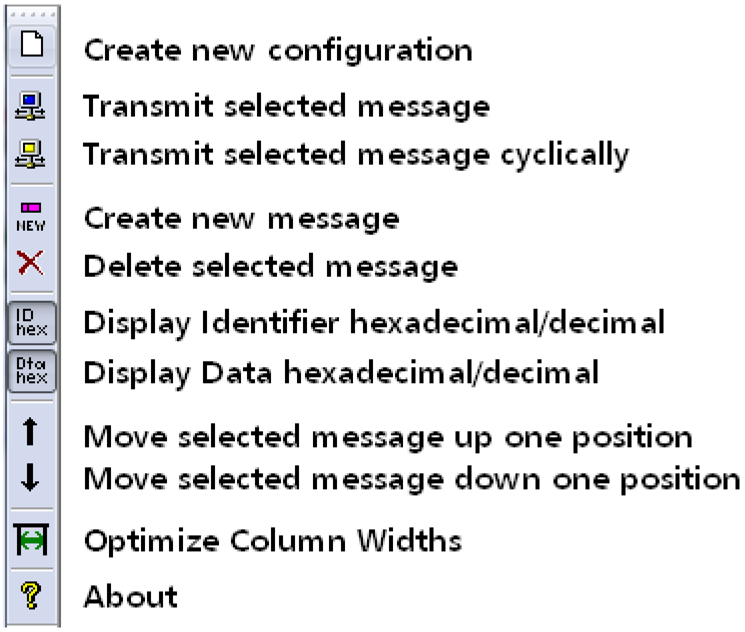
5.3.9 Toolbar

The most important functions of the Transmit module can also be called via the toolbar (Fig. 5.32).

5.3.10 Status bar

The status bar of the Transmit module contains an LED symbol which displays the status of the control panel or of the Transmit module:

LED color	Meaning
Green	Control panel and Transmit module are started
Red flashing	Control panel is stopped.



Create new configuration

Transmit selected message

Transmit selected message cyclically

Create new message

Delete selected message

Display Identifier hexadecimal/decimal

Display Data hexadecimal/decimal

Move selected message up one position

Move selected message down one position

Optimize Column Widths

About

Figure 5.32: Toolbar of the Transmit module

5.4 Trace module

5.4.1 Overview

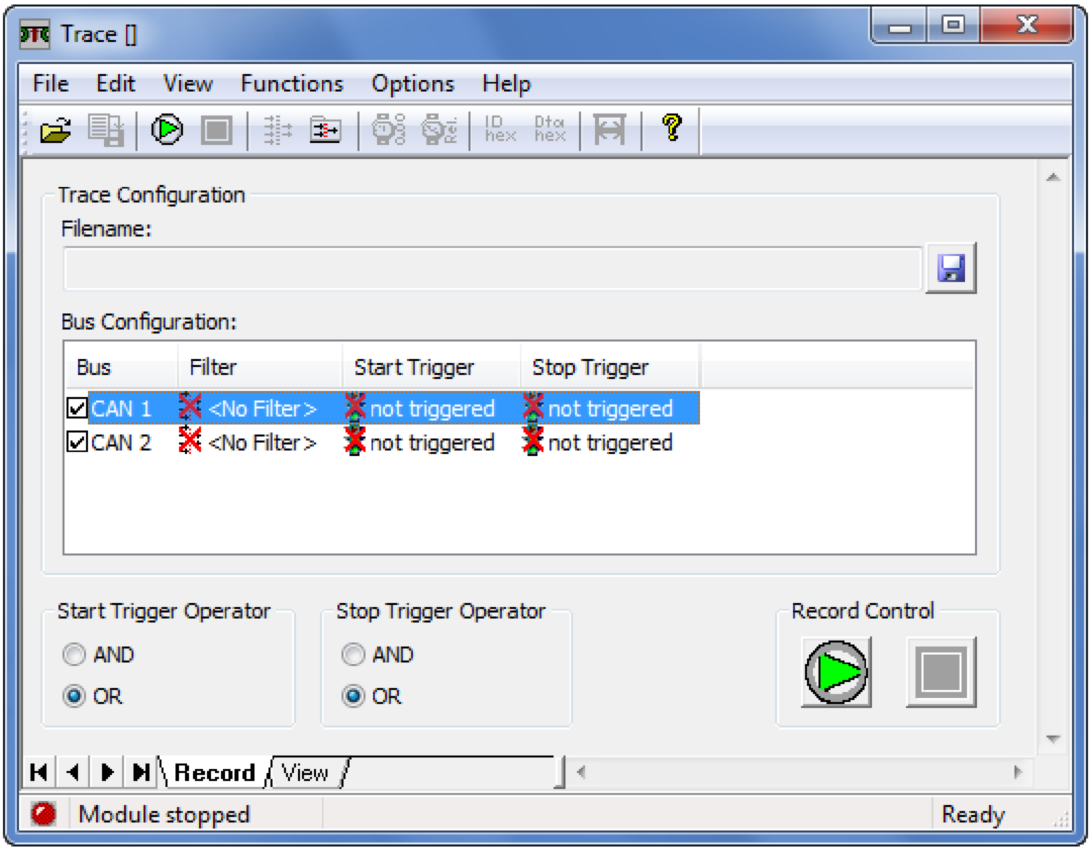
The Trace module is used for the simultaneous recording of bus traffic of several buses. It is therefore possible to analyse the message traffic encompassing all buses and after expiry of a complete communication cycle.
Using various triggers, the trace recording can be automatically started and stopped 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
  - 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 shows two index cards:

Register	Use
Record	Setting of parameters for trace recording (selection of the buses/controllers to be recorded, configuration of the filters and trigger events)
View	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 index card.



Trace Configuration

Filename:

Bus Configuration:

Bus	Filter	Start Trigger	Stop Trigger
☑ CAN 1	<No Filter>	not triggered	not triggered
☑ CAN 2	<No Filter>	not triggered	not triggered

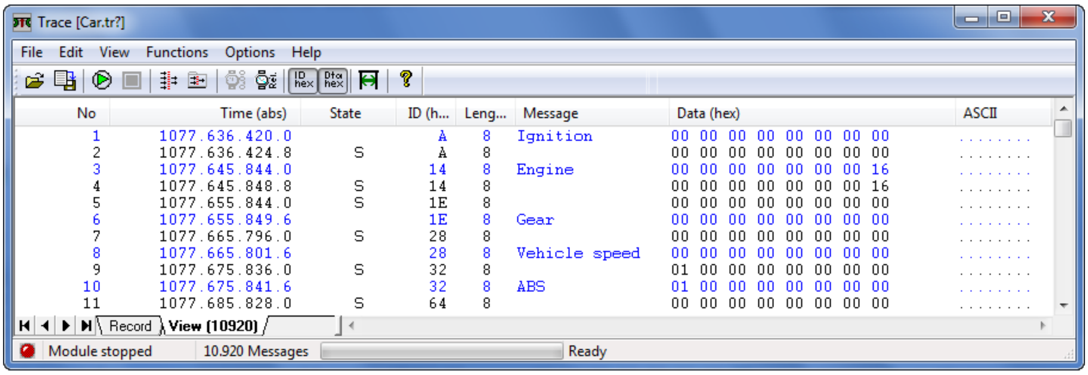
Start Trigger Operator
○ AND
● OR

Stop Trigger Operator
○ AND
● OR

Record Control

Module stopped   Ready

Figure 5.33: Trace module, index card trace

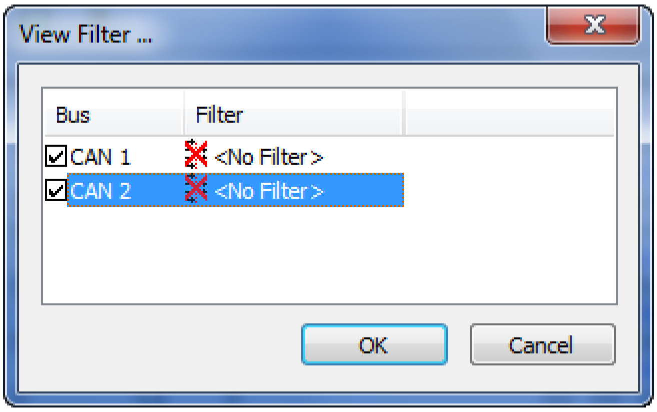


No	Time (abs)	State	ID (h...	Leng...	Message	Data (hex)	ASCII
1	1077.636.420.0		A	8	Ignition	00 00 00 00 00 00 00 00	....
2	1077.636.424.8	S	A	8		00 00 00 00 00 00 00 00	....
3	1077.645.844.0		14	8	Engine	00 00 00 00 00 00 00 16	....
4	1077.645.848.8	S	14	8		00 00 00 00 00 00 00 16	....
5	1077.655.844.0	S	1E	8		00 00 00 00 00 00 00 00	....
6	1077.655.849.6		1E	8	Gear	00 00 00 00 00 00 00 00	....
7	1077.665.796.0	S	28	8		00 00 00 00 00 00 00 00	....
8	1077.665.801.6		28	8	Vehicle speed	00 00 00 00 00 00 00 00	....
9	1077.675.836.0	S	32	8		01 00 00 00 00 00 00 00	....
10	1077.675.841.6		32	8	ABS	01 00 00 00 00 00 00 00	....
11	1077.685.828.0	S	64	8		00 00 00 00 00 00 00 00	....

Module stopped   10.920 Messages   Ready

Figure 5.34: Trace module, index card Record

In the index card **Record** (Fig. 5.33) 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 **Filename**. By clicking with the mouse on the **Open** 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 index card **View** (Fig. 5.34). The recorded telegrams are displayed in order of the time they were received and marked in color according to the bus from which they originate. The color of the marking is assigned in the menu **Options | View options**.
The number of visible telegrams can be reduced via the menu command **Functions | View filter...** in the filter dialog by filtering messages or hiding a complete bus/controller.



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

Figure 5.35: Display filter

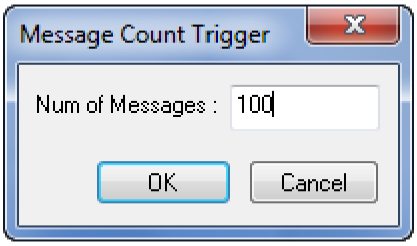
5.4.3 Trace recording

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 recorded messages are 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 5.10.1 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 index card **View**.
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).
The start and stop triggers of the buses/controllers to be recorded are linked to each other via **AND**- or **OR**-operators.

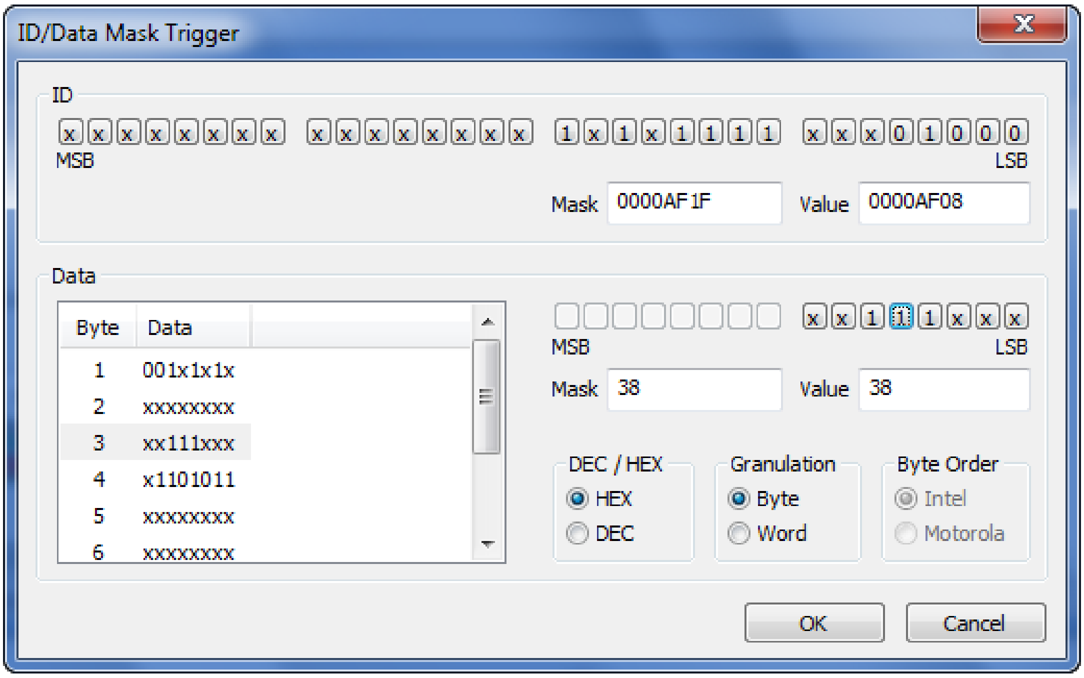
Operator	Function
OR	The trigger event occurs as soon as the trigger condition of one of the start or stop triggers is fulfilled
AND	The trigger event occurs when the trigger conditions of all start or stop triggers are fulfilled



Message Count Trigger

Num of Messages : 100

Figure 5.36: Message count trigger



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

□ □ □ □ □ □ □ □   x x 1 1 1 x x x

MSB LSB

Mask 38 Value 38

DEC / HEX: ● HEX ○ DEC

Granulation: ● Byte ○ Word

Byte Order: ● Intel ○ Motorola

Figure 5.37: ID/Data mask trigger

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.

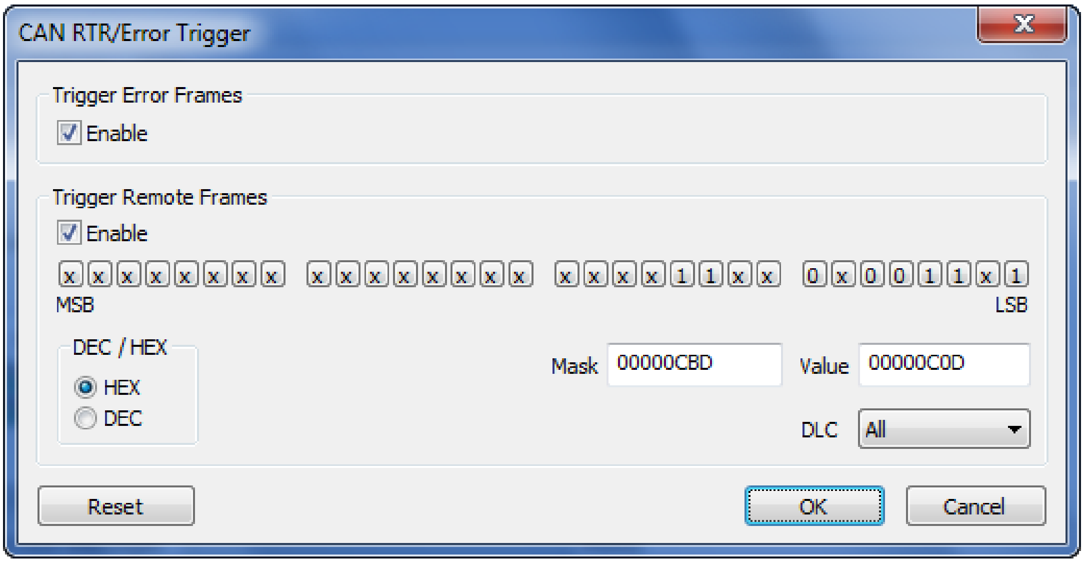
Trigger	Function
Message Count Trigger	Triggering on a configurable number of received messages
ID/Data Mask Trigger	Triggering on identifiers and/or databyte values, which are specified via bit-masks
CAN RTR/Error Trigger	Triggering on CAN remote and/or error frames

Message count trigger

The trigger condition of the message count trigger 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, 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.



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 5.38: CAN RTR/error trigger

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.
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, 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 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 Further processing of a trace recording

A saved trace recording can be imported into the Trace module for another analysis via the menu point **File | Open Trace...**.
In the Signal module, a trace recording can be interpreted with the database specified in the control panel. The results of the interpretation are written in a file to be specified.

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.

5.4.6 Menu reference

File menu

Menu point	Function
Open Trace...	Opens a previously saved trace recording
Export Messages...	Exports the displayed trace recording to a text file
Exit	Exits the Trace module

Edit menu

Menu point	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 point	Function
ID hex	Displays the message identifiers on the index card View in hexadecimal or decimal notation
Data hex	Displays the message data on the index card View in hexadecimal or decimal notation
Time rel.	Displays the time on the index card View absolute or relative to the previously displayed message
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the status bar
Autosize Columns	Regulates ideal column widths

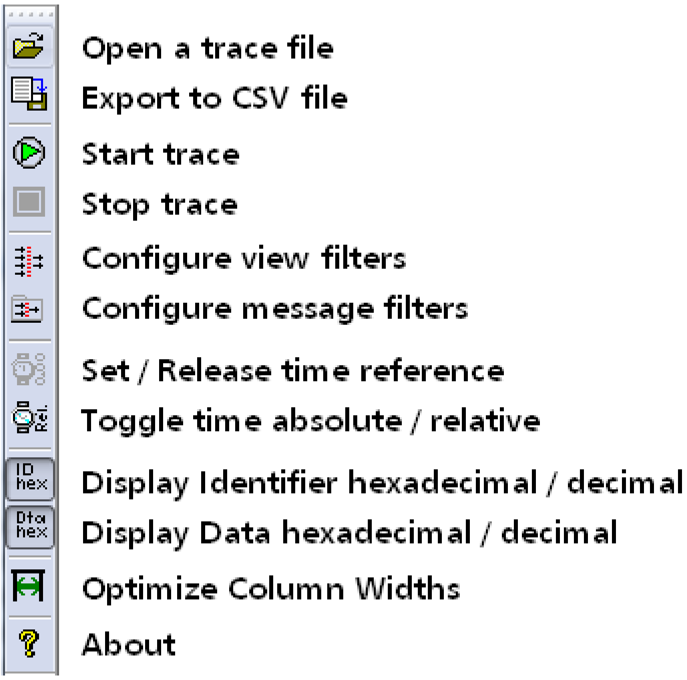

Open a trace file
Export to CSV file
Start trace
Stop trace
Configure view filters
Configure message filters
Set / Release time reference
Toggle time absolute / relative
Display Identifier hexadecimal / decimal
Display Data hexadecimal / decimal
Optimize Column Widths
About

Figure 5.39: Toolbar of the Trace module

Functions menu

Menu point	Function
Start	Begins a trace recording
Stop	Ends a trace recording
Available Filters...	Adjust application wide available message filters see section 5.10.1)

Options menu

Menu point	Function
Fonts...	Opens a dialog to select the font type in which the messages are displayed on the index card View
View Options...	Opens a dialog to select the colors in which the messages of the various buses are displayed on the View page View
Data with Line Break	Displays the data field over several lines if the specified column width is not sufficient to display the whole data field
Data Granulation	Displays the data bytewise or combines two databytes in each case to one WORD; with WORD display it is possible to choose between Little Endian (Intel) or Big Endian format (Motorola)

Help menu

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

5.4.7 Toolbar

The most important functions of the Trace module can also be called via the toolbar (Fig. 5.39).

5.4.8 Status bar

The status bar of the Trace module contains an LED symbol that displays the status of the control panel or of the Trace module:

LED color	Meaning
Green	Control panel and message recording in Trace module are started
Red flashing	Control panel is stopped
Red	Message recording in Trace module is stopped

5.4.9 Hotkeys

Ctrl+TAB	Switch between Record View and Trace View
F1	Online-Help
F2	Go to Next Marker in Trace View
Shift+F2	Go to Previous Marker in Trace View
Ctrl+F2	Toggle Marker in Trace View
F3	Search next
F5	Start message reception
Shift+F5	Stop message reception
F8	Clear all Views
Ctrl+C	Copy marked lines CSV formatted to clipboard
Ctrl+E	Export the received messages to a file
Ctrl+F	Open the search dialog
Ctrl+I	Configure Filter
Ctrl+O	Open an existing trace recording
PageDown	Scroll one page ahead in current Trace View
PageUp	Scroll one page backward in Trace View
Ctrl+PageDown	Scroll 1000 messages ahead in Trace View
Ctrl+PageUp	Scroll 1000 messages backward in Trace View
Ctrl+0	Jump to Time Reference message in Trace View
Ctrl+1..9	Jump to 10%..90% of Trace View

5.5 Replay module

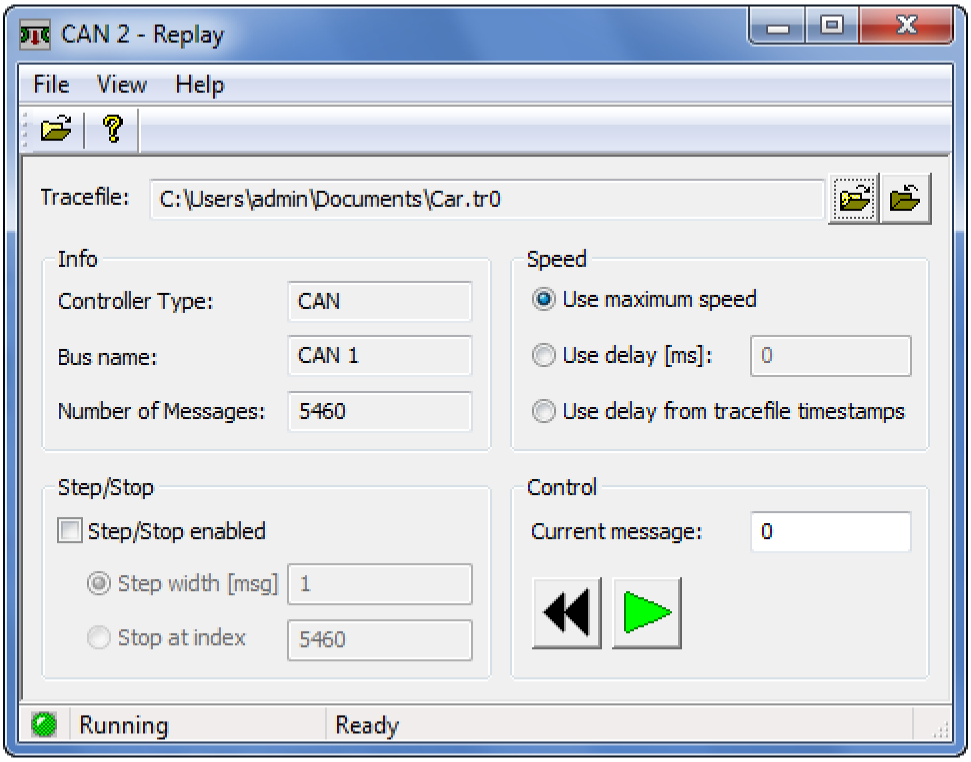
5.5.1 Overview

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.
- 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. 5.40) which influences the replay.



Tracefile: C:\Users\admin\Documents\Car.tr0

Info
Controller Type: CAN
Bus name: CAN 1
Number of Messages: 5460

Speed
● Use maximum speed
○ Use delay [ms]: 0
○ Use delay from tracefile timestamps

Step/Stop
□ Step/Stop enabled
● Step width [msg] 1
○ Stop at index 5460

Control
Current message: 0

Running Ready

Figure 5.40: Replay module

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** option you are able to do a stepwise replay with adjustable step size.
The **Stop** option may be used to stop the replay at a given message index.
In then **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 right are for rewind or start replay.

5.5.3 Menu reference

File menu

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

View menu

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

Help menu

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

5.5.4 Status bar

The status bar contains an LED icon which displays the status of the control panel:

LED color	Meaning
Green	Control Panel is started
Red flashing	Control Panel is stopped

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.

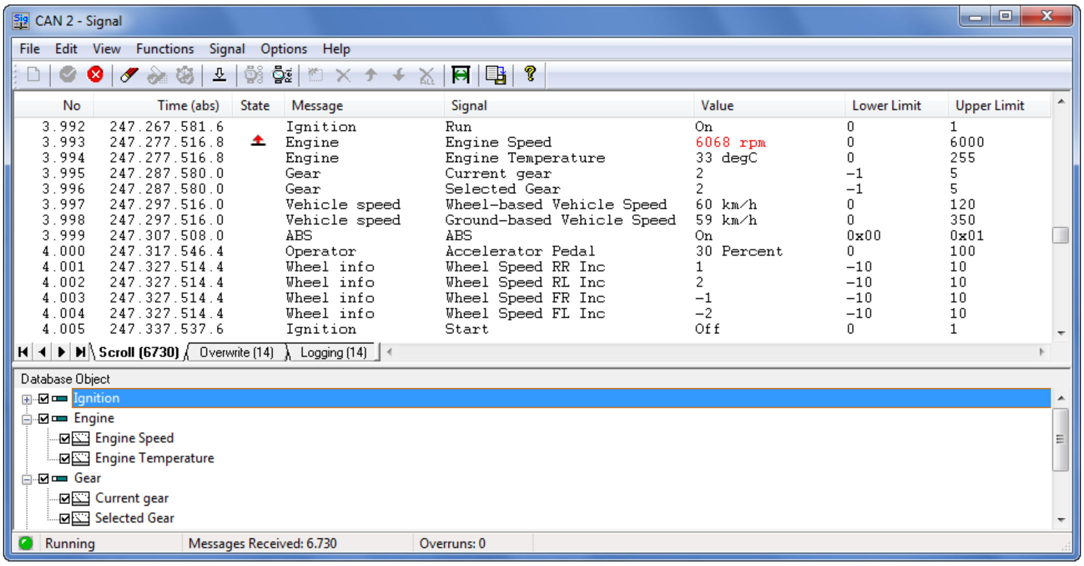
With the Signal module (Fig. 5.41) it is possible to display received layer-2 message contents in interpreted form. This analysis module is therefore particularly suitable for the installation, testing and maintenance of fieldbus systems, as it enables easy handling of physical and logical parameters.

The 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.

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

The interpreted signals can be monitored in terms of changes in their values and observance of the cycle time defined in the database.



No	Time (abs)	State	Message	Signal	Value	Lower Limit	Upper Limit
3.992	247.267.581.6		Ignition	Run	On	0	1
3.993	247.277.516.8	⬆	Engine	Engine Speed	6068 rpm	0	6000
3.994	247.277.516.8		Engine	Engine Temperature	33 degC	0	255
3.995	247.287.580.0		Gear	Current gear	2	-1	5
3.996	247.287.580.0		Gear	Selected Gear	2	-1	5
3.997	247.297.516.0		Vehicle speed	Wheel-based Vehicle Speed	60 km/h	0	120
3.998	247.297.516.0		Vehicle speed	Ground-based Vehicle Speed	59 km/h	0	350
3.999	247.307.508.0		ABS	ABS	On	0x00	0x01
4.000	247.317.546.4		Operator	Accelerator Pedal	30 Percent	0	100
4.001	247.327.514.4		Wheel info	Wheel Speed RR Inc	1	-10	10
4.002	247.327.514.4		Wheel info	Wheel Speed RL Inc	2	-10	10
4.003	247.327.514.4		Wheel info	Wheel Speed FR Inc	-1	-10	10
4.004	247.327.514.4		Wheel info	Wheel Speed FL Inc	-2	-10	10
4.005	247.337.537.6		Ignition	Start	Off	0	1

Database Object

- ☑ Ignition
- ☑ Engine
  - ☑ Engine Speed
  - ☑ Engine Temperature
- ☑ Gear
  - ☑ Current gear
  - ☑ Selected Gear

Running | Messages Received: 6.730 | Overruns: 0

Figure 5.41: Signal module main window

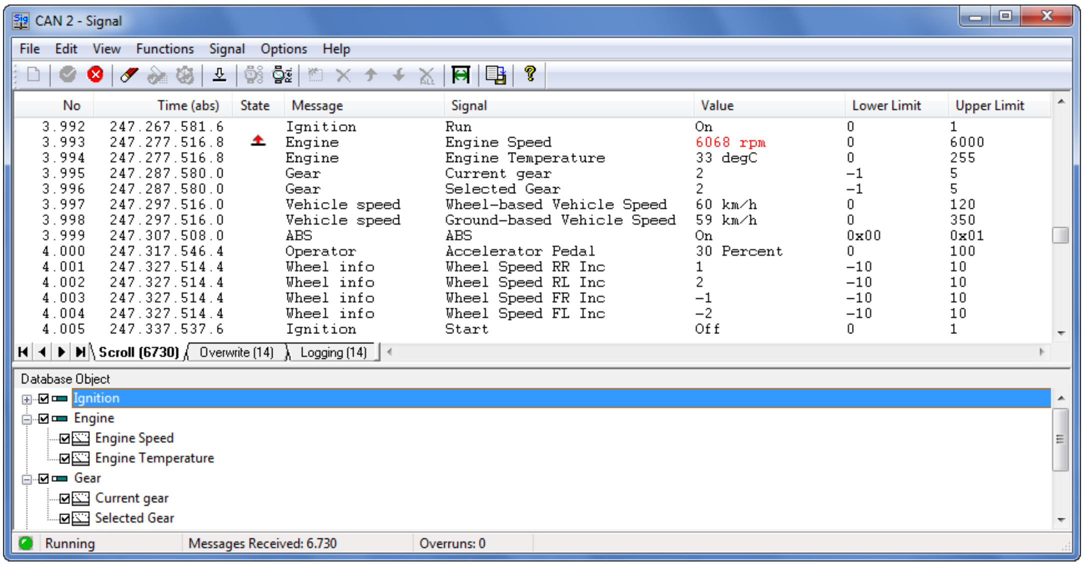
To be able to use the signal module, the project database to be used for the associated bus system must be specified in the control panel. For the description of the definition of variables and the creation of a project database, please refer to the manual of the Database editor.
In the Signal module, the received data can be displayed online in two different modes:

- Scroll View
- Overwrite View

Both displays are updated simultaneously. The Signal module can be configured in such a way that it only receives certain signals. Specific analysis of these messages is therefore very easy. 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 (only possible with canAnalyser standard).
The main window of the Signal module is divided into two halves: In the top half, the received messages are being displayed. The bottom half shows the structure of the project database used for the interpretation.
Important: The messages 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. Messages which are not interpreted are not listed by the Signal module in the Scroll View or are marked gray in the overwrite display.

5.6.2 Scroll View

The **Scroll** index page (Fig. 5.42) displays the signals in the order in which they were received. For signals with the same time stamp, only the time stamp for the first signal is displayed. The relation can therefore be seen at first glance.
The individual columns of the receive display in **Scroll** View have the following meaning:



No	Time (abs)	State	Message	Signal	Value	Lower Limit	Upper Limit
3.992	247.267.581.6		Ignition	Run	On	0	1
3.993	247.277.516.8	▲	Engine	Engine Speed	6068 rpm	0	6000
3.994	247.277.516.8		Engine	Engine Temperature	33 degC	0	255
3.995	247.287.580.0		Gear	Current gear	2	-1	5
3.996	247.287.580.0		Gear	Selected Gear	2	-1	5
3.997	247.297.516.0		Vehicle speed	Wheel-based Vehicle Speed	60 km/h	0	120
3.998	247.297.516.0		Vehicle speed	Ground-based Vehicle Speed	59 km/h	0	350
3.999	247.307.508.0		ABS	ABS	On	0x00	0x01
4.000	247.317.546.4		Operator	Accelerator Pedal	30 Percent	0	100
4.001	247.327.514.4		Wheel info	Wheel Speed RR Inc	1	-10	10
4.002	247.327.514.4		Wheel info	Wheel Speed RL Inc	2	-10	10
4.003	247.327.514.4		Wheel info	Wheel Speed FR Inc	-1	-10	10
4.004	247.327.514.4		Wheel info	Wheel Speed FL Inc	-2	-10	10
4.005	247.337.537.6		Ignition	Start	Off	0	1

Scroll (6730) | Overwrite (14) | Logging (14)

Database Object

- ☑ Ignition
- ☑ Engine
  - ☑ Engine Speed
  - ☑ Engine Temperature
- ☑ Gear
  - ☑ Current gear
  - ☑ Selected Gear

Running | Messages Received: 6.730 | Overruns: 0

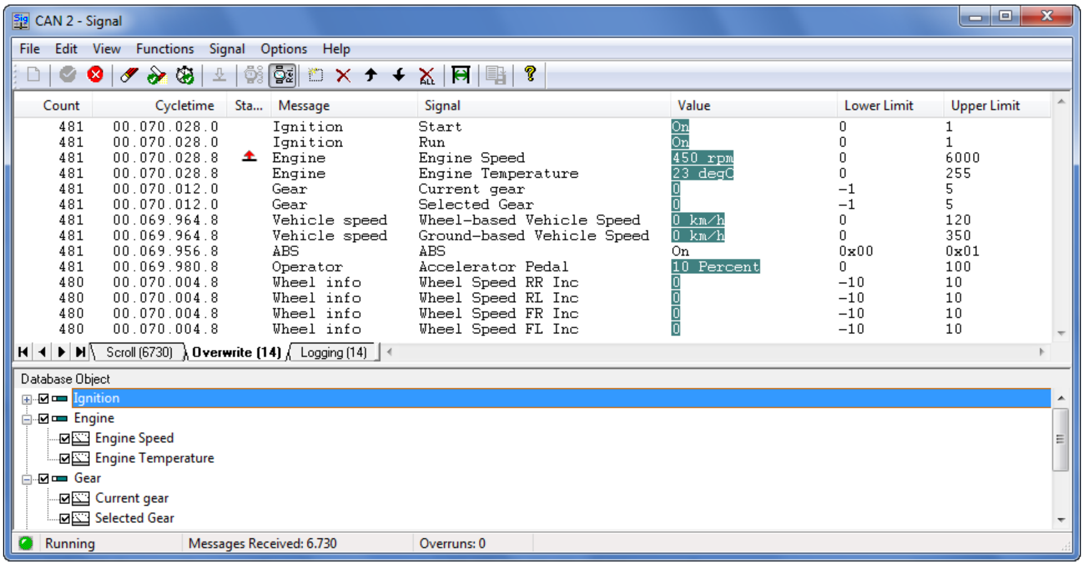
Figure 5.42: Signal module Scroll View

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 ▲
Message	Name of the message in the database used; is usually the alias of the identifier
Signal	Signal name in the database
Value	Value of the signal
Lower Limit	Signal value's lower limit
Upper Limit	Signal value's upper limit

5.6.3 Overwrite View

The most recent values of selected signals are displayed in each case on the index card **Overwrite** (Fig. 5.43). For this, the signals to be displayed have to be selected first from database view by double-clicking or via main menu. 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 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:



Count	Cycletime	Sta...	Message	Signal	Value	Lower Limit	Upper Limit
481	00.070.028.0		Ignition	Start	On	0	1
481	00.070.028.0		Ignition	Run	On	0	1
481	00.070.028.8	⬆	Engine	Engine Speed	450 rpm	0	6000
481	00.070.028.8		Engine	Engine Temperature	23 degC	0	255
481	00.070.012.0		Gear	Current gear	0	-1	5
481	00.070.012.0		Gear	Selected Gear	0	-1	5
481	00.069.964.8		Vehicle speed	Wheel-based Vehicle Speed	0 km/h	0	120
481	00.069.964.8		Vehicle speed	Ground-based Vehicle Speed	0 km/h	0	350
481	00.069.956.8		ABS	ABS	On	0x00	0x01
481	00.069.980.8		Operator	Accelerator Pedal	10 Percent	0	100
480	00.070.004.8		Wheel info	Wheel Speed RR Inc	0	-10	10
480	00.070.004.8		Wheel info	Wheel Speed RL Inc	0	-10	10
480	00.070.004.8		Wheel info	Wheel Speed FR Inc	0	-10	10
480	00.070.004.8		Wheel info	Wheel Speed FL Inc	0	-10	10

Database Object

- ☑ Ignition
- ☑ Engine
  - ☑ Engine Speed
  - ☑ Engine Temperature
- ☑ Gear
  - ☑ Current gear
  - ☑ Selected Gear

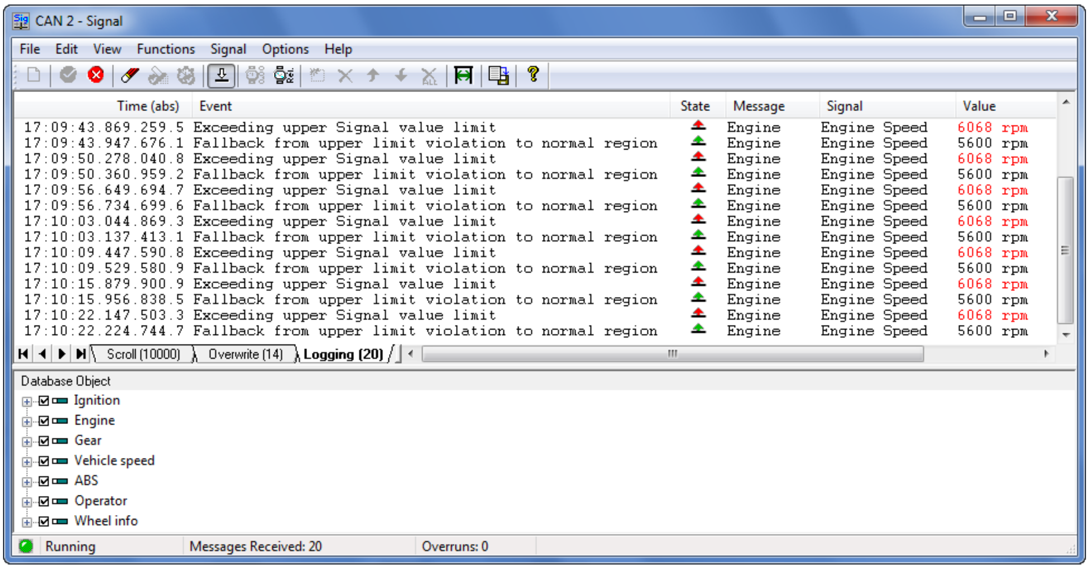
● Running | Messages Received: 6.730 | Overruns: 0

Figure 5.43: Signal module Overwrite View

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
Message	Name of the message in the database used (is usually the alias of the identifier)
Signal	Signal name in the database
Value	Value of the signal
Lower Limit	Signal value's lower limit
Upper Limit	Signal value's upper limit

5.6.4 Logging View

The **Logging** index page (Fig. 5.44) 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

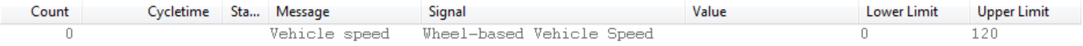


Time (abs)	Event	State	Message	Signal	Value
17:09:43.869.259.5	Exceeding upper Signal value limit	▲	Engine	Engine Speed	6068 rpm
17:09:43.947.676.1	Fallback from upper limit violation to normal region	▲	Engine	Engine Speed	5600 rpm
17:09:50.278.040.8	Exceeding upper Signal value limit	▲	Engine	Engine Speed	6068 rpm
17:09:50.360.959.2	Fallback from upper limit violation to normal region	▲	Engine	Engine Speed	5600 rpm
17:09:56.649.694.7	Exceeding upper Signal value limit	▲	Engine	Engine Speed	6068 rpm
17:09:56.734.699.6	Fallback from upper limit violation to normal region	▲	Engine	Engine Speed	5600 rpm
17:10:03.044.869.3	Exceeding upper Signal value limit	▲	Engine	Engine Speed	6068 rpm
17:10:03.137.413.1	Fallback from upper limit violation to normal region	▲	Engine	Engine Speed	5600 rpm
17:10:09.447.590.8	Exceeding upper Signal value limit	▲	Engine	Engine Speed	6068 rpm
17:10:09.529.580.9	Fallback from upper limit violation to normal region	▲	Engine	Engine Speed	5600 rpm
17:10:15.879.900.9	Exceeding upper Signal value limit	▲	Engine	Engine Speed	6068 rpm
17:10:15.956.838.5	Fallback from upper limit violation to normal region	▲	Engine	Engine Speed	5600 rpm
17:10:22.147.503.3	Exceeding upper Signal value limit	▲	Engine	Engine Speed	6068 rpm
17:10:22.224.744.7	Fallback from upper limit violation to normal region	▲	Engine	Engine Speed	5600 rpm

Database Object
- ☑ Ignition
- ☑ Engine
- ☑ Gear
- ☑ Vehicle speed
- ☑ ABS
- ☑ Operator
- ☑ Wheel info

Running | Messages Received: 20 | Overruns: 0

Figure 5.44: Signal module Logging View



Count	Cycletime	Sta...	Message	Signal	Value	Lower Limit	Upper Limit
0			Vehicle speed	Wheel-based Vehicle Speed		0	120

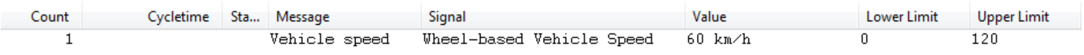
Figure 5.45: Signal added to the Overwrite view

(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:

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 ▲
Message	Name of the message in the database used; is usually the alias of the identifier
Signal	Signal name in the database
Value	Value of the signal
Lower Limit	Signal value's lower limit
Upper Limit	Signal value's upper limit

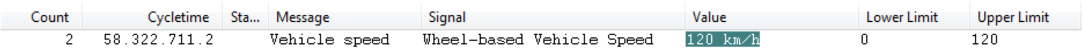
5.6.5 Change Detection

The change detection takes place in the **Overwrite** view of the signal module and can be activated and deactivated via the menu command **Options**. There are change detectors implemented that check against value changes and value range violations. Another check monitors the expected signal cycle time. If a check finds a deviation, a corresponding icon will be shown in the State column. The display of this icon stays, i.e. it will be shown until manually reset, or until the entire View is cleared.
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 message (Fig. 5.45).



Count	Cycletime	Sta...	Message	Signal	Value	Lower Limit	Upper Limit
1			Vehicle speed	Wheel-based Vehicle Speed	60 km/h	0	120

Figure 5.46: First reception of a signal in the Overwrite view



Count	Cycletime	Sta...	Message	Signal	Value	Lower Limit	Upper Limit
2	58.322.711.2		Vehicle speed	Wheel-based Vehicle Speed	120 km/h	0	120

Figure 5.47: Second reception of the signal with changed value

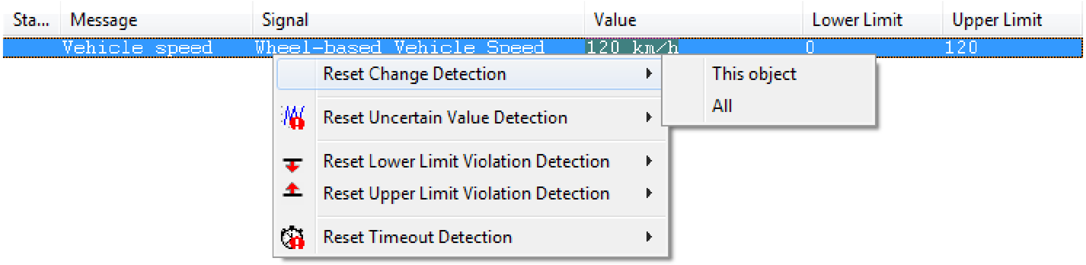
On first reception, the signal is displayed in black font in the overwrite view and with the received value. No **Cycletime** is displayed yet, as for the calculation of this the signal must have been received at least twice (Fig. 5.46).
After the second reception of the signals, the changed value compared with the first reception is highlighted in color in the column **Value** if the data change detection is switched on (Fig. 5.47).
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. 5.48).
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 Filtering

In the tree structure of the database display, all elements, except for the multiplexer mask, have a checkbox symbol (☑). These checkboxes are used in the Signal module for filter configuration. All signals whose checkboxes were set (☑) pass the filter and are displayed. All signals whose checkboxes are not set (□) are filtered out and not displayed.

5.6.7 Interpretation of a trace recording

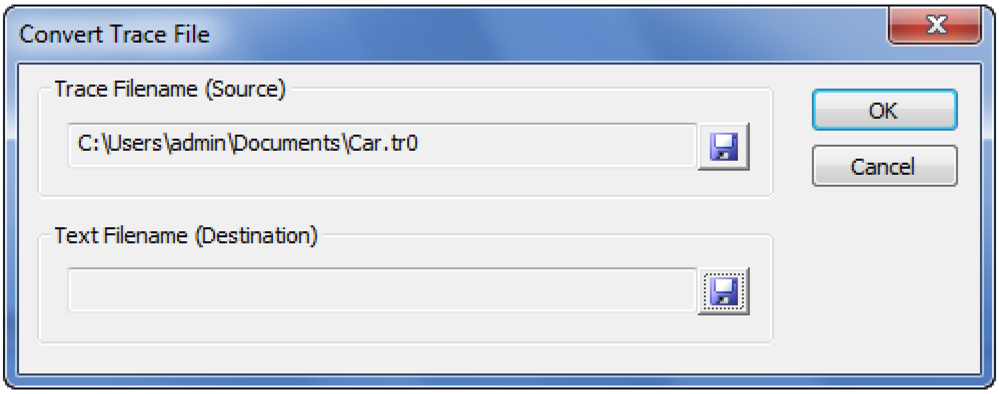
A trace recording generated by the trace module can be interpreted in a signal trace file with the project database. This is coded as a CSV (Comma Separated Value) textfile and contains all columns displayed by the signal module separated by comma or semicolon characters. Further processing of the signal trace file is thus possible in standard tools such as Excel.
In order to interpret a trace recording in a CSV text file, the menu entry **File | Convert Trace File...** is selected. A dialog then opens (Fig. 5.49), in which the trace file to be interpreted and



Sta...	Message	Signal	Value	Lower Limit	Upper Limit
	Vehicle speed	Wheel-based Vehicle Speed	120 km/h	0	120

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

Figure 5.48: Context menu of Overwrite View



Trace Filename (Source)

C:\Users\admin\Documents\Car.tr0

Text Filename (Destination)

Figure 5.49: Interpretation of a trace recording

the name of the signal trace file to be generated is defined. The file is selected via the button 💾.

After acknowledging the dialog with the **OK** button, the Signal module interprets the trace recording in the specified text file.

Note:

Interpretation of trace recordings considers the current filter settings and is only possible with stopped interpretation in the Signal module.

5.6.8 Menu reference

File menu

Menu point	Function
New	Creates a new, empty configuration
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 View...	Exports the current View's contents to an ASCII file
Convert Trace File...	Converts a trace recording to an ASCII file
Exit	Exits the Signal module

Edit menu

Menu point	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

* Not available in Overwrite View

View menu

Menu point	Function
Time rel.	Displays the time stamp absolute or relative to the previously displayed signal
Show recent Frames	Always the most recent telegrams are displayed in scroll mode
Toolbar	Shows/hides the toolbar
Status Bar	Shows/hides the toolbar
Autosize Columns	Regulates ideal column widths

Functions menu

Menu point	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 Uncertain Value Detection All	Resets the uncertain value flag 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
Clear All	Deletes the contents of all Views

Signal menu

Menu point	Function
Add	Adds the selected signal to the overwrite view
Remove	Removes the selected signal from the overwrite view
Remove All	Removes all signals from the overwrite view
Move up	Moves the selected signal upwards by one position
Move down	Moves the selected signal downwards by one position

Options menu

Menu point	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
Fonts...	Opens a dialog to select the font type in which the data are displayed in all Views

Help menu

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

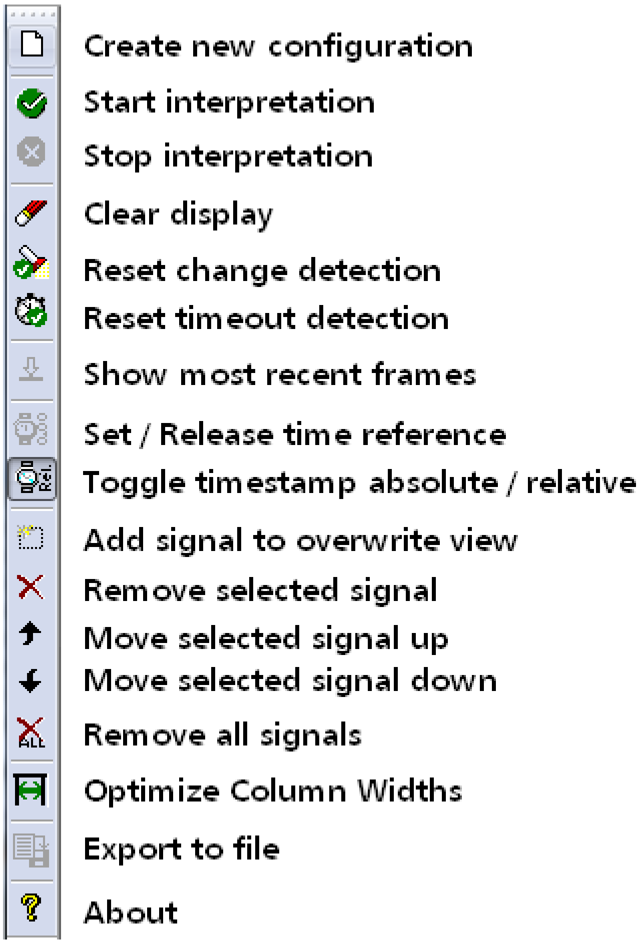


Figure 5.50: Toolbar of the Signal module

5.6.9 Toolbar

The most important functions of the Signal module can be called via the toolbar (Fig. 5.50).

5.6.10 Status bar

The status bar of the Signal module contains an LED symbol which displays the status of the control panel or of the Signal module:

LED color	Meaning
Green	Control panel and Signal module are started
Red flashing	Control panel is stopped
Red	Signal module is stopped.

5.6.11 Hotkeys

TAB	Switch between message Views and database Signals selection
Ctrl+TAB	Switch between the different Views
F1	Online-Help
F2	Go to Next Marker
Shift+F2	Go to Previous Marker
Ctrl+F2	Toggle Marker
F5	Start message reception
Shift+F5	Stop message reception
F8	Clear all Views
Ctrl+C	Copy marked lines CSV formatted to clipboard
Ctrl+E	Export current View contents to a file
Ctrl+O	Load module settings from a file
Ctrl+S	Save module settings to a file
PageDown	Scroll one page ahead in current View
PageUp	Scroll one page backward in current View
Ctrl+PageDown	Scroll 1000 messages ahead in current View
Ctrl+PageUp	Scroll 1000 messages backward in current View
Ctrl+0	Jump to Time Reference message
Ctrl+1..9	Jump to 10%..90% of current View

5.7 Statistic module

5.7.1 Overview

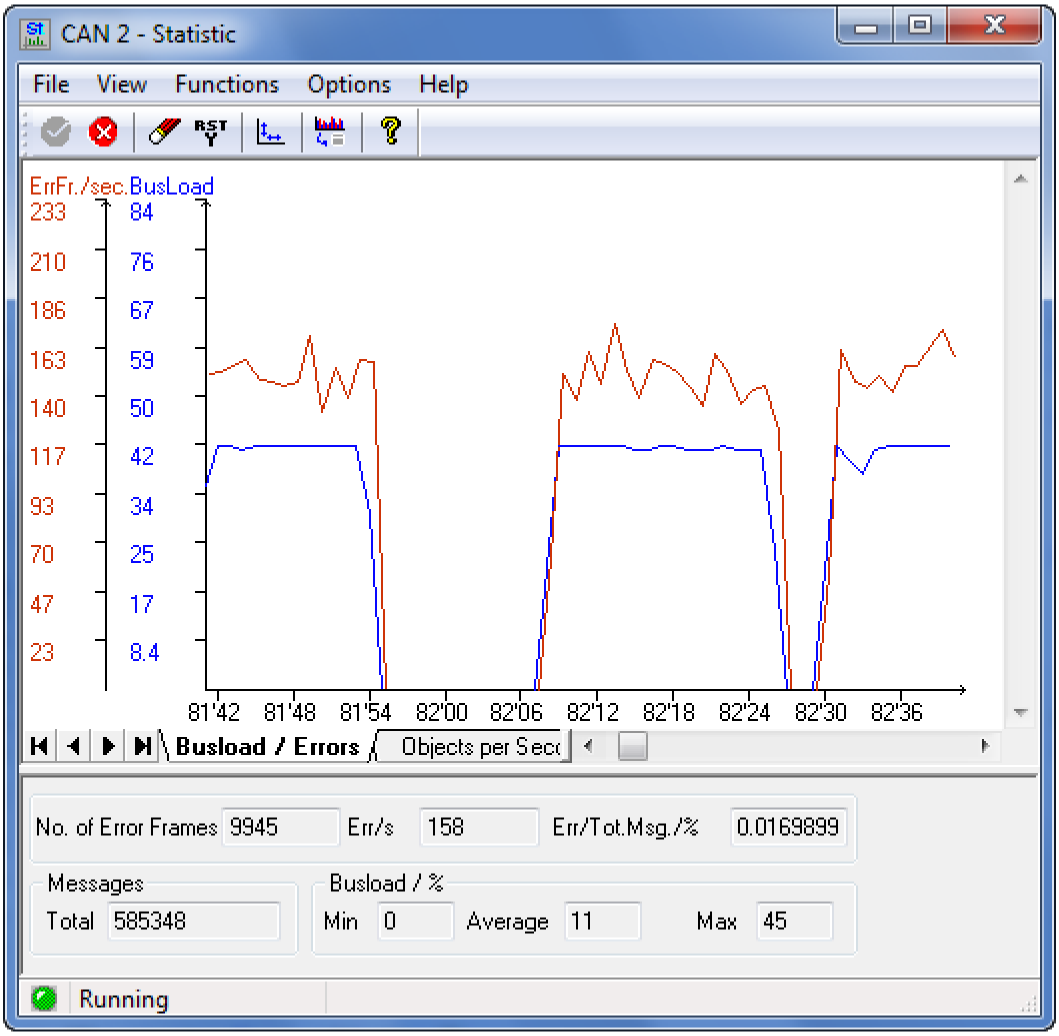
The Statistic module enables the bus load, the number of error telegrams and the identifier-related allocation of messages to be determined and displayed. Three analyses are offered here:

- Bus load: display of the variation over time of the bus load and the error telegrams over a freely definable time axis
- Objects/sec.: Object distribution per second
- Objects total: Display of the object distribution over the measured period of time

5.7.2 Program window

The program window of the Statistic module (Fig. 5.51) is divided into two halves: The top half displays the statistic functions, the bottom half contains general statistical parameters.
The Statistic module provides the following additional analysis functions:

- Saving and loading of statistic data: The data recorded in the recording interval can be saved in a file and loaded at a later point in time (e.g. for documentation purposes)
- Online protocol of the bus load: Bus load and error telegrams per second can be recorded in a text file



No. of Error Frames 9945 Err/s 158 Err/Tot.Msg./% 0.0169899

Messages
Total 585348

Busload / %
Min 0 Average 11 Max 45

Running

Figure 5.51: Statistic module

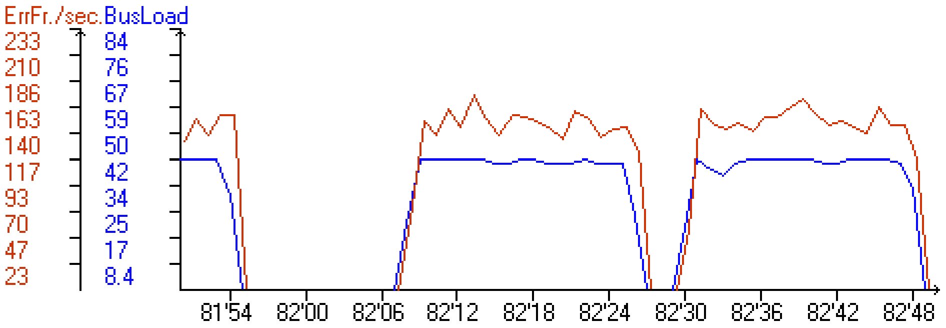


Figure 5.52: Bus load view, error rate

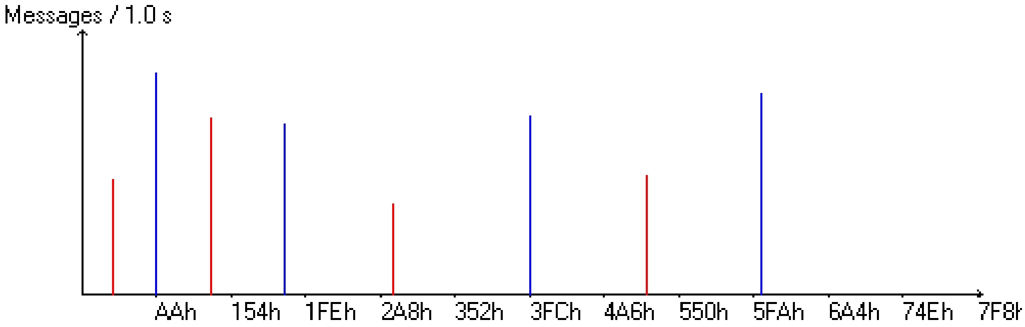


Figure 5.53: Distribution of messages, display of messages/sec

5.7.3 Variation of bus load and error telegrams

The variation of the bus load and error telegrams is displayed graphically in the view **Busload/-sec** (Fig. 5.52). The vertical scale shows the bus load in per cent.
If error telegrams have been received, these are displayed on a 2nd axis in red. The vertical scale shows telegrams per second.
If automatic adaptation of the scaling has been set in the configuration dialog, the division of the axes adapts automatically on start-up to the maximum recorded bus load since the start-up time or to the maximum number of error telegrams received per second.

5.7.4 Statistical distribution of messages

The view **Objects/sec** (Fig. 5.53) provides information on the statistical distribution of messages according to the identifiers. The number of messages received per second is superimposed on the message identifier. For better differentiation in the case of adjacent identifiers, they are displayed in different colors.
The automatic scaling of the axis can be switched on or off in the configuration dialog.
Graphic display of the message distribution is only possible for the identifier 0 - 2047.

5.7.5 Absolute message distribution

In the view **Objects** (Fig. 5.54) all messages during the observation period are displayed accumulated and superimposed on the object identifiers.
Graphic display of the message distribution is only possible for the identifier 0 - 2047.

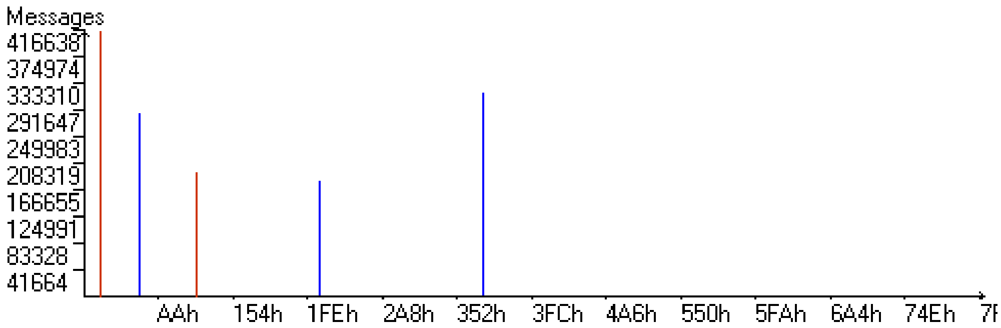
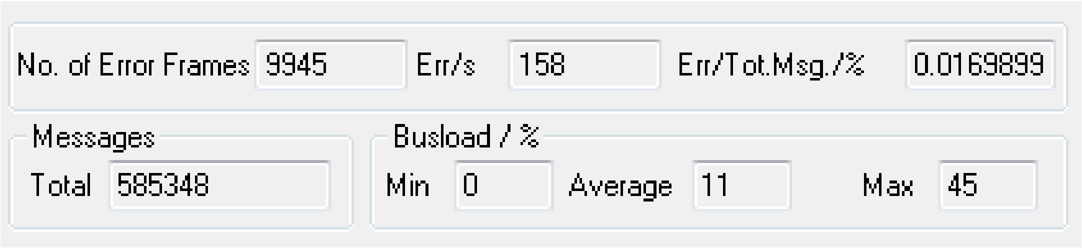


Figure 5.54: Distribution of messages, display of messages absolute



No. of Error Frames 9945 Err/s 158 Err/Tot.Msg./% 0.0169899

Messages

Total 585348

Busload / %

Min 0 Average 11 Max 45

Figure 5.55: Numerical view

5.7.6 General parameters for message traffic

The following statistical parameters are displayed numerically in the bottom half of the display window (Fig. 5.55):

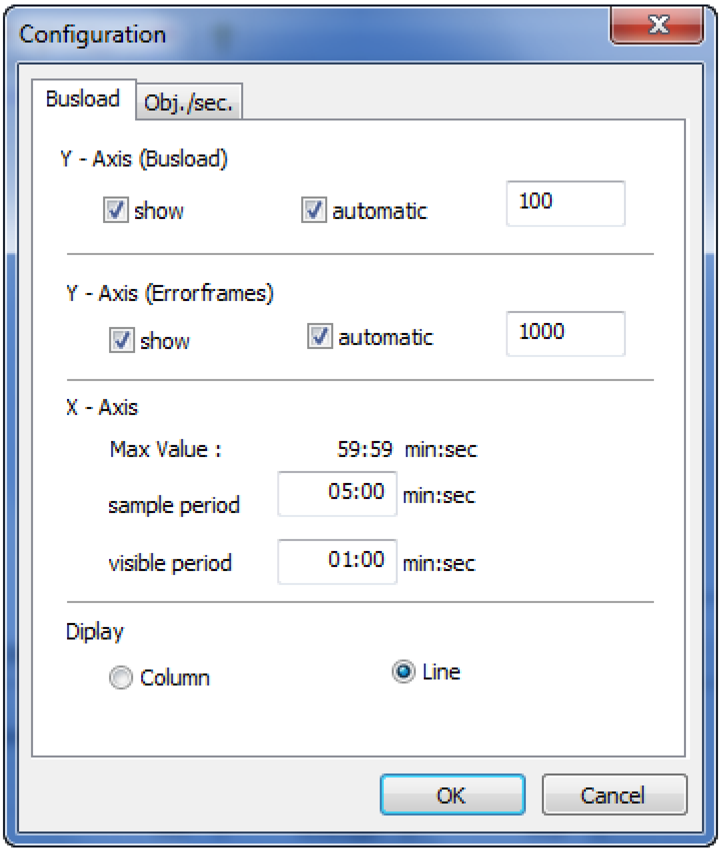
Parameter	Meaning
Number error frames	The number of error telegrams which have occurred since the canAnalyser was started
Messages Total	The number of all messages received
Errors/sec	Number of error telegrams per second
Errors/t. message[%]	Ratio of error telegrams to CAN messages received in per cent
Bus load min	Minimum bus load during the observation time
Bus load average	Average bus load during the observation time
Bus load max	Maximum bus load which has occurred during the observation time

5.7.7 Configuration

The view of the display can be scaled in the **Configuration** dialog. The possible settings for this are described in the following.

Configuration of bus load and error telegram display

The settings for the display of the bus load are made in the index card **Bus load** in the **Configuration** dialog (Fig. 5.56):



Configuration

Busload | Obj./sec.

Y - Axis (Busload)

☑ show ☑ automatic 100

Y - Axis (Errorframes)

☑ show ☑ automatic 1000

X - Axis

Max Value : 59:59 min:sec

sample period 05:00 min:sec

visible period 01:00 min:sec

Diplay

○ Column ● Line

Figure 5.56: Bus load configuration

Y-axis (bus load)

Setting	Function
show	Switches the display of the bus load on or off
automatic	Switches the automatic scaling of the Y-axis on or off; if this field is not selected, the value entered in the field behind it (given in per cent) is used as the maximum value of the axis

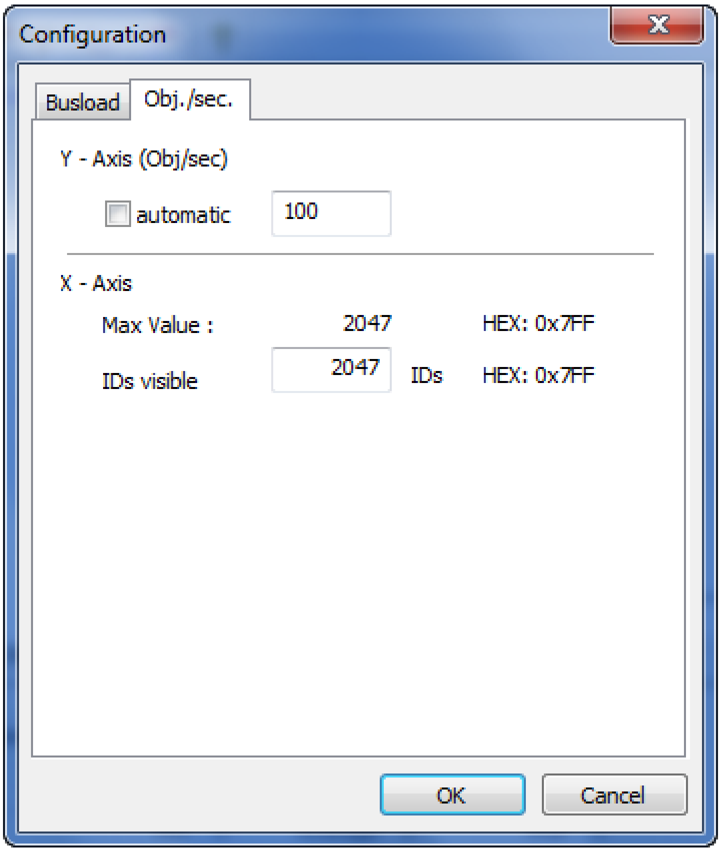
Y-axis (error frames)

Setting	Function
show	Switches the display of the error telegrams on or off
automatic	Switches the automatic scaling of the Y-axis on or off; if this field is not selected, the value entered in the field behind it (given in error telegrams per second) is used as the maximum value of the axis

X-axis

Setting	Function
Sample period	Total recording time
Visible period	Visible area of the recorded time period

The recording duration can be max. 59 minutes 59 seconds, the smallest possible value is 10 seconds.



Y - Axis (Obj/sec)

□ automatic 100

X - Axis

Max Value : 2047 HEX: 0x7FF

IDs visible 2047 IDs HEX: 0x7FF

Figure 5.57: Configuration of the display "Messages/second or messages absolute

Displays

Setting	Function
Column	Display as bar diagram
Line	Display as line diagram

Configuration of message distribution

The setting possibilities for displaying messages are on the second index card (Fig. 5.57):

Y-axis (Obj/sec.)

Setting	Function
automatic	Automatic scaling of the Y-axis; if this field is not selected, the value entered in the field behind it (given in messages per second or messages total) is used as the maximum value of the axis

X-axis

Setting	Function
IDs visible	Area of the display (the larger this area is, the less space is available per identifier); the maximum value is 2047 (0x7FF)

5.7.8 Menu reference

File menu

Menu point	Function
Load	Opens a file with saved statistic data
Save as...	Saves the current statistic data in a file
Online Logfile	Starts the recording of statistic data in a file
Exit	Exits the Statistic module

View menu

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

Functions menu

Menu point	Function
Start	Starts the display of the statistics of the received messages
Stop	Stops the display of the statistics
Reset Y-Axis	Resets the values of the (vertical) Y-axes. The size of the Y-axis is re-calculated (with automatic height calculation)
Clear all	Clears the graphic displays and sets the values of the time axis to zero

Options menu

Menu point	Function
Configuration	Opens a dialog in which the module can be configured (see bus load and objects/sec)

Help menu

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

5.7.9 Toolbar

The most important functions of the Statistic module can also be called via the toolbar (Fig. 5.58).

5.7.10 Status bar

The status bar of the Statistic module contains an LED symbol which displays the status of the control panel or of the Statistic module:

LED color	Meaning
Green	Control panel and Statistic module are started
Red flashing	Control panel is stopped
Red	Statistic module is stopped

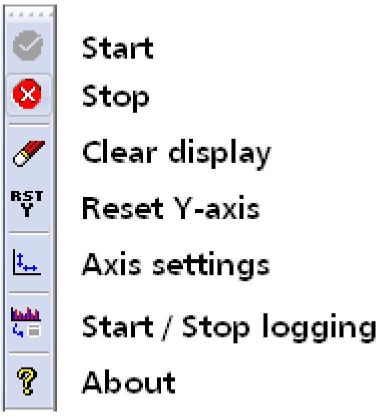


Figure 5.58: Toolbar of the Statistic module

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

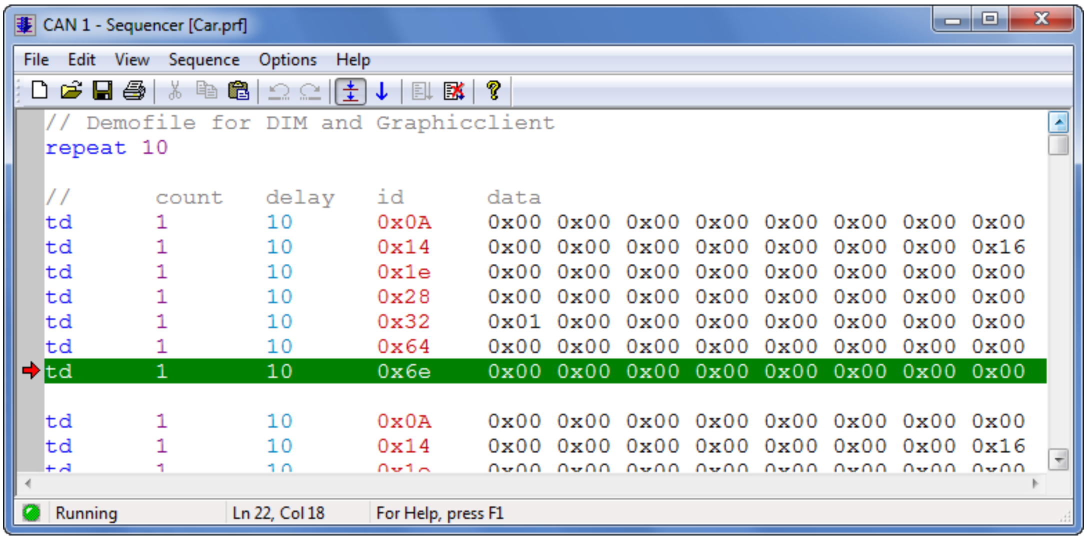
After every command it is possible to add a delay time.
The user interface of the Sequencer module consists of a menu bar, a toolbar, a status bar and the editor window. When processing a sequence, the currently transmitted message of the sequence is visualized via a scroll bar.

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:



```
// 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

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

The Sequencer module supports the following commands:

Command	Meaning	Action
td	transmit data	Transmission of a message
tds	transmit data (std)	Transmission of a message (std)
tde	transmit data (ext)	Transmission of a message (ext)
tr	transmit remote	Transmission of a message request
trs	transmit remote (std)	Transmission of a message request (std)
tre	transmit remote (ext)	Transmission of a message request (ext)
wd	wait for data	Waiting for a message
wr	wait for remote	Waiting for a message request
delay	delay	Adding a delay time in ms
pause	pause	Waiting for user input
repeat	repeat	Repeat of a block
endrep	end repeat	End of a block

Important: The parameter <Delay_time> indicates how long the sequence mode is delayed after a command. This time period can be greater 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 an object, <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 with
//  Data Length Code = 4
//  there is a delay time of 200 ms between messages in each case
//
tr 10 200 33 4
```

Waiting for a message:

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

The sequence mode 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 mode 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
```

Repeat of a block:

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

The command **repeat** indicates the <Number_of_repeats> with which 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 point	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: .tr?) 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 the Sequencer module

Edit menu

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

Message sequence menu

Menu point	Function
Start	Starts the transmission of the currently loaded message sequence
Stop	Stops the Sequence mode
Single step	Execute next single program step

Options menu

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

5.8.5 Toolbar

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

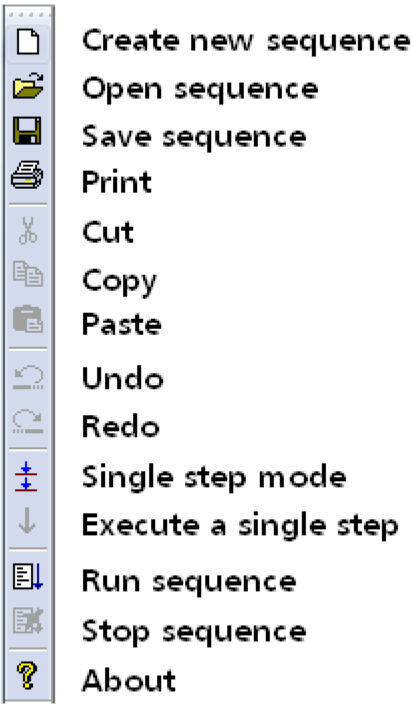


Figure 5.60: Toolbar of the Sequencer module

5.8.6 Status bar

The status bar contains an LED symbol which displays the status of the control panel or of the Sequencer module:

LED color	Meaning
Green	Control panel and sequencer module are started
Red flashing	Control panel is stopped.

5.9 Graphic module

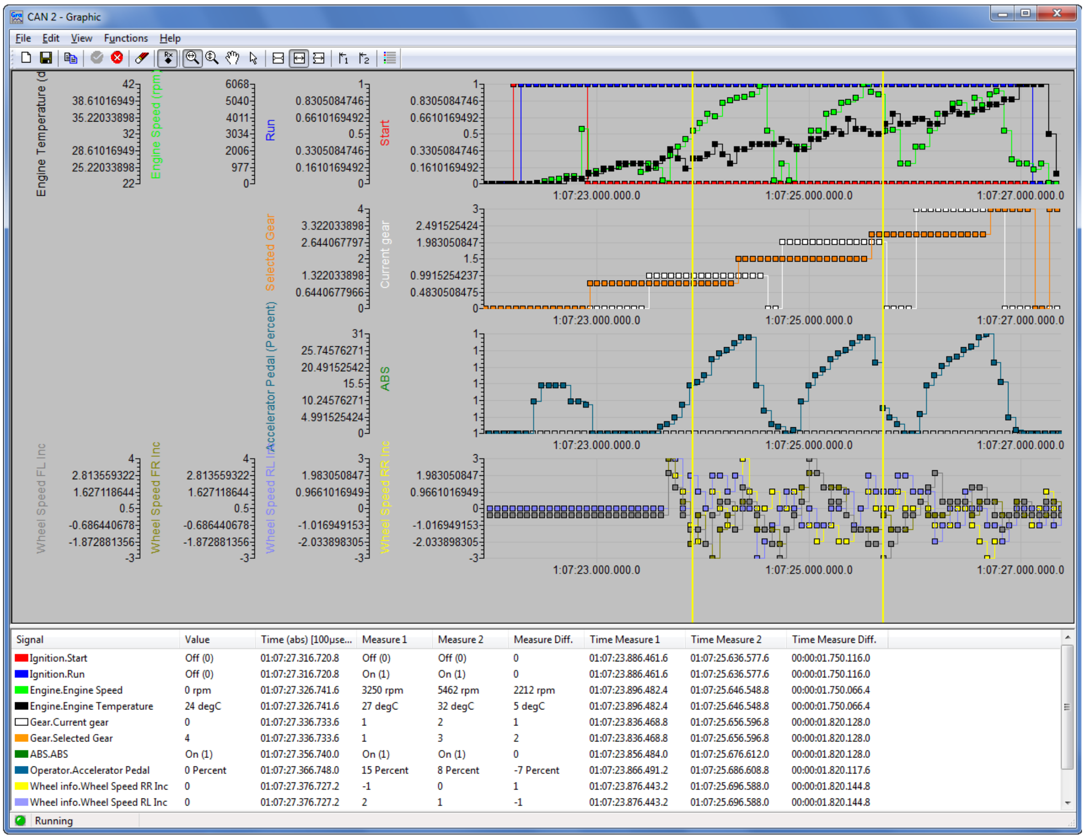
5.9.1 Overview

The basic functionality of the canAnalyser enables the analysis of the messages transmitted in a CAN system based on layer-2 CAN messages (identifier and data). For certain applications, the analysis of a system by interpretation of the transmitted data, which are displayed graphically and in plaintext based on physical variables, is advantageous.

The Graphic module is a module which enables reception of CAN message contents in a format specifically prepared for the user. This module is therefore particularly suitable for installation, testing and maintenance of CAN-based systems, as it graphically visualizes the behavior over time of required measuring or controlling variables.

The Graphic module enables quick and easy analysis of certain measuring and controlling variables. Interrelationships between individual signals can therefore be quickly established and application errors localised more quickly.

The program window of the graphic module (Fig. 5.61) is divided into two halves. In the bottom half, the displayed signals are listed in plaintext with the associated values, the time stamp and the value range. In the top half the signal values are displayed graphically.



Signal	Value	Time (abs) [100µse...	Measure 1	Measure 2	Measure Diff.	Time Measure 1	Time Measure 2	Time Measure Diff.
🟥 Ignition.Start	Off (0)	01:07:27.316.720.8	Off (0)	Off (0)	0	01:07:23.886.461.6	01:07:25.636.577.6	00:00:01.750.116.0
🟦 Ignition.Run	Off (0)	01:07:27.316.720.8	On (1)	On (1)	0	01:07:23.886.461.6	01:07:25.636.577.6	00:00:01.750.116.0
🟩 Engine.Engine Speed	0 rpm	01:07:27.326.741.6	3250 rpm	5462 rpm	2212 rpm	01:07:23.896.482.4	01:07:25.646.548.8	00:00:01.750.066.4
⬛ Engine.Engine Temperature	24 degC	01:07:27.326.741.6	27 degC	32 degC	5 degC	01:07:23.896.482.4	01:07:25.646.548.8	00:00:01.750.066.4
⬜ Gear.Current gear	0	01:07:27.336.733.6	1	2	1	01:07:23.836.468.8	01:07:25.656.596.8	00:00:01.820.128.0
🟧 Gear.Selected Gear	4	01:07:27.336.733.6	1	3	2	01:07:23.836.468.8	01:07:25.656.596.8	00:00:01.820.128.0
🟩 ABS.ABS	On (1)	01:07:27.356.740.0	On (1)	On (1)	0	01:07:23.856.484.0	01:07:25.676.612.0	00:00:01.820.128.0
🟦 Operator.Accelerator Pedal	0 Percent	01:07:27.366.748.0	15 Percent	8 Percent	-7 Percent	01:07:23.866.491.2	01:07:25.686.608.8	00:00:01.820.117.6
🟨 Wheel info.Wheel Speed RR Inc	0	01:07:27.376.727.2	-1	0	1	01:07:23.876.443.2	01:07:25.696.588.0	00:00:01.820.144.8
🟪 Wheel info.Wheel Speed RL Inc	0	01:07:27.376.727.2	2	1	-1	01:07:23.876.443.2	01:07:25.696.588.0	00:00:01.820.144.8

Figure 5.61: Graphic module

5.9.2 Description of functions

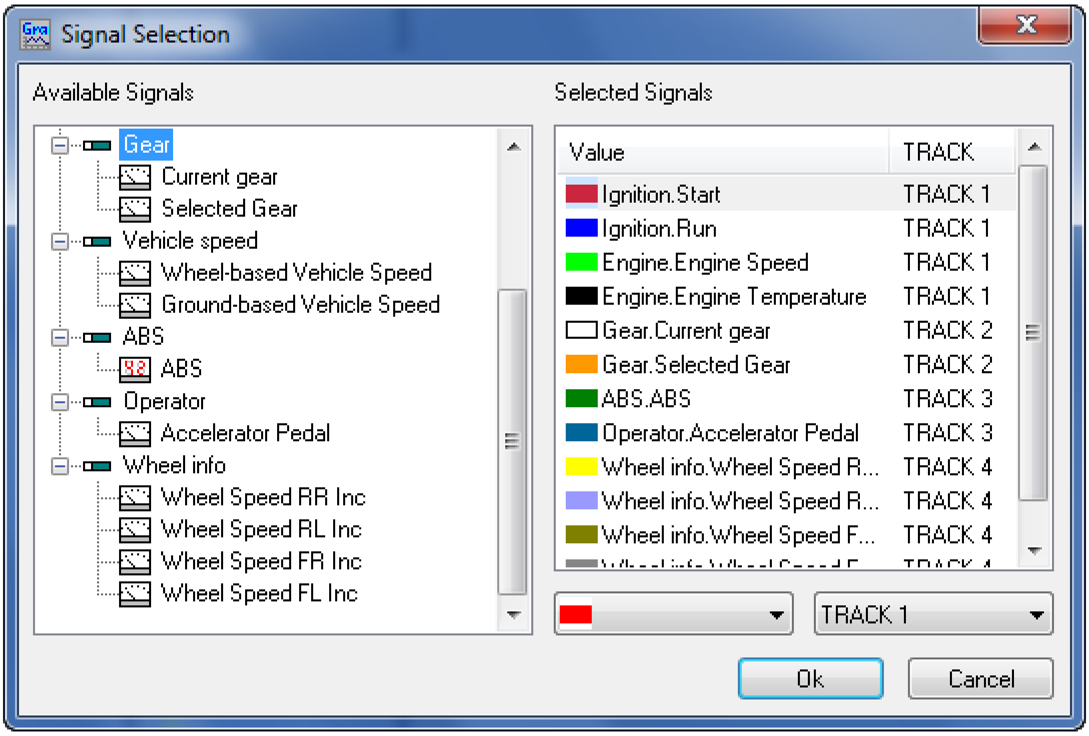
The interpretation and graphic display of the data transmitted in CAN messages is based on a project database. This can be created by the user with the aid of a convenient Database editor. In the project data basis, a CAN message is assigned a symbolic name. This message may contain up to 64 individual data (signals). Each signal can represent analog information, status information or a string:

- Analog signals are described by their bit position and length in the CAN message, data type and data format (Intel/Motorola), scaling and offset, value range and physical unit
- Individual values of signals can be assigned symbolic names, which are displayed as text
- With the string type, the corresponding part of the CAN message is interpreted as an ASCII string

The Data in the project database are saved in XML format. The project database can be extended by further import and export filters for other data formats.
The Graphic module can display up to 16 signals simultaneously, distributed over up to 4 time axes.

5.9.3 Selecting the project database

The Graphic module requires a project database in which the interpretation rules of the data transmitted the data field of a message are stored. The project database is defined in the control panel in the bus settings (see Section 5.1.9).



Value	TRACK
🟥 Ignition.Start	TRACK 1
🟦 Ignition.Run	TRACK 1
🟩 Engine.Engine Speed	TRACK 1
⬛ Engine.Engine Temperature	TRACK 1
□ Gear.Current gear	TRACK 2
🟧 Gear.Selected Gear	TRACK 2
🟩 ABS.ABS	TRACK 3
🟦 Operator.Accelerator Pedal	TRACK 3
🟨 Wheel info.Wheel Speed R...	TRACK 4
🟪 Wheel info.Wheel Speed R...	TRACK 4
🟫 Wheel info.Wheel Speed F...	TRACK 4

Figure 5.62: Signal selection dialog in the Graphic module

5.9.4 Selection of the signals to be visualized

The Graphic module can display up to 16 signals simultaneously in up to 4 coordinate systems. These are selected in advance by the user. After opening a project database, the signals to be displayed are selected via a dialog (Fig. 5.62) which is called via the menu point **Functions | Select signals...** or by the ☰ icon of the toolbar.
The dialog for signal selection contains the following elements:

Element	Meaning
Available signals	Tree of the data interpreter project database with all signals that can be displayed
Selected signals	List of all signals whose value variation is to be displayed in the Graphic module
Color	Assignment of the color in which a signal is displayed
Track	Assignment of the track (coordinate system) in which the signal is to be displayed; four tracks are available

Assignment of a signal to the list of signals to be displayed is made by double clicking on an available signal, resp. by Drag-and-Drop. The same way a selected signal could be removed again from the list of signals to be displayed. The sequence of selected signals may be changed using Drag-and-Drop of any list entry.

5.9.5 Start interpretation

The Interpretation is started via the menu entry **Functions | Start** or via the ✓ button of the toolbar.
The Graphic module then begins with the graphic display of the received values for the selected signals. It is possible to scroll and zoom online (Fig. 5.61).

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 by a CAN message for a signal, this is apparent from the missing time stamp. The Graphic module shows the default value entered in the data interpreter project database for a signal until a first value is received.
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.
- 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.

5.9.6 Analysis

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 "Value" (Fig. 5.61).
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.
- 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.

5.9.7 Creating a project database

For the description of the definition of signals and creation of a project database, please refer to the Database Editor manual.

5.9.8 Notes

Data types that can be displayed

The Graphic module cannot display any signals of string type. In addition, selection of integer 64-bit signed/unsigned signals is not possible.

CAN protocol type

The Graphic module can only function correctly if the CAN protocol type of the project database matches the CAN protocol type set in the control panel.

Size of the FIFO buffer

The Graphic module works on the principle of a FIFO buffer. If the memory provided for signal values is full, the oldest values are always overwritten with the newly received messages. The FIFO buffer holds 10,000 CAN messages.

5.9.9 Menu reference

File menu

Menu point	Function
New	Reset Graphic module
Import Options...	Import module settings from a file (see section 1.4)
Export Options...	Export module setting to a file (see section 1.4)
Export Signal Curves...	Create an image file from the currently visible time window
Exit	Exit the Graphic module

Edit menu

Menu point	Function
Copy	Copies the current graphic and the displayed values to the clipboard. They could be inserted into other programs via the "Insert contents" command.

View menu

Menu point	Function
Toolbar	Shows/hides the toolbar
Status bar	Shows/hides the status bar

Functions menu

Menu point	Function
Start	Start interpretation
Stop	Stop interpretation
Select Signals...	Select signals to be viewed
Mark Signal Receptions	Shows/hides marking on signal reception
Clear all	Reset interpretation

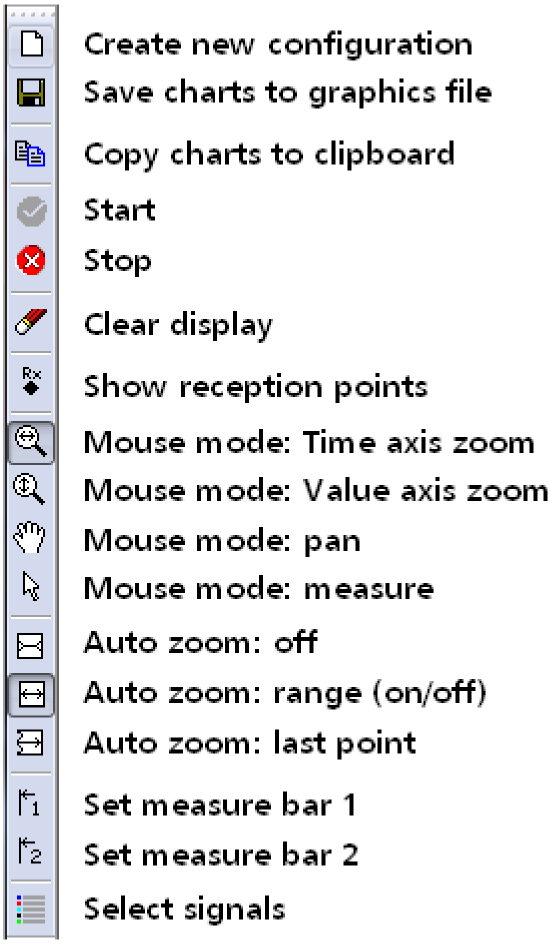


Figure 5.63: Toolbar Graphic module

Help menu

Menu point	Function
Help Topics	Opens the online help of the Graphic module
About Graphic...	Opens the display of the version information of the Graphic module

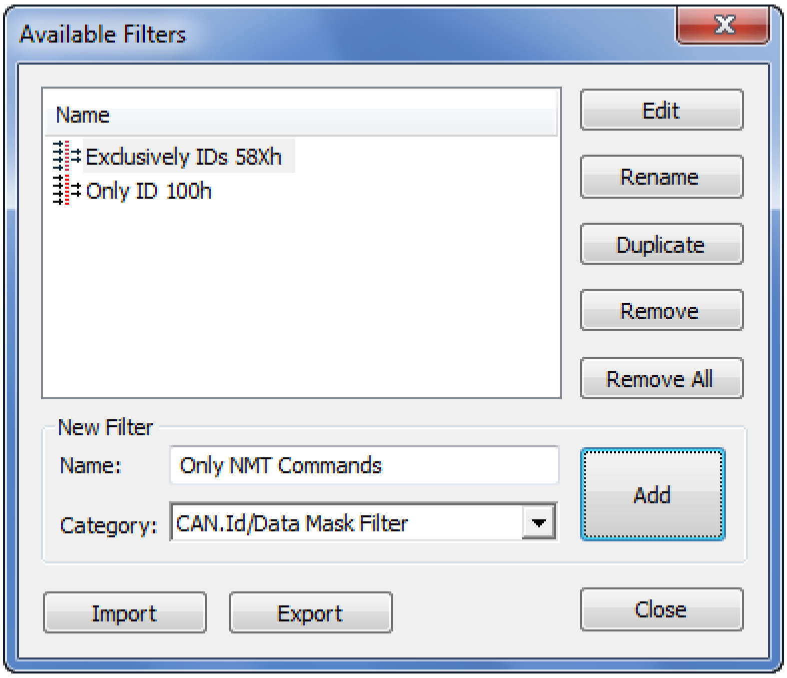
5.9.10 Toolbar

The most important functions of the Graphic module can also be called via the toolbar (Fig. 5.63).

5.9.11 Status bar

The status bar contains an LED symbol which displays the status of the control panel or of the Graphic module:

LED color	Meaning
Green	Control panel and Graphic module are started
Red flashing	Control panel is stopped
Red	Graphic module is stopped.



Available Filters

Name

Exclusively IDs 58Xh

Only ID 100h

New Filter

Name: Only NMT Commands

Category: CAN.Id/Data Mask Filter

Figure 5.64: Creating and configuring filters

5.10 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.

5.10.1 Filter configuration

Creating filters is done with the aid of the **Available Filters** dialog (Fig. 5.64) 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

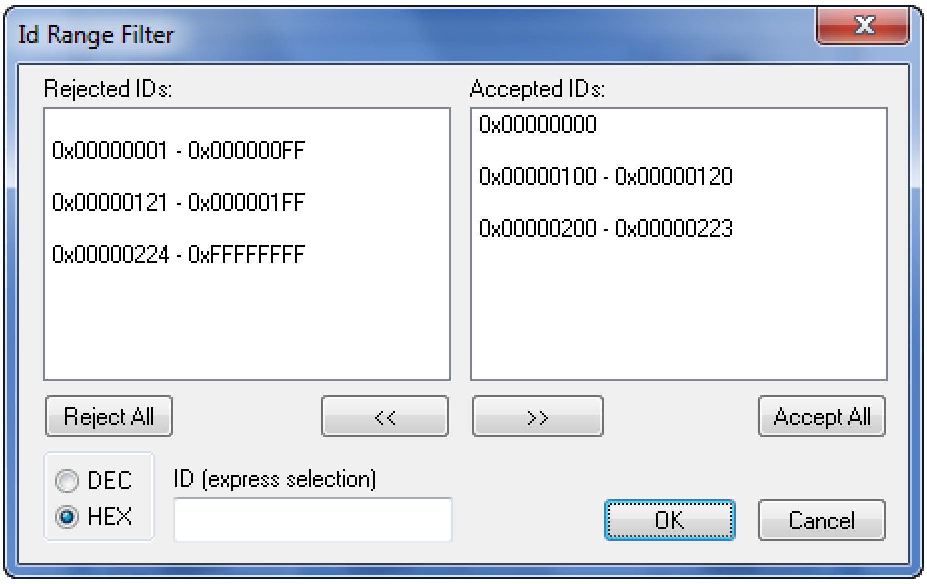
To create a new filter you have to specify a name and a filter category for it. The subsequent click on the button **Add** creates the filter and automatically opens it's configuration dialog. As soon as you closed this dialog with **OK** the new filter will be available within the analysis modules.

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.



Id Range Filter

Rejected IDs:

0x00000001 - 0x000000FF

0x00000121 - 0x000001FF

0x00000224 - 0xFFFFFFFF

Accepted IDs:

0x00000000

0x00000100 - 0x00000120

0x00000200 - 0x00000223

○ DEC
● HEX

ID (express selection)

Figure 5.65: Id Range Filter configuration

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>**.

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).

5.10.2 Id Range Filter

With the aid of the Id Range filter, certain messages become visible or invisible (Fig. 5.65). 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:

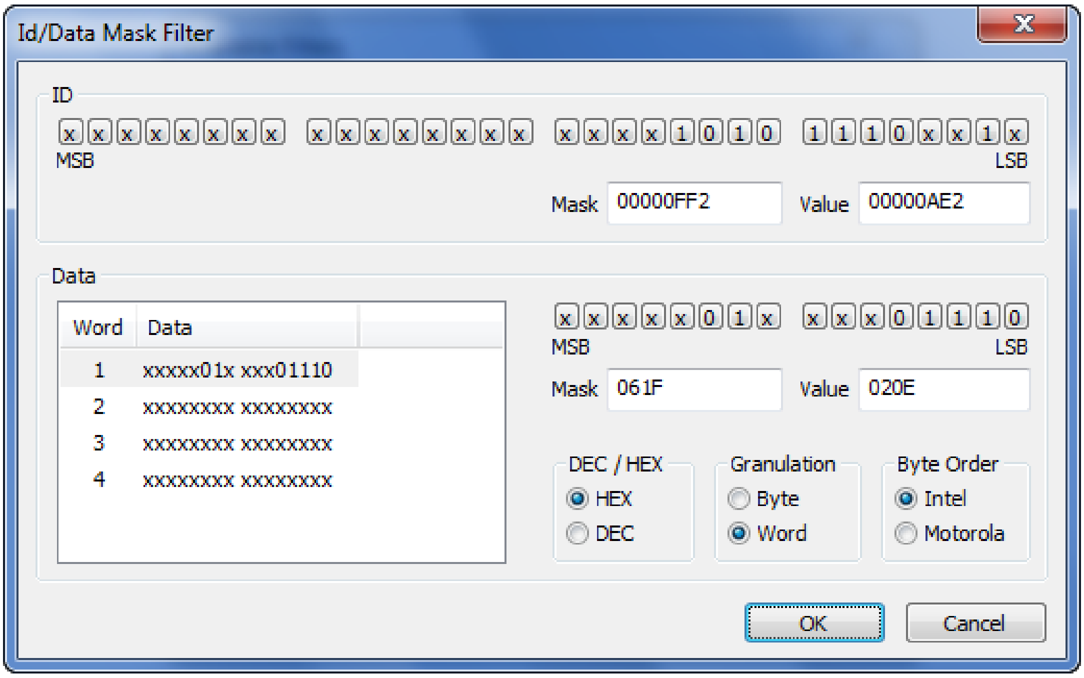
Command	Meaning
-ID	Move identifier ID into the list of rejected IDs
-ID1,ID2	Move identifier array ID1 to ID2 into the list of rejected IDs
+ID	Move identifier ID into the list of accepted IDs
+ID1,ID2	Move identifier array ID1 to ID2 into the list of accepted IDs
z.B.: -3,8	Moves the identifiers 3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

5.10.3 Id/Data Mask Filter

With the Id/data filter (Fig. 5.66), 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 5.66: Id/Data Mask Filter configuration

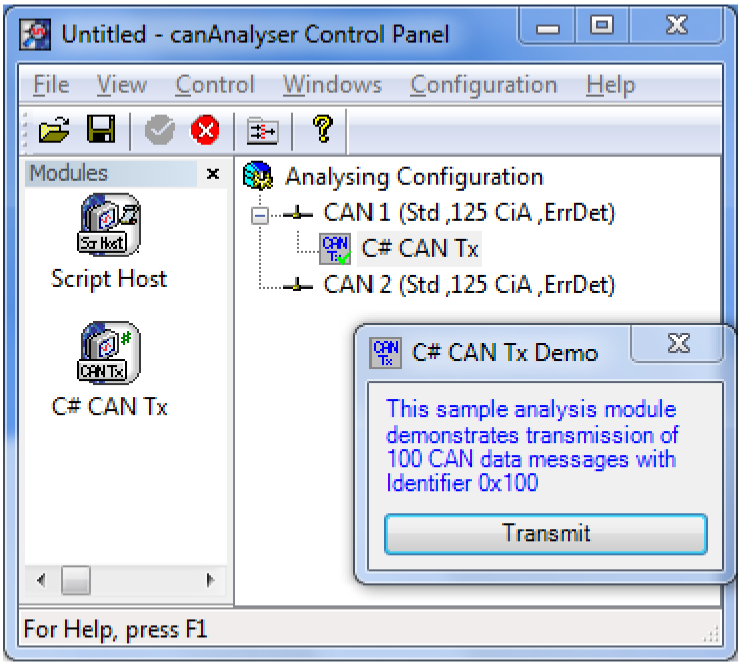
5.11 Integrating own analysis modules

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 V1.1 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 copied into the subdirectory API\UDModules. 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. 5.67).
Any modification at the source code of an analysis module does not go into operation before a restart of the canAnalyser application therefore.


Untitled - canAnalyser Control Panel
File View Control Windows Configuration Help
Modules
Script Host
C# CAN Tx
Analysing Configuration
CAN 1 (Std ,125 CiA ,ErrDet)
C# CAN Tx
CAN 2 (Std ,125 CiA ,ErrDet)
C# CAN Tx Demo
Transmit
For Help, press F1

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

Figure 5.67: Control Panel with "C# CAN Tx" sample analysis module

5.12 Script Host

5.12.1 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.

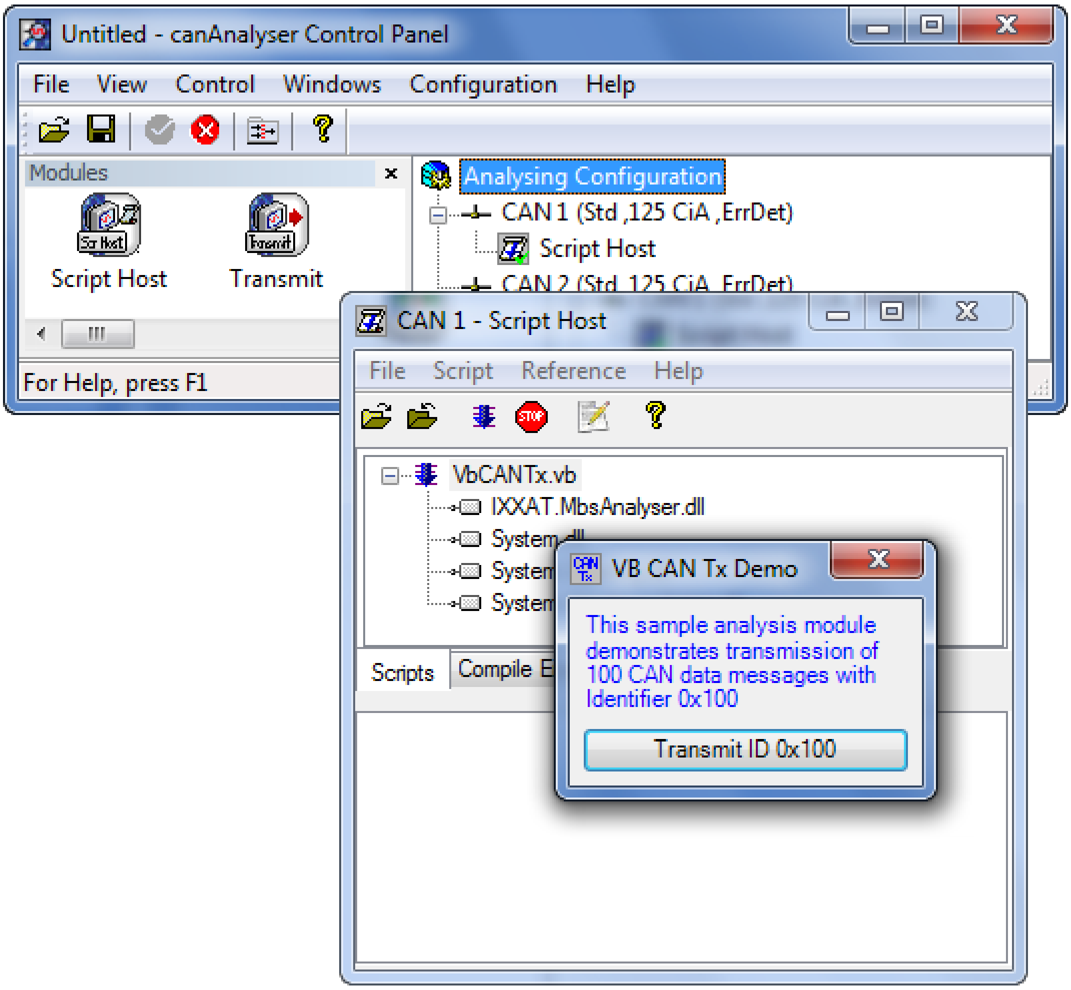
For configuring and executing scripts the canAnalyser Control Panel provides a Script Host as analysis module within the Modules window (Fig. 5.68). In here executable scripts are based on the same .NET 2.0 programming interface as used for integration of own analysis modules (Fig. 5.69). The Script 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.

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 into the Control Panel (Fig. 5.69). This way integrated debugging is possible anyhow. All programming samples installed with the canAnalyser can alternatively be executed as script or compiled



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

Figure 5.68: GUI and console programming sample executed as script

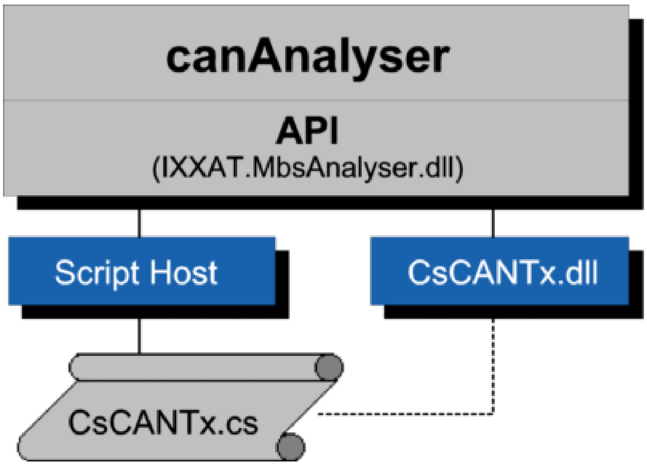


Figure 5.69: Integration of "C# CAN Tx" sample as assembly DLL and via Script Host.

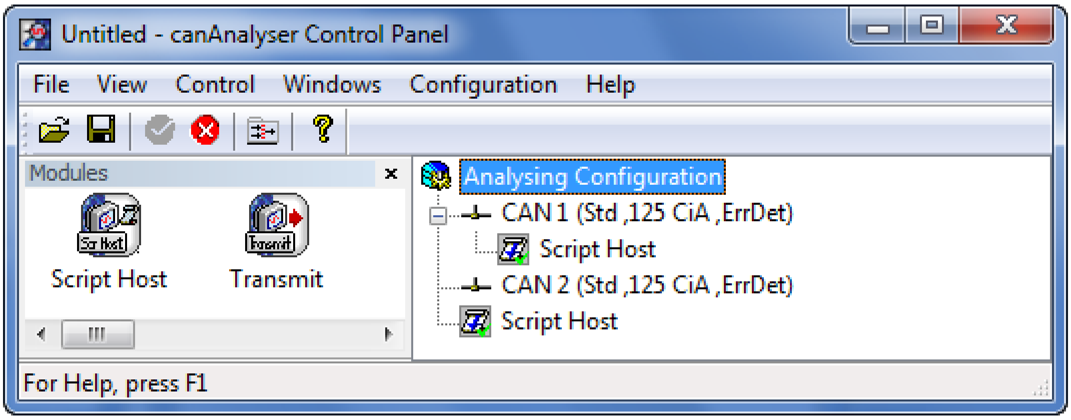


Figure 5.70: Script Host may have access to one or all busses

into an assembly as analysis module. This is only valid for GUI scripts (with graphical user interface). Console scripts cannot be integrated as analysis module. Therefore there is really no way for integrated debugging these.

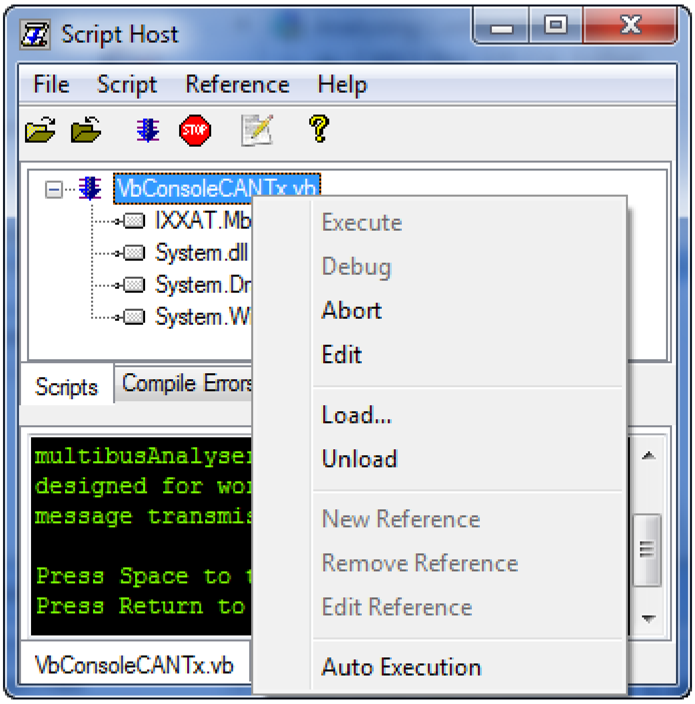
5.12.2 The Script Host within the analysis configuration

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. 5.70). 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.

5.12.3 The Script Host window

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



```
multibusAnalyse
designed for wo
message transmi

Press Space to 
Press Return to
```

Figure 5.71: Script Host with programming samples

- **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. 5.71) 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:

- 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:

- Subdirectory API\UDModules
- Directory \Documents and Settings\All Users\Application Data\IXXAT\canAnalyser\<version>\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.

5.12.4 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.

5.12.5 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.

5.12.6 Stopping scripts

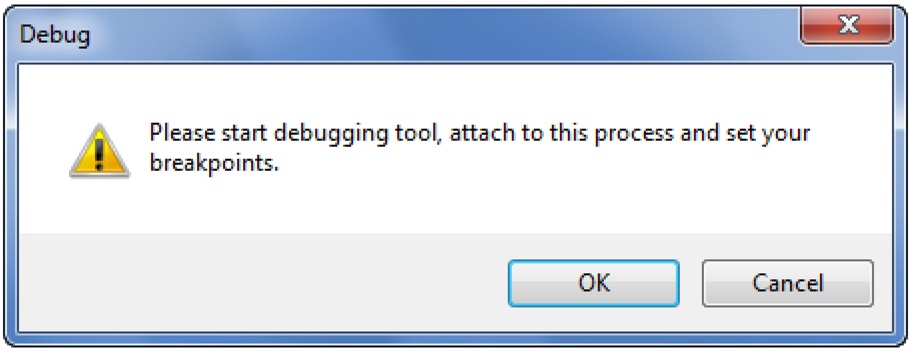
An executed script can be stopped by clicking the button STOP or the corresponding context menu entry or by manually closing the script window. This is signalled by the Script Host with status icon ≣. Also unloading the script via button 📂 or the context menu entry aborts the scripts.

5.12.7 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:

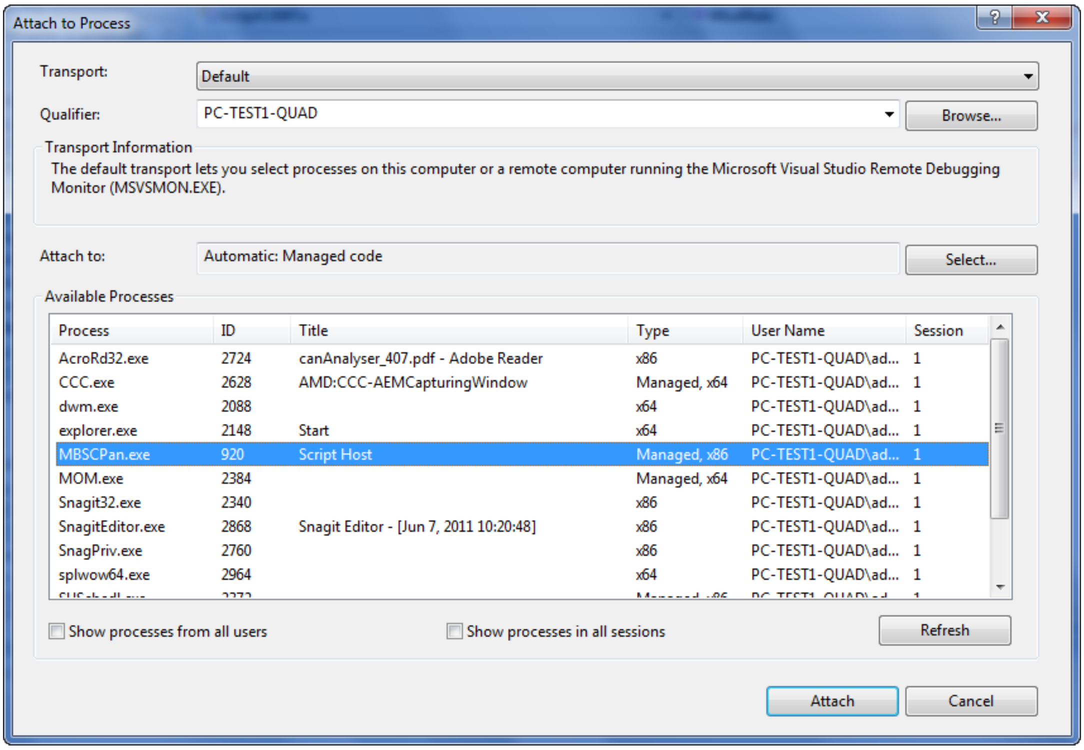
- After the Script Host displayed the dialog shown by Fig. 5.72 start Visual Studio 2008 and execute menu item **Tools | Attach to Process...**.



Debug

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

Figure 5.72: Script Host waiting for attaching debugger



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 5.73: Attaching debugger to the canAnalyser

- In the displayed **Attach to Process** dialog (Fig. 5.73) 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. 5.72).

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.

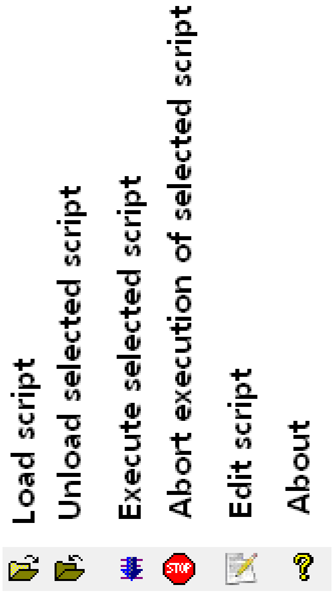


Figure 5.74: Toolbar of the Script Host

5.12.8 Menu reference

File menu

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

Script menu

Menu point	Function
Execute	Executes the selected script
Debug	Executes the selected script with debug information (Chapter)
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

5.12.9 Toolbar

The most important functions of the Script Host module can also be called via the toolbar (Fig. 5.74).

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 2000

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 2.0

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

Microsoft

Apache Software Foundation

Lua.org, PUC-Rio