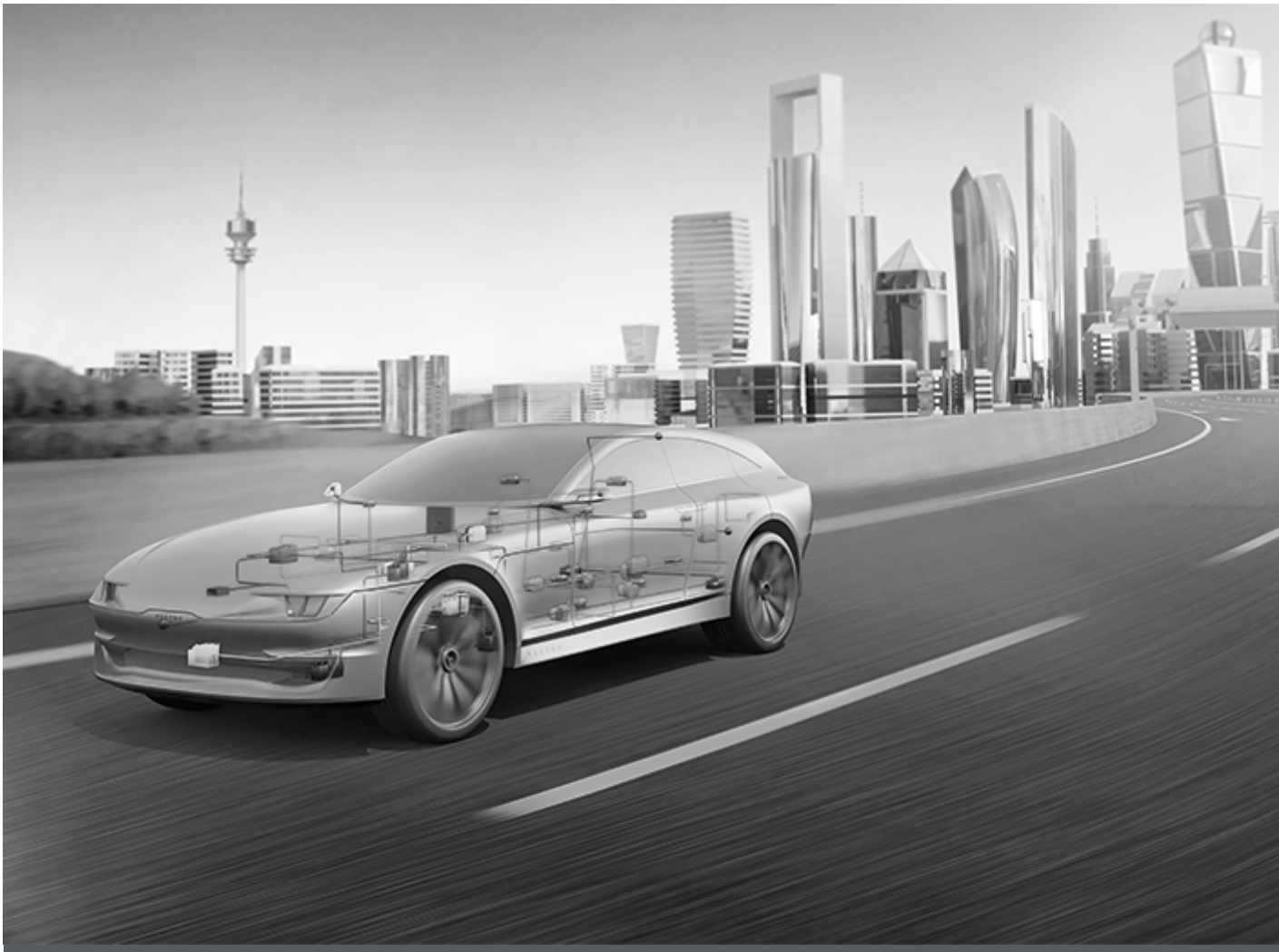
VECTOR



# CANalyzer

Product Information

## Table of Contents

1 Introduction to CANalyzer ..... 4
1.1 Features and Advantages ..... 4
1.2 Application Areas ..... 5
1.3 Bus Systems and Protocols ..... 5
1.4 CANalyzer Variants ..... 5
1.5 System Requirements ..... 6
1.6 Further Information ..... 6
2 Functions ..... 6
2.1 Special Functions ..... 7
2.2 Database Support ..... 7
3 Analysis and Stimulation ..... 7
3.1 Analysis Windows ..... 7
3.1.1 Measurement Setup ..... 8
3.1.2 Trace Window ..... 9
3.1.3 Graphics Window ..... 10
3.1.4 Scope Window ..... 11
3.1.5 Data Window ..... 12
3.1.6 Statistics Window ..... 12
3.1.7 Write Window ..... 13
3.1.8 Map Window ..... 14
3.1.9 Interactive Generator ..... 15
3.1.10 Triggers and Filters ..... 16
3.1.11 Logging/Replay ..... 16
3.2 Data Interpretation ..... 17
3.3 Offline Evaluation ..... 17
3.4 Export Functions ..... 17
3.5 Integrated Desktops ..... 17
3.6 Set Start Values ..... 18
3.7 Symbol Mapping ..... 18
4 Diagnostics ..... 19
4.1 Further Information ..... 22
5 Programming ..... 23
5.1 CAPL Interface ..... 23
5.1.1 C-Like Syntax ..... 23
5.1.2 Event-Oriented Control ..... 23
5.1.3 Symbolic Access ..... 24
5.1.4 Application-Specific Language Extensions ..... 24
5.1.5 CAPL Browser ..... 25
5.2 Visual Sequencer ..... 26
6 Panels ..... 26
7 Software Interfaces ..... 27
7.1 Further Information ..... 27
8 Hardware Interfaces ..... 27

9 Option .Scope ..... 28
9.1 Application Areas ..... 28
9.2 Overview of Advantages ..... 28
9.3 Supported Protocols ..... 28
9.4 Supported Oscilloscope Hardware ..... 29
9.5 Oscilloscope Software ..... 29
9.5.1 Configuration Functions ..... 29
9.5.2 Trigger Functions ..... 29
9.5.3 Analysis Functions ..... 30
9.5.4 Offline Functions ..... 30
10 Training ..... 30

V6.3 06/2020

Valid for CANalyzer from Version 13.0.

This document presents the CANalyzer use areas of analysis, stimulation, diagnostics and their individual functions. The document also contains a brief overview of programming in CANalyzer, supplemental programs and hardware interfaces.

**Product information** and **technical data** on CANalyzer with the .FlexRay, .LIN, .MOST and .CANopen options are presented in separate documents.

# 1 Introduction to CANalyzer

CANalyzer is a universal analysis tool for ECU networks and distributed systems. In addition to observing and analyzing, it is also quite easy to supplement the data traffic in the supported bus systems.

The tools powerful basic functions and user programmability cover all needs from simple network analysis to focused debugging of complex problems.

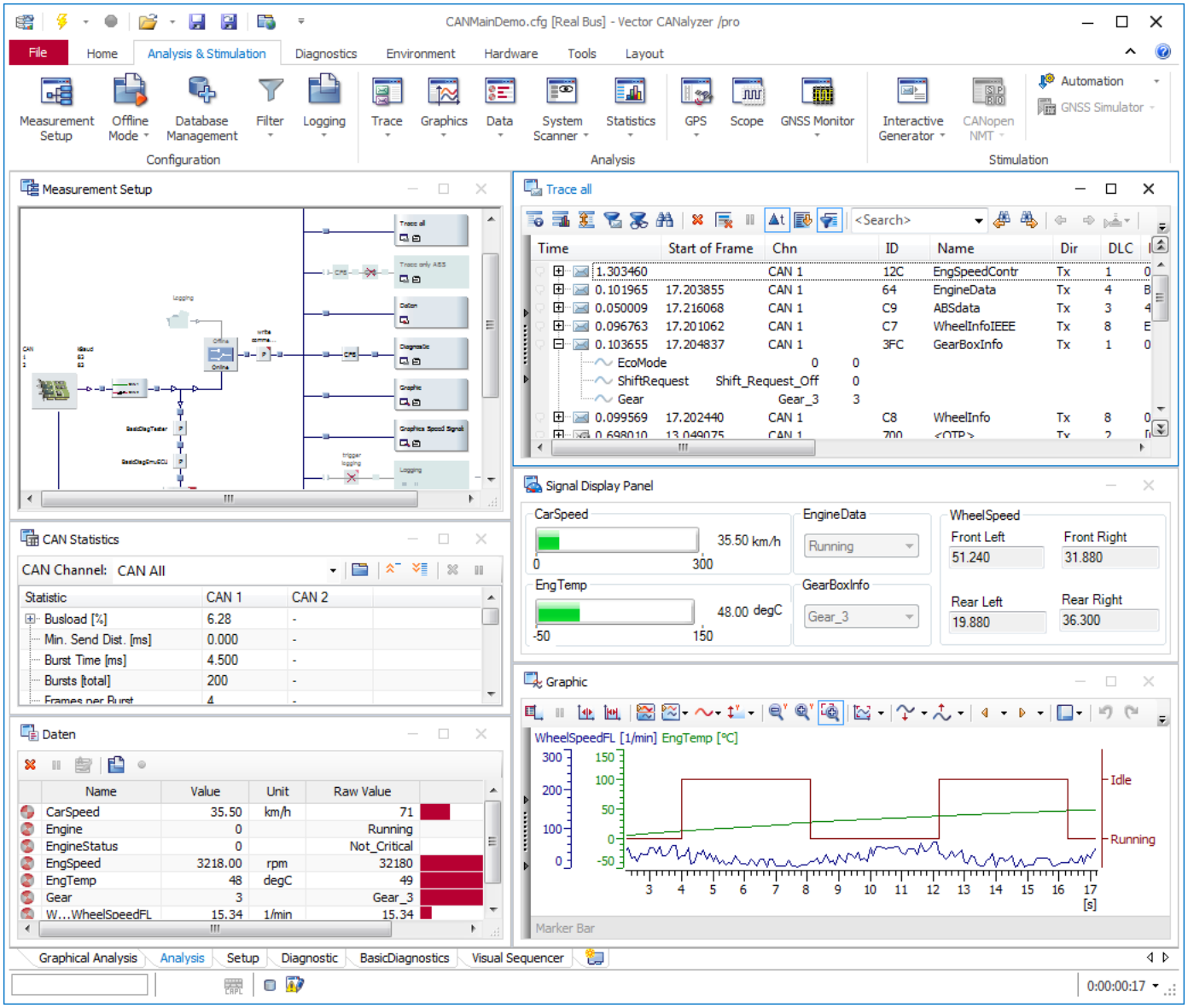



**Figure 1:** Standard CANalyzer configuration for analysis of a CAN system

## 1.1 Features and Advantages

User operation of CANalyzer is very intuitive and is based on a graphic block diagram that depicts the data flow from the bus, over the computer interface and to the various analysis windows of the computer screen and to the logging file. The system is parameterized in this block diagram. Function blocks such as Filter, Generators or Replay Blocks can also be placed and configured there.

## 1.2 Application Areas

CANalyzer is a tool that optimally covers all application areas from a simple network analysis to a high-performance analysis and emulation system. For example, a Replay Block may be placed in the send branch to replay the previously logged data traffic of a functional bus system. Portions of the data traffic may be hidden by adding a downstream filter. This creates a perfect test environment for an ECU in the laboratory.

Other examples of CANalyzer applications are:

- Emulate bus modules
- Create simple remaining bus simulations and test sequences
- Gateway between two buses with ability to manipulate messages
- Program triggered logging of critical phases
- User-specific online evaluation with clear text message
- Simple ECU diagnostics

## 1.3 Bus Systems and Protocols

In CANalyzer, different CANalyzer options are available for the various bus systems and CAN-based protocols; these options may be used in any combination.

CANalyzer supports the following **bus systems**: CAN, CAN FD, LIN, MOST, FlexRay, J1708, Ethernet, K-Line, A429, WLAN and AFDX®[1]

Option .CAN is the basis for these supported **CAN-based protocols**: J1939, CANopen, CANaero.
Others upon request.

You will find detailed information on the options in separate product information documents.

## 1.4 CANalyzer Variants

- **CANalyzer fun**: The "fundamental" variant is ideal for simple analysis purposes, and it provides all standard interactive functions required for them. It does not offer programmability, the diagnostic tester or user control panels.
- **CANalyzer exp**: The "expert" variant is ideal for nearly all standard applications, and it provides all functions and extensions without limitation. The only feature lacking in this variant is the ability to create and execute CAPL programs.
- **CANalyzer pro**: The "professional" variant provides all functions and extensions without limitation. It supports all use cases, ranging from simple observation of the bus traffic to complex analysis, stimulation and testing of heterogeneous systems.

[1] AFDX® is a registered trademark of Airbus

### 1.5 System Requirements

| Component | Recommended | Minimum |
|---|---|---|
| Processor | > Intel Core i7 or comparable<br>> ≥ 3 GHz<br>> ≥ 4 cores | > Intel compatible<br>> 2 GHz<br>> 2 cores |
| | CANalyzer benefits from higher clock rates rather than higher number of cores. | |
| Memory (RAM) | ≥ 32 GB | 8 GB |
| Hard Disk Space | ≥ 20 GB SSD/NVMe | 8 GB HDD/SSD |
| | Depending on options used and required operating system components. | |
| Screen resolution | Full HD | 1280×1024 Pixels |
| Operating system* | > Windows 10 64 bit (≥ version 1803) | > Windows 10 64 bit (≥ version 1803)<br>> Windows 8.1 64 bit<br>> Windows 7 64 bit (≥ SP1) |
| | * Not virtualized. Running in a virtual machine is possible but not tested. Operation with Vector hardware may be affected by virtualization (e.g. higher latencies may occur). | |

### 1.6 Further Information

- Vector Support & Downloads
- **Demo versions**
  Various demo versions are available on the web for CANalyzer. They contain sample configurations for the various application areas as well as a detailed help function, in which all CANalyzer functions are described.
- **Application notes**
  In the following chapters, we refer to additional application notes that offer in-depth information on the individual application areas.
- CANalyzer Feature Matrix
  More information on variants, channels and bus system support is presented in the feature matrix.

## 2 Functions

The product's basic functions give you a broad range of use options; they include:

- Tracing of the bus data traffic
- Graphic and textual display of signal values
- Interactive sending of predefined messages
- Sending out logged messages
- Diagnostic communications per KWP2000 and UDS and use as a full-fledged diagnostic tester
- Statistics on messages
- Statistics on bus utilization and bus disturbances
- Logging of messages for replay or for offline evaluation
- Generation of bus disturbances
- Intuitive user interface with flexible docking concept and user-friendly menu structures
- Support of new Vector bus hardware:
  - VN1610 (2 channels – CAN)

- VN1611 (2 channels – CAN and LIN/K-Line)
- VN1630 (4 channels – CAN and LIN/K-Line)
- VN1640 (4 channels – CAN and LIN/K-Line)

### 2.1 Special Functions

CANalyzer highlights include:

- Simple, intuitive user interface concept
- Fast, interactive process for creating stimulation and test sequences with the Visual Sequencer
- User programmable with CAPL
- Bus-specific modifications to various function blocks
- Diagnostic functions:
  - Parameterization of diagnostics by diagnostic descriptions as ODX 2.0.1/2.2.0, MDX 2.0/3.0 or CDD
  - Definition of simple diagnostic services with the Basic Diagnostic Editor
  - Support of physical and functional addressing
  - Quick and simple On-Board Diagnostics with built-in OBD-II tester
  - Diagnostic observer for UDS and KWP2000 based on parameterizable diagnostic descriptions
  - Transport protocol observer for ISO/DIS 15765-2
  - Support of DoIP (Diagnostics over IP) and HSFZ (High Speed Fahrzeugzugang)

### 2.2 Database Support

CANalyzer supports system descriptions based on the following formats: DBC (CAN), LDF (LIN), XML (MOST), FIBEX (FlexRay) and AUTOSAR descriptions (CAN/FlexRay/Ethernet).

CANalyzer can process the following diagnostic descriptions: CDD (CANdelaStudio), ODX 2.0.1/2.2.0 as PDX files and MDX 2.0/3.0.

Information of these databases can be symbolically displayed and used in CANalyzer.

## 3 Analysis and Stimulation

The basis for analysis in CANalyzer is the data flow from the data source to its display or logging. The data may also be processed. For example, filters can be integrated that define which data should be explicitly considered in an analysis and which data should not. The data traffic can also be influenced by various stimulation options.

Highlights

- Easy to configure the analysis window by drag and drop. For example, this method can be used to copy or move messages or signals from one analysis window to another.
- For a multifunctional analysis, one type of window (e.g. Graphics Window) can be integrated multiple times in the data flow, which enables parallel analysis.
- Easy start and stop logging directly from the status bar.

### 3.1 Analysis Windows

CANalyzer supplies the user with windows and blocks such as those described below.

### 3.1.1 Measurement Setup

The data flow is graphically represented and configured in the Measurement Setup.

- **Define data source** (online/offline)
  The simulated bus or the real bus connected via the hardware (e.g. CANcardXL) serves as the online data source. A file with logged data serves as the offline data source.
- **Insert analysis windows**
  Data can be shown differently in the individual windows depending on analysis requirements, e.g. to graphically represent signal waveforms or to display signal values.
- **Insert CAPL program nodes**
  A CAPL program node can be used for such tasks as filtering data or implementing various arithmetic operations.
- **Insert filters**
  Filters can be used to obtain a more understandable representation of the data; they define which data should be passed and which should be explicitly blocked. Filters may be active during or after the measurement, and the objects filtered may range from individual signals to the channels of an entire bus system.
- **Insert trigger conditions**
  Like filters, trigger conditions can also be used to reduce data. Triggers are specifically configured as a reaction to bus events and can be combined with one another.
- **Log data**
  For an analysis after the measurement, data can be logged in a logging file that can later be reused as an offline data source and replayed.

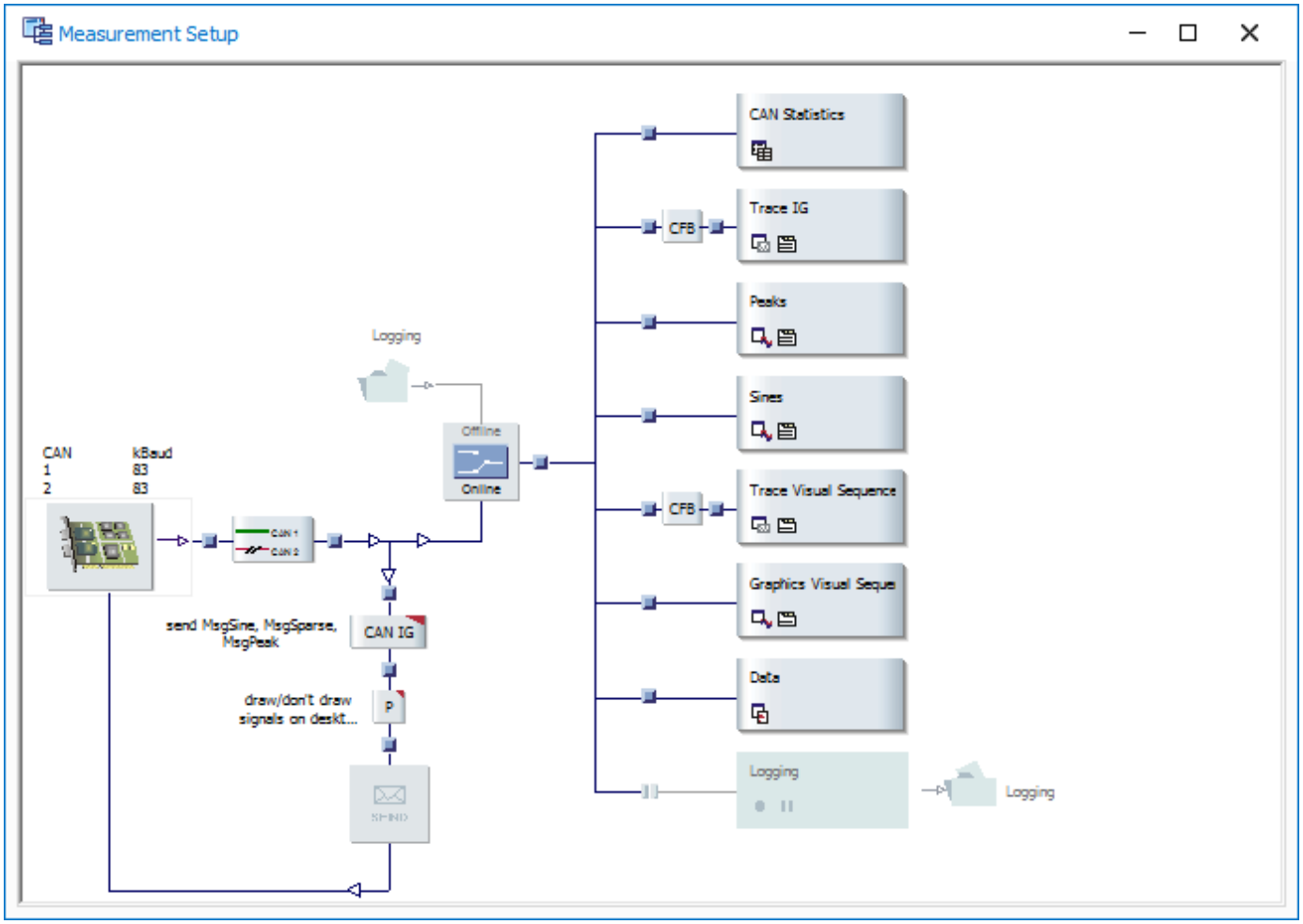



**Figure 2:** Measurement Setup with online data source

### 3.1.2 Trace Window

Bus activities – such as the sending of messages or Error Frames – are listed in the Trace Window. Individual signal values may be displayed for each message. Functions such as those listed below are available for analyzing the data:

- **Insert filters**
  There are various types of filters in the Trace Window. They can be used to reduce the amount of data displayed, and data can even be deleted from the data stream.
- **Hide unchanged data**
  To improve ease of viewing, data that does not change is slowly faded or removed entirely from the screen.
- **Color events**
  Important events and messages can be highlighted in color.
- **Set markers**
  Markers can be set to identify and quickly find events. The marker is assigned to an event and therefore to its time stamp as well. The set markers can also be displayed in other analysis windows.
- **Show statistics**
  Various aspects of messages/signals, including their values, can be displayed in different views in detail. Differences between the time stamps or signal values may also be calculated.
- **Log data**
  It is possible to export some or all of the Trace Window contents. Files that have already been exported can be converted to a different format afterwards, e.g. to further process the same dataset in different programs.

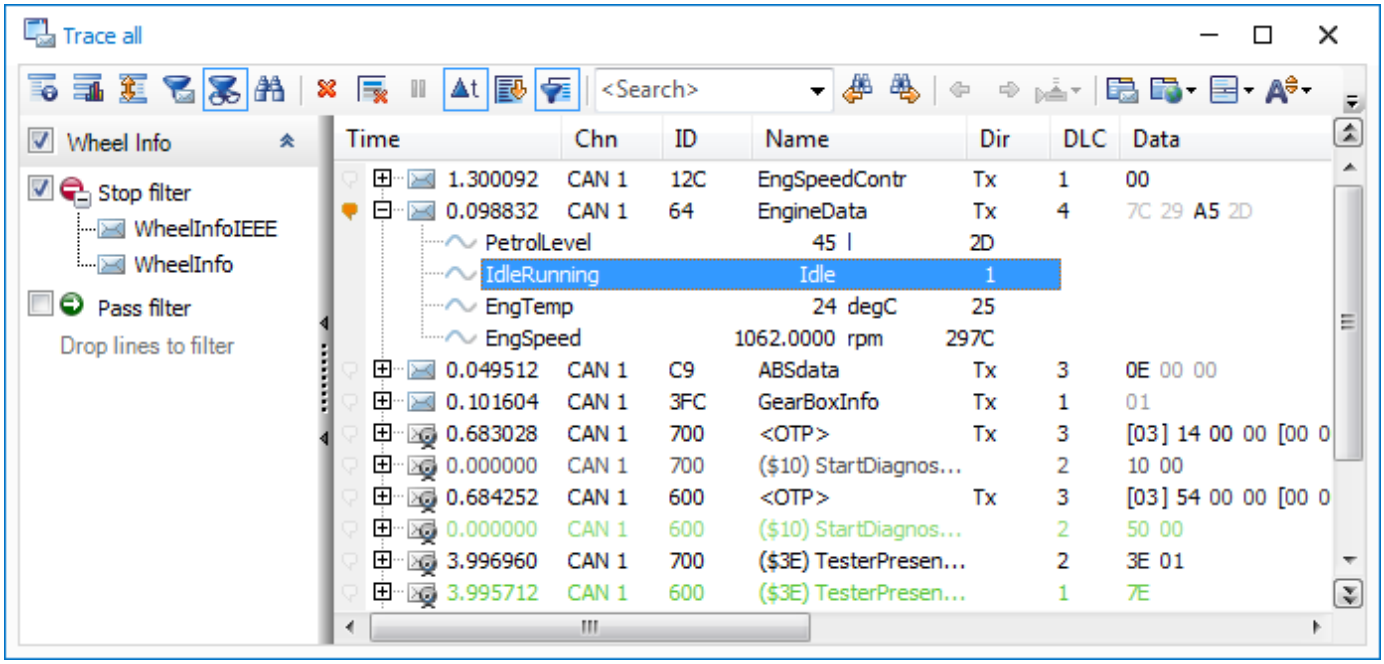



**Figure 3:** Trace Window with active Stop filter and set marker

### 3.1.3 Graphics Window

The Graphics Window is used to graphically display the values of signals, environment data and diagnostic parameters as curves. Listed below are some of the functions available for measurement and evaluation of these curves:

- **Show measurement markers/difference markers**
  Measurement or difference markers can be used to perform absolute or relative analyses of measurement values. The measurement marker can be synchronized to the Trace Window display.
- **Set markers**
  Markers can be set to identify and quickly find events. The marker is assigned to one event and therefore to its time stamp as well. The set markers can also be displayed in other analysis windows.
- **Show measurement columns**
  In the legend, global or local minima and maxima may be shown for each signal, or Y-differences between signals of the same type can be displayed.
- **Show statistics**
  Statistical data such as minimum, maximum, mean value and standard deviation can be compiled for selected signals or all signals of the Graphics Window.
- **X/Y mode**
  By right clicking in the signal list every signal can also be configured as x-axis.
- **Log data**
  Signals of the Graphics Windows can be logged automatically or manually during the measurement. This involves extracting the signals from the messages and saving them in binary form in signal based MDF files.
  In the Graphics Window, the entire signal waveform or just a visible section of the signal waveform can be saved to a file.

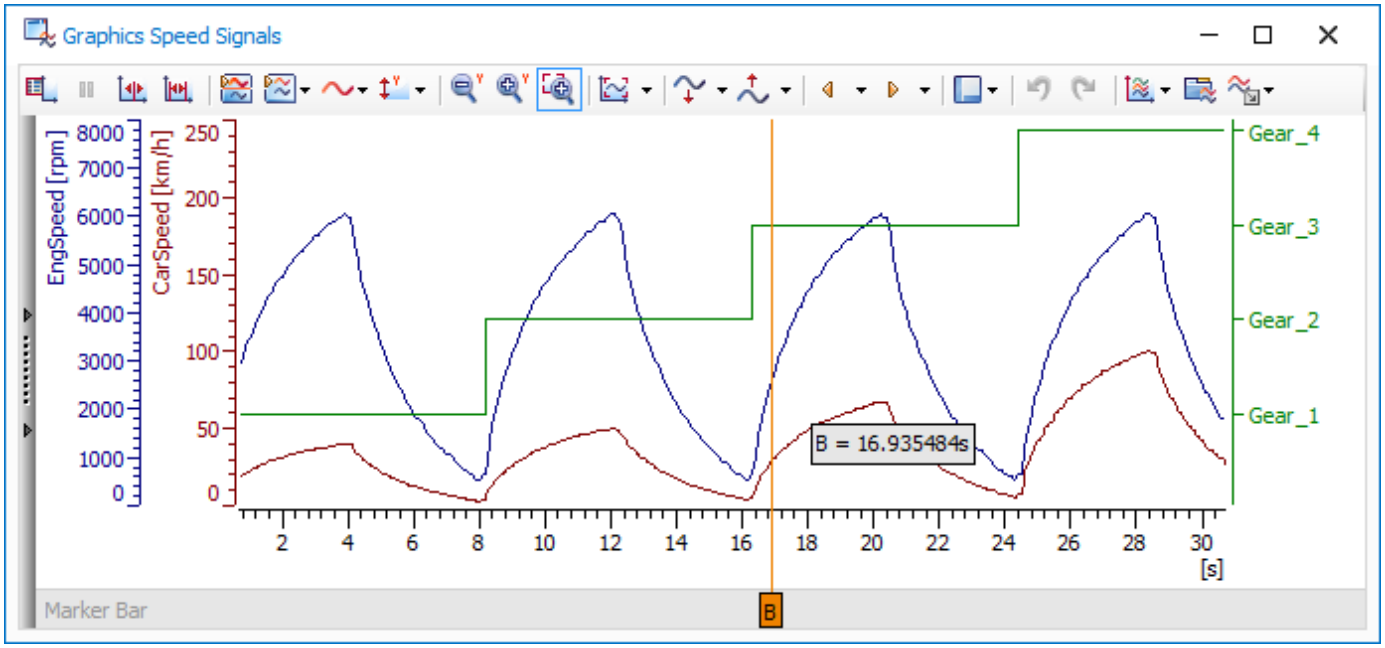



**Figure 4:** Graphics Window with set marker

### 3.1.4 Scope Window

The Scope Window graphically depicts bus level measurements and is used for the analysis of protocol errors (see also option .Scope, Chapter 9).

- **Set triggers**
  In the Scope Window, it is possible to trigger manually, via CAPL or via preconfigured conditions. Any number of trigger conditions may be created, and the individual trigger conditions can be combined in a logical OR relationship.
- **Analyze measurement values**
  The diagram graphically depicts the measurement values and their logical interpretation.
- **Compare signals**
  The user may choose different approaches to comparing data. For example, it is possible to compare different time-based sections of the same data acquisition or time-based sections of different data acquisitions.
- **Log data**
  Acquired data may be exported and then imported for later analysis.
- **Measurement cursors**
  A measurement cursor is useful to measure physical data like time and voltage as well as measuring the difference of these values related to another cursor. Cursor tooltips show the logical values (e.g. dominant /recessive) of bits. All physical values are listed in a separate legend.
- **Global markers**
  In the Scope Window, markers can be set for marking significant points of a measurement. Each marker has a name and a defined time stamp.
- **Eye diagram**
  Enhanced analysis possibilities e.g. superposition of bits or presentation of the "ideal bit".

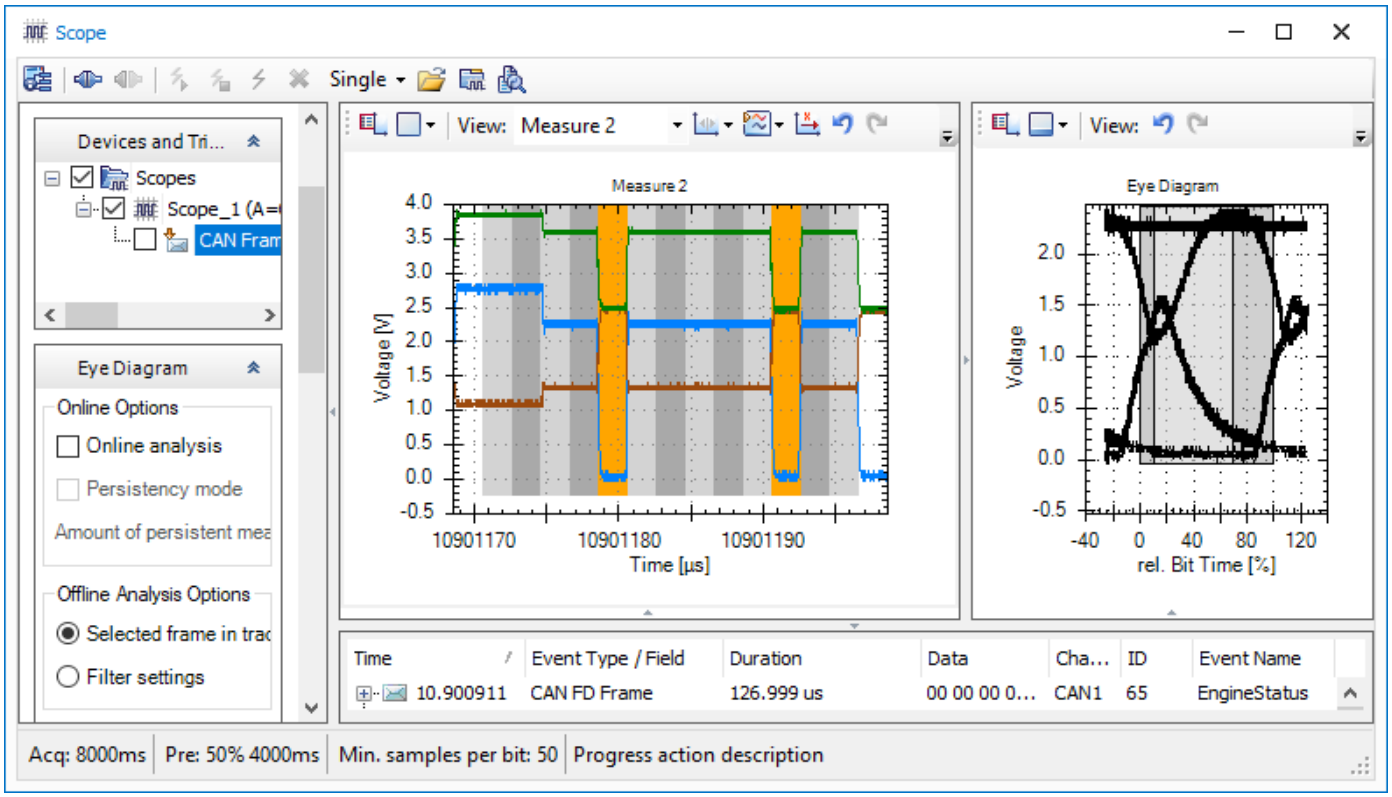



**Figure 5:** Scope Window with eye diagram

### 3.1.5 Data Window

The Data Window is used to display the values of signals, system variables and diagnostic parameters in different types of representation.

- **Show values**
  The data may be displayed as raw or symbolic values. Other display variants are scientific notation and the display of global and local min/max values.
- **Log data**
  Signals can be logged during the measurement and saved to MDF binary format.

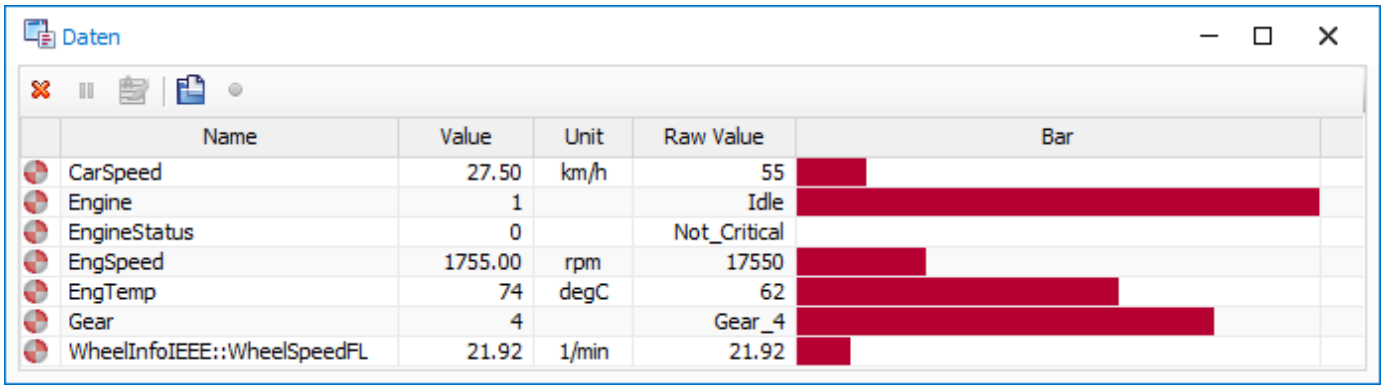



**Figure 6:** Data Window with different representation types for incoming values

### 3.1.6 Statistics Window

The Statistics Window shows statistical information about bus activities (CAN, LIN, FlexRay) during a measurement. This includes such information as bus load on node and frame level, burst counter/duration, counters/rates for frames and errors, and controller states.

- **Show statistical data of individual channels**
  The display of statistical data can be limited to a specific channel, or it can be configured for all available channels.
- **Set updating interval**
  This is used to modify the interval for updating the display.
- **Pause statistics**
  The display of statistical data can be paused during a measurement.

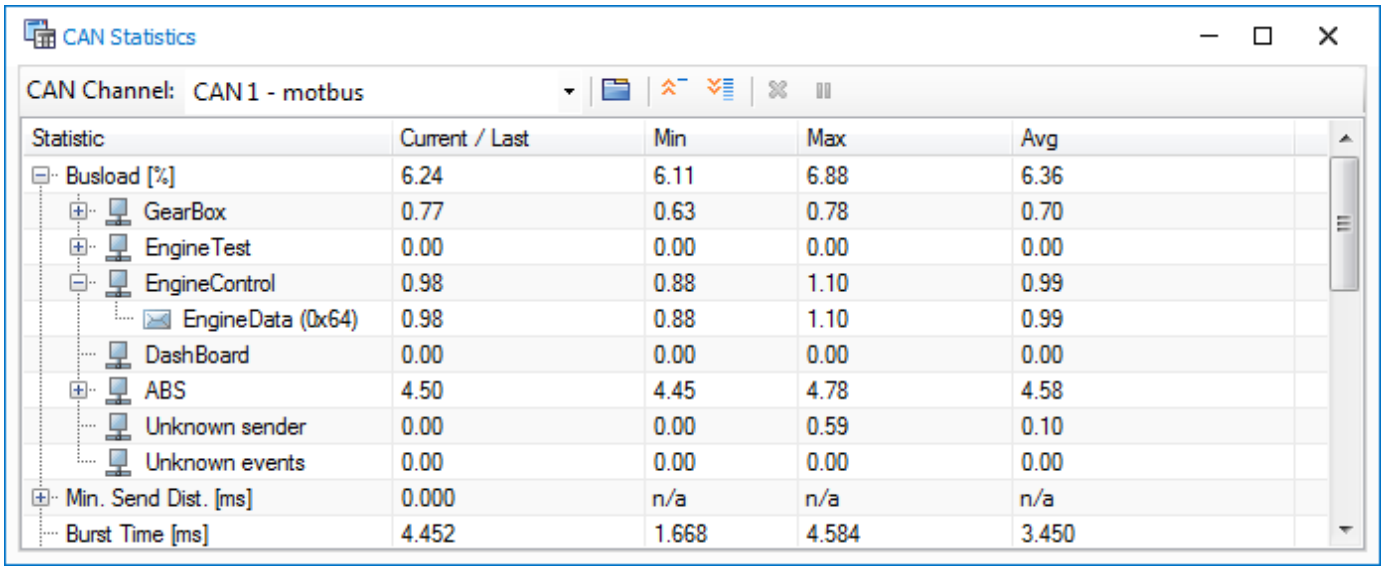



**Figure 7:** CAN Statistics Window with statistical data for one channel (CAN 1)

Certain CAN/LIN/FlexRay statistics can be evaluated in analysis windows such as the Graphics Window, or in program nodes via automatically defined statistical system variables. These system variables are available for each configured network channel and are updated independently of the Statistics Window.

### 3.1.7 Write Window

The Write Window displays system messages and user-specific outputs from CAPL programs.

- **Configure output**
  The Write Window offers different views for filtering system messages according to their source.
- **Log output**
  The Write Window output may be saved to a file or copied to the clipboard as text and be copied to other Windows applications from there.
- **Status display**
  Informs about new unread warning and error messages in the Write Window.

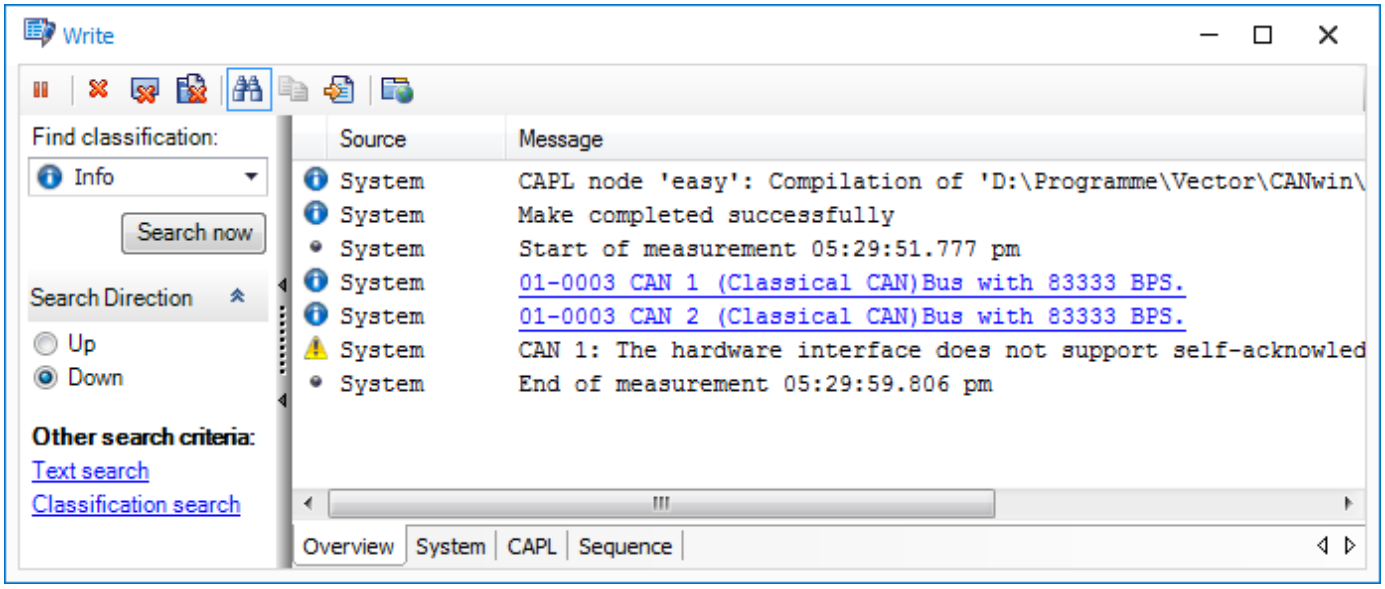



**Figure 8:** Write Window with system messages and CAPL outputs

### 3.1.8 Map Window

The Map Window can be used to integrate GNSS information and maps in CANalyzer. The Map Window is part of the basic feature set of CANalyzer. With the option .Car2x additional Car2x information can be displayed.

- **Display data**
  GNSS data and maps can be displayed in analysis windows.
- **Offline analysis**
  GNSS data can be logged and replayed for later analysis.
- **Window synchronization**
  GNSS data can be synchronized with other analysis windows.
- **Display card**
  The momentary vehicle position and trip route covered are displayed on an electronic map in the Map Window. This enables consideration of geographic aspects in the interpretation of logged measurement data.

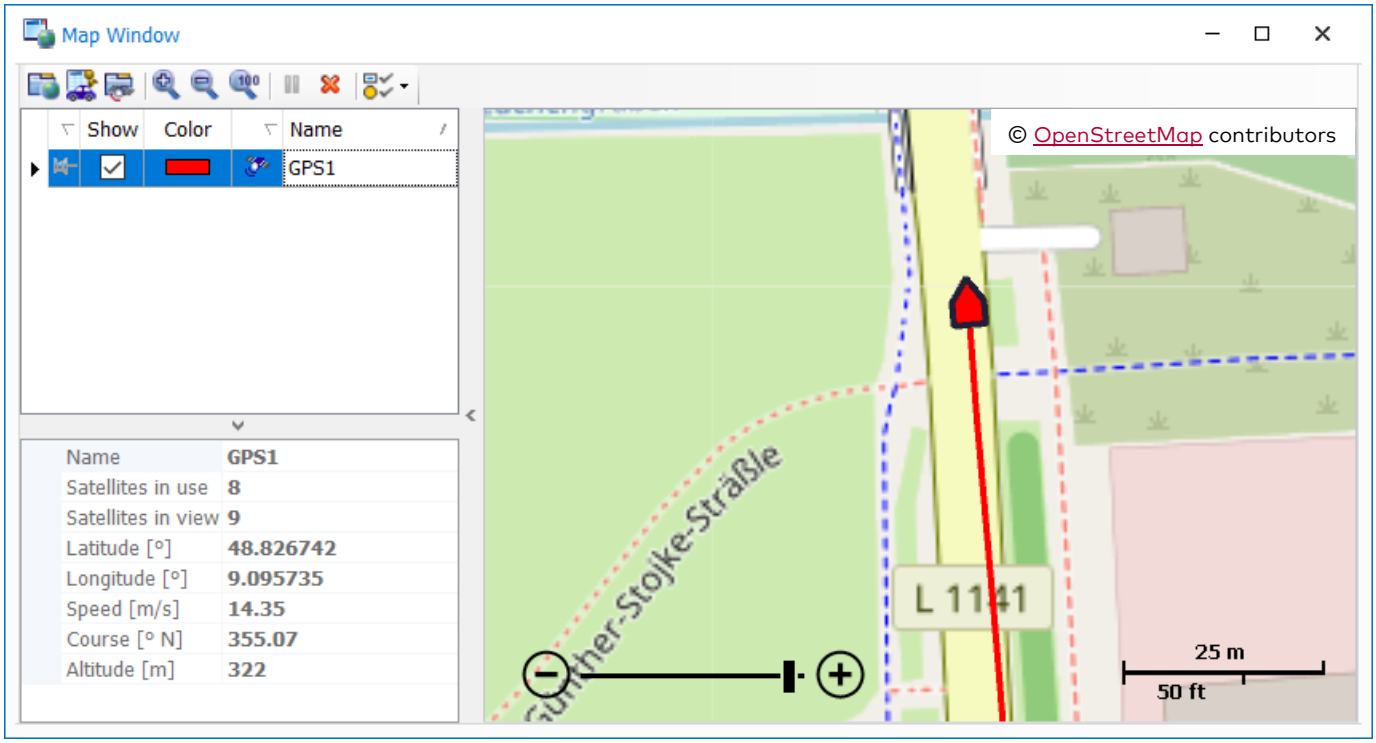



**Figure 9:** Map Window

### 3.1.9 Interactive Generator

The Interactive Generator (IG) can be used to send messages as well as to set the corresponding signal values. This is an easy way to stimulate the bus.

- **Define messages**
  Messages can be defined manually in the send list or with a database. The properties of the messages can be customized.
- **Send messages**
  Messages which are configured in the send list can be sent periodically when a specific screen button is pressed or by pressing a predefined keyboard key.
- **Change signal values**
  In the Interactive Generator, the raw data of each message can be modified in the signal list. For configured signals of a specific message signal waveforms (signal curves) may be defined with the **integrated Signal Generator**. Raw data and signal values can be easily sent with the associated message, e.g. to check the reaction of an ECU.
- **Layer 7 protocols**
  Depending on the bus system the Interactive Generator supports simple layer 7 protocols (e.g. J1939 or GMLAN) and the transmit of multiplexing messages.

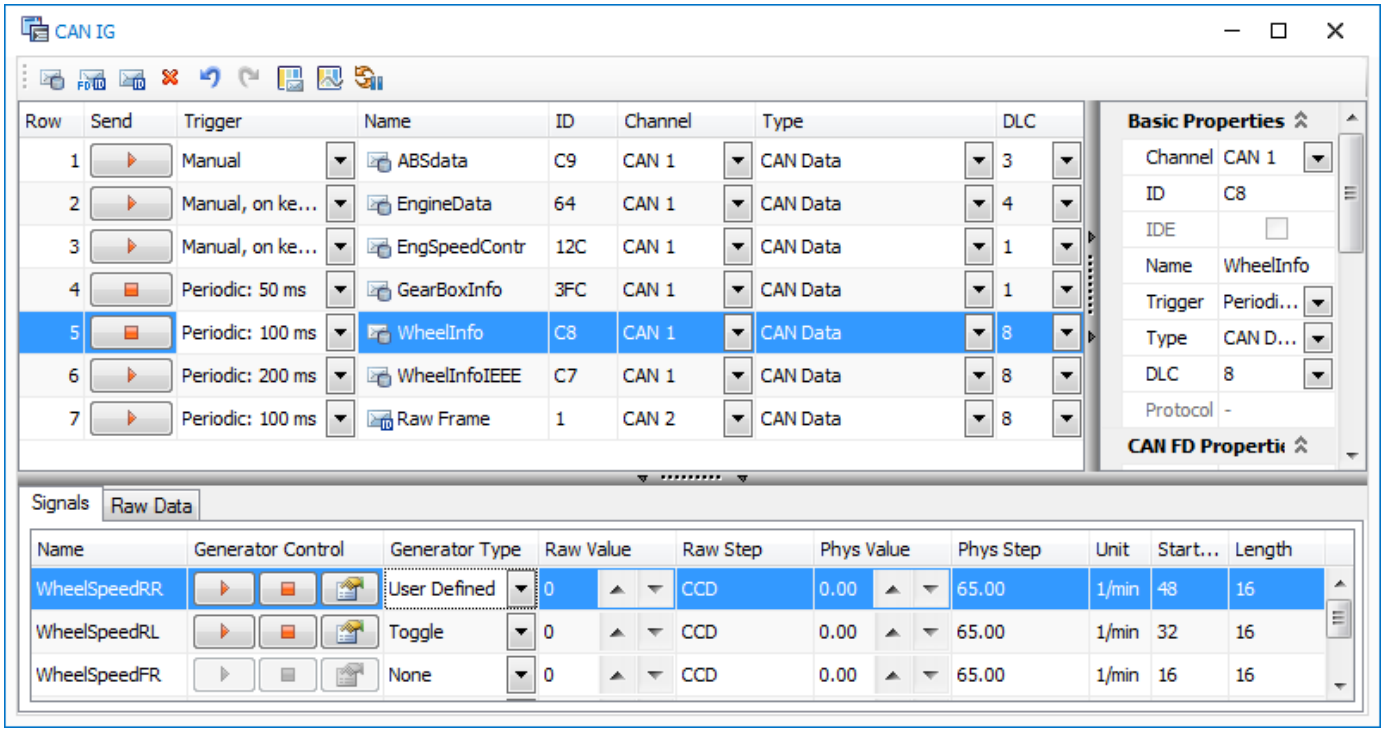



**Figure 10:** Interactive Generator with configured messages and their signals

### 3.1.10 Triggers and Filters

Triggers and filters can react to specific bus events, and they serve to reduce the amount of displayed or logged data. Examples of trigger conditions are error states, messages, signals and signal changes (edges). Complex system states can be triggered by forming groups and linking them with logical operators.

- **Filters in the Measurement Setup**
  Various filters are available in the Measurement Setup that can be used to define which data should be passed to the specific analysis windows and/or which data should be explicitly blocked. All filters can be used as Stop and Pass Filters.
- **Triggers in the Measurement Setup**
  Different trigger conditions can be used in the Measurement Setup to influence the logging of data to a logging file.
- **Filters in the Trace Window**
  In the Trace Window, data can be reduced for analysis both during and after the measurement using various filters. For example, you could set predefined filters to filter for individual signals and signal values or set different column filters.
- **Filters in the hardware**
  The CAN controllers use acceptance filtering to control which received messages are passed to CANalyzer.

### 3.1.11 Logging/Replay

Data can be logged in CANalyzer and replayed later in a post-measurement analysis.

- **Replay**
  The Replay Block can be used to replay measurement sequences that have been logged in a logging file. The messages contained in the logging file are introduced into the data flow.
- **Logging**
  The Logging block can log the bus traffic in the BLF and ASCII formats. The logged data can then be replayed in offline mode or with a Replay Block.

## 3.2 Data Interpretation

An observer for the transport protocol per ISO/DIS 15765-2 is included. This observer interprets transport protocol messages that are sent over the CAN bus and displays the results in the Trace Window in clear text. Diagnostic interpretation is also possible per KWP2000, UDS and via OEM-specific CANdela Studio description files (CDD), ODX/PDX and MDX files. The Basic Diagnostic descriptions defined in CANalyzer can be used for diagnostic interpretation as well. Along with the interpretation in the Trace Window, individual diagnostic parameters can also be displayed in the Graphics and Data Windows.

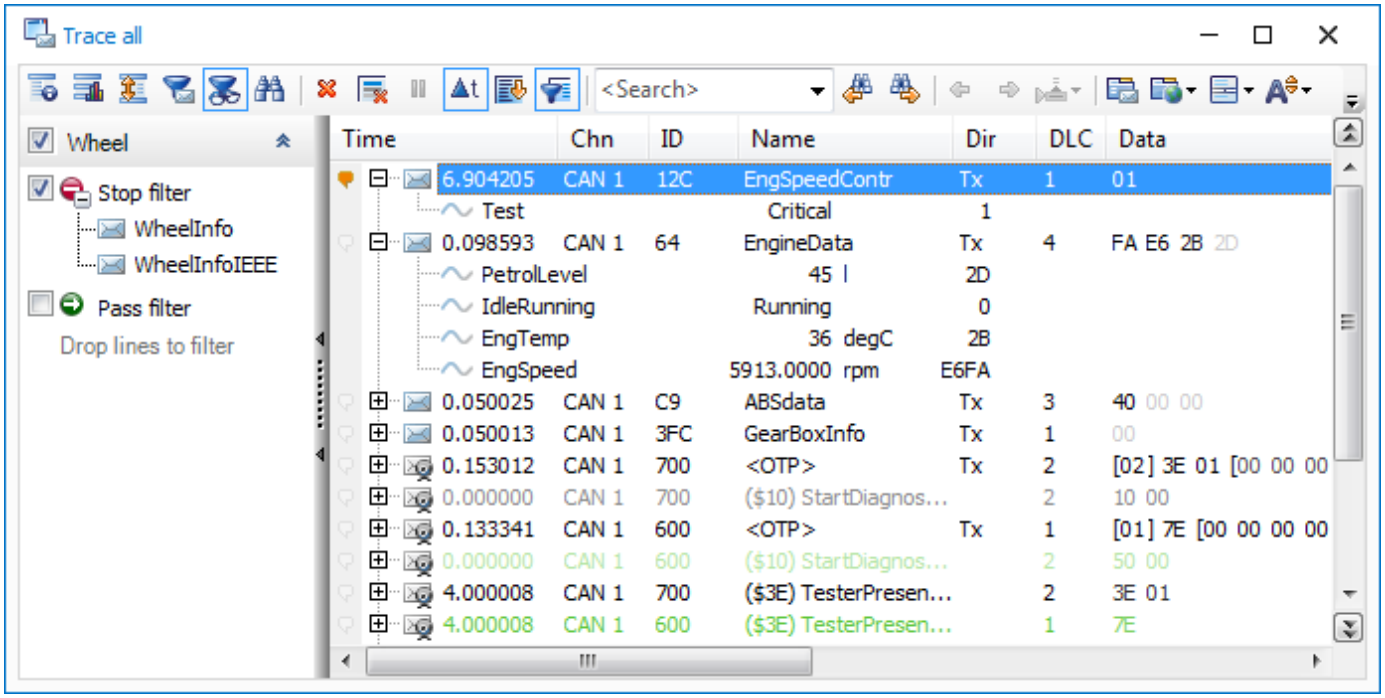



**Figure 11:** Trace Window with analysis filters and diagnostic interpretation

## 3.3 Offline Evaluation

The logged message traffic can be evaluated at a later point in time, in offline mode, using all CANalyzer functions. This makes it possible, for example, to conveniently analyze extensive recordings of a measurement drive afterwards in the workplace. Collections of logging files can also be specified, which are then chronologically replayed into the analysis based on their time stamps. In addition, logging files can be directly imported into the Trace Window for quick analyses.

Offline mode directly supports the logging format of the GL1000/GL2000/GL3000/GL4000 logger families and of CANlog from Vector. This eliminates conversion steps before analysis.

## 3.4 Export Functions

Export functions for logging and the Trace and Graphics Windows enables conversion of logged files or window contents to other file formats, e.g. to *.csv. In export of the logging files, individual signal information is extracted from the logged, message based data.

## 3.5 Integrated Desktops

CANalyzer users often have many analysis windows opened. That is why CANalyzer offers an integrated desktop concept, in which users can switch between any number of virtual desktops on tabbed pages. Opened windows can be distributed to desktops, and information can be sorted by work processes or subject matter.

## 3.6 Set Start Values

In the Start Values Window, values that are set on measurement start can be preassigned for system variables.

The list of start values can be exported to a file and loaded from a file. Thus, for example, simulation parameters can be conveniently assigned using various sets of start values.

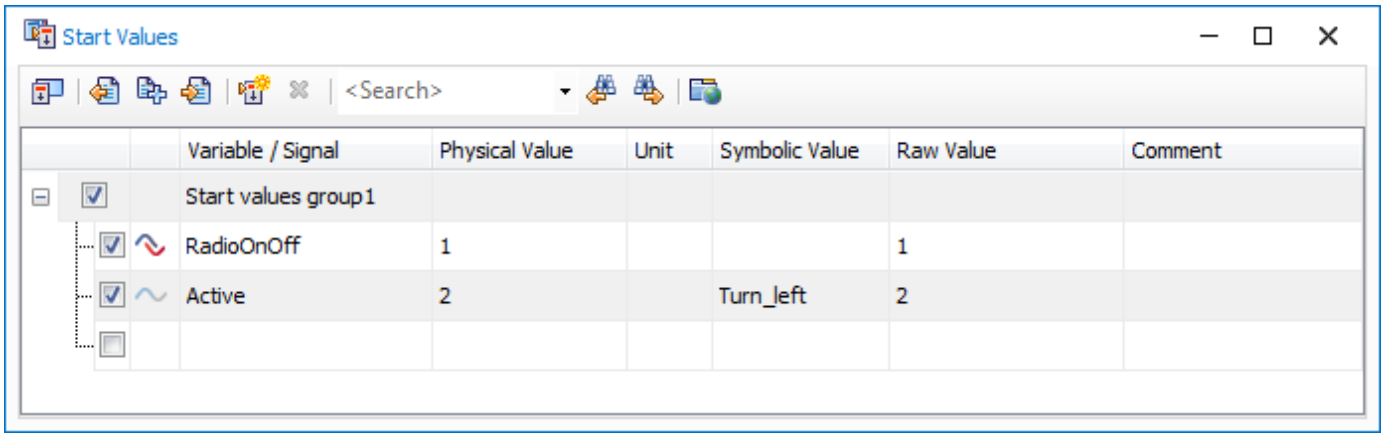



**Figure 12:** Start Value Window

## 3.7 Symbol Mapping

Using the Symbol Mapping dialog, system variables can be mapped.

When the value of a source variable changes during measurement, the value of the destination variable is automatically set. Optionally, you can apply a linear conversion formula.

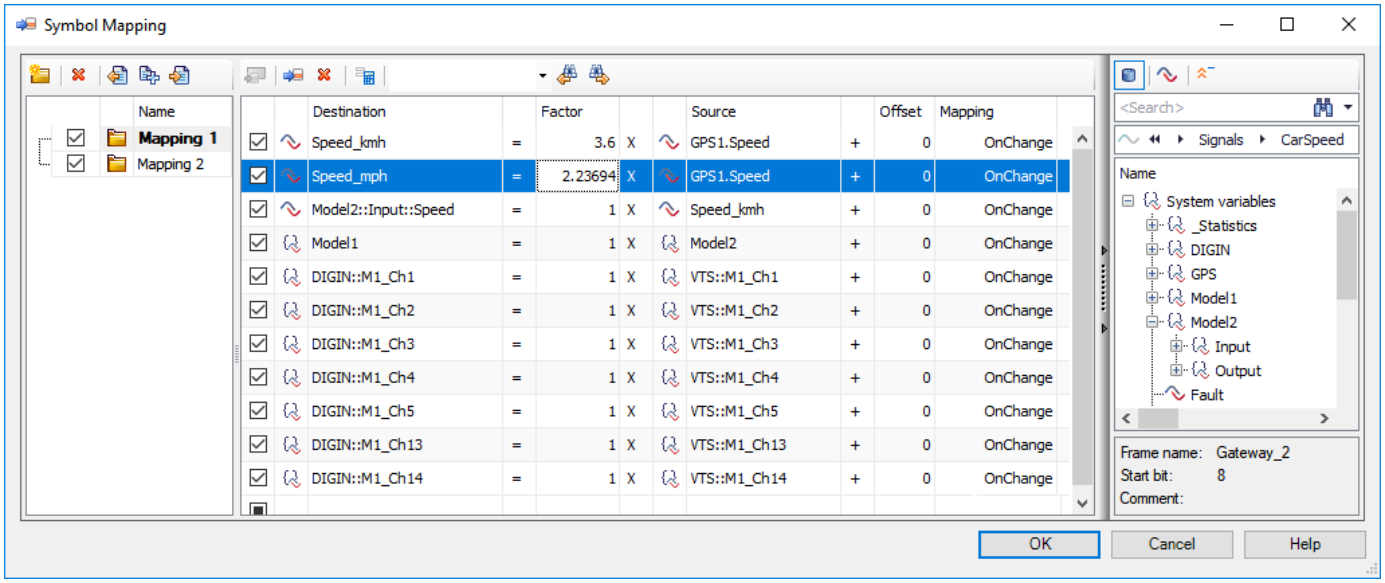



**Figure 13:** Mapping dialog

## 4 Diagnostics

CANalyzer is used essentially to define and send diagnostic services for KWP2000 or UDS. In addition, diagnostic parameters can be defined and local identifiers (LID) allocated. This makes it possible to read information from the ECU via diagnostics quickly and easily.

The following concepts and functions are available, for example:

- Support of all important diagnostic description formats for KWP2000 and UDS (ISO 14229):
  - ODX 2.0.1/2.2.0 as PDX files
  - MDX 2.0/3.0/4.0
  - CANdelaStudio (CDD)
- Option of modifying key diagnostic communication parameters of the diagnostic description (transport and diagnostic layer) in the Diagnostic/ISO-TP Configuration dialog
- Basic Diagnostic Editor for quickly defining simple diagnostic services, if no diagnostic description is available (for CAN, LIN, FlexRay, Ethernet and K-Line)
- Interactive diagnostic tester with Diagnostic Console, Fault Memory Window and Diagnostic Session Control with configurable Security DLL
- Interactive Diagnostic Parameters Window for cyclic or manual query and display of diagnostic parameters from diagnostic responses
- Preconfigured OBD-II tester with related Diagnostic Console and Fault Memory Window
- Support of several addressing schemes (e.g. normal, extended, normal fixed and mixed) and addressing types (functional/physical)
- Analysis of diagnostic communications on the service and parameter levels (i.e. symbolic representation based on the diagnostic description) in the Trace, Data and Graphics Windows
- Display of protocol errors in the Trace Window
- Optional use of panels to display diagnostic parameters and stimulation of ECUs via diagnostic requests
- Support of all important network types in the automotive industry (CAN, LIN, FlexRay, Ethernet and K-Line)
- Support of DoIP (Diagnostics over IP), HSFZ (High Speed Fahrzeugzugang) and DoSoAd (Diagnostics over AUTOSAR Socket Adaptor)
- Logging and replay of diagnostic sequences via macros

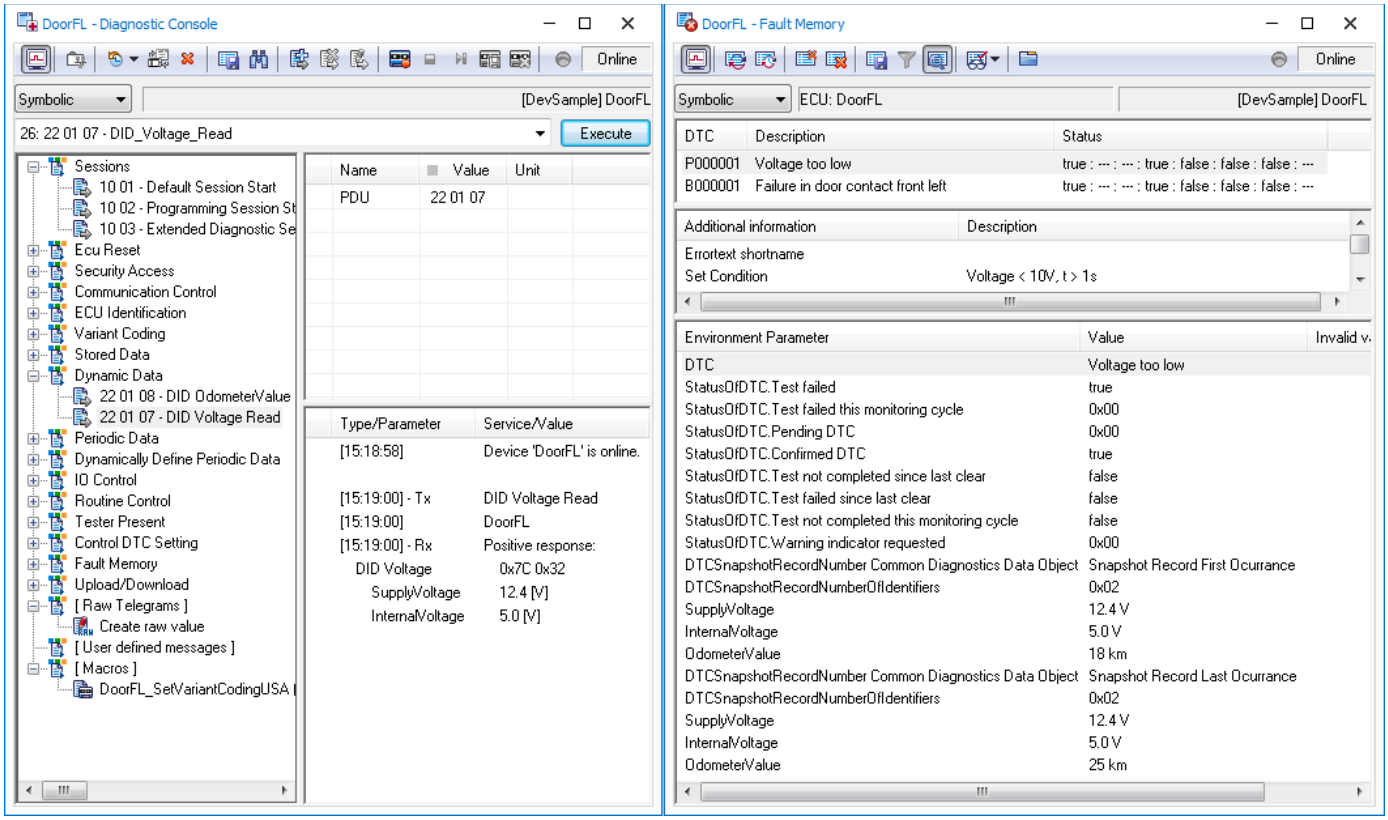



**Figure 14:** Diagnostic Console and Fault Memory Window

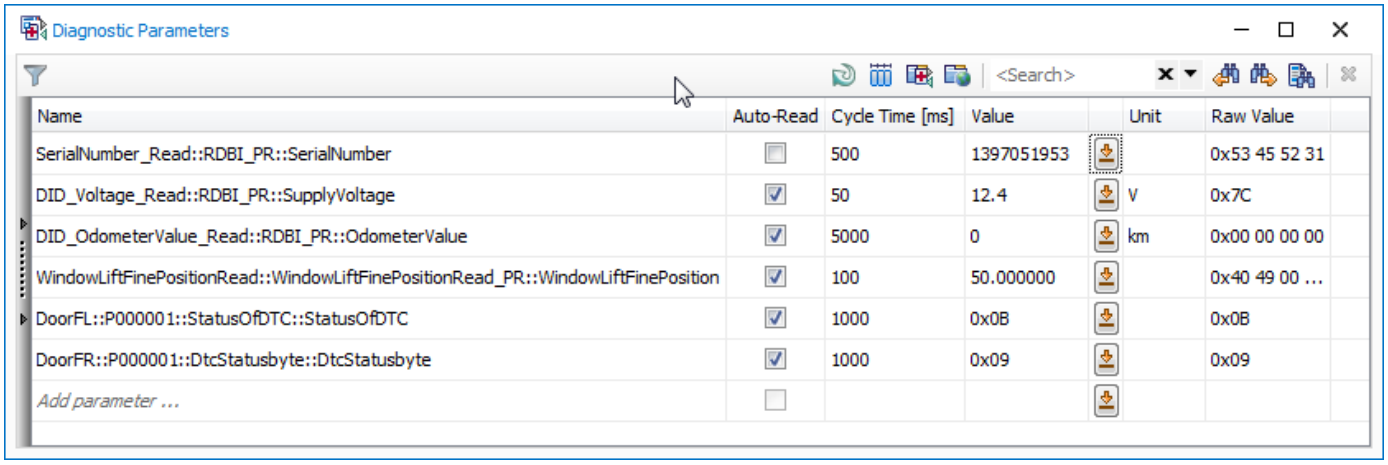



**Figure 15:** Diagnostic Paramters Window

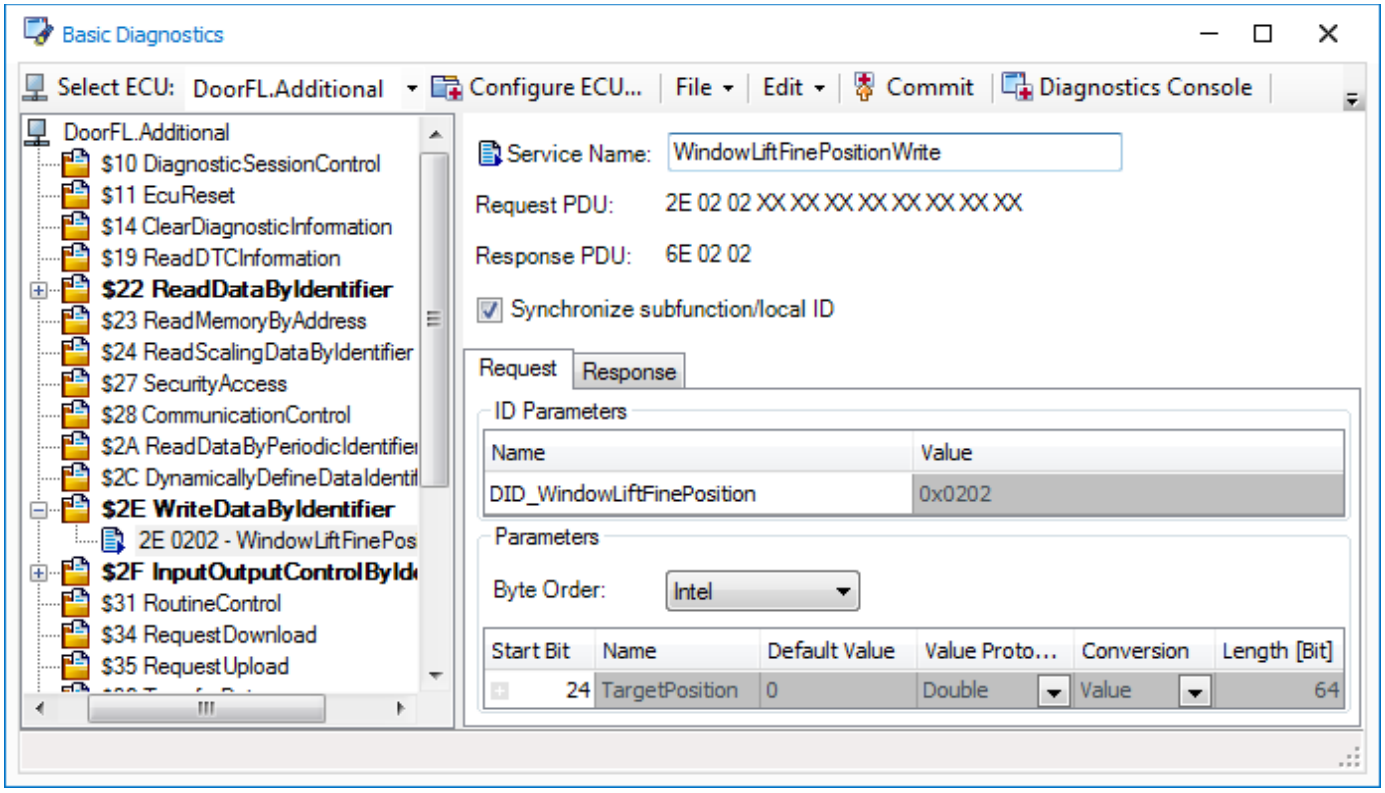



**Figure 16:** Basic Diagnostic Editor

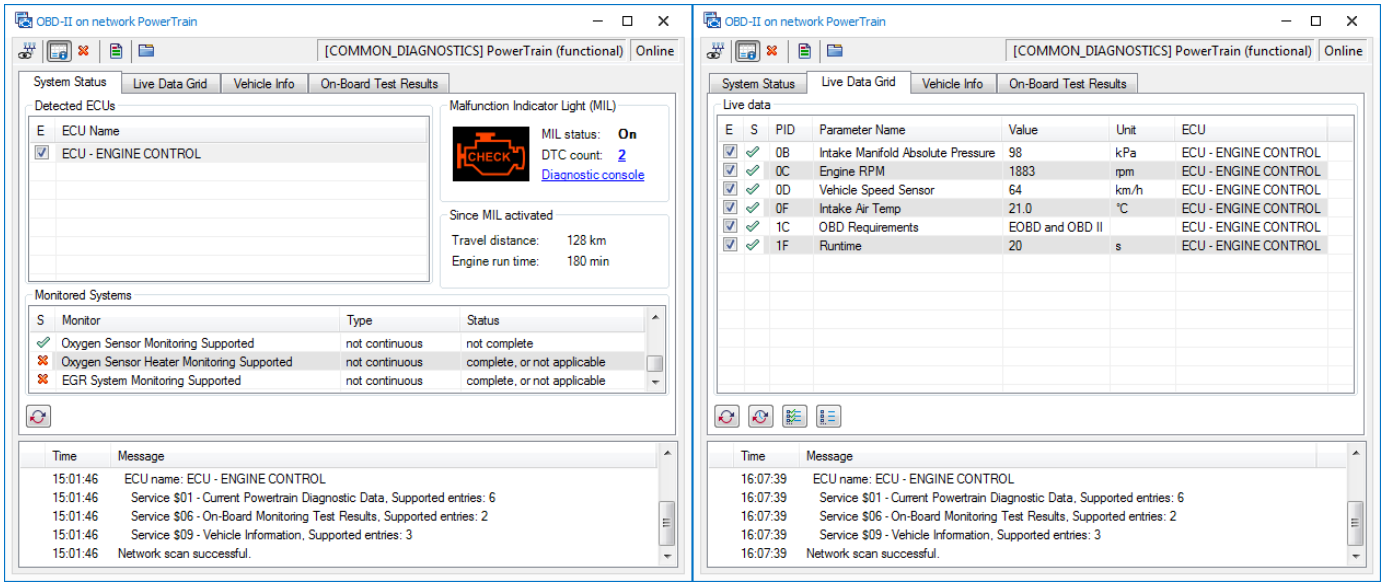



**Figure 17:** OBD-II Window

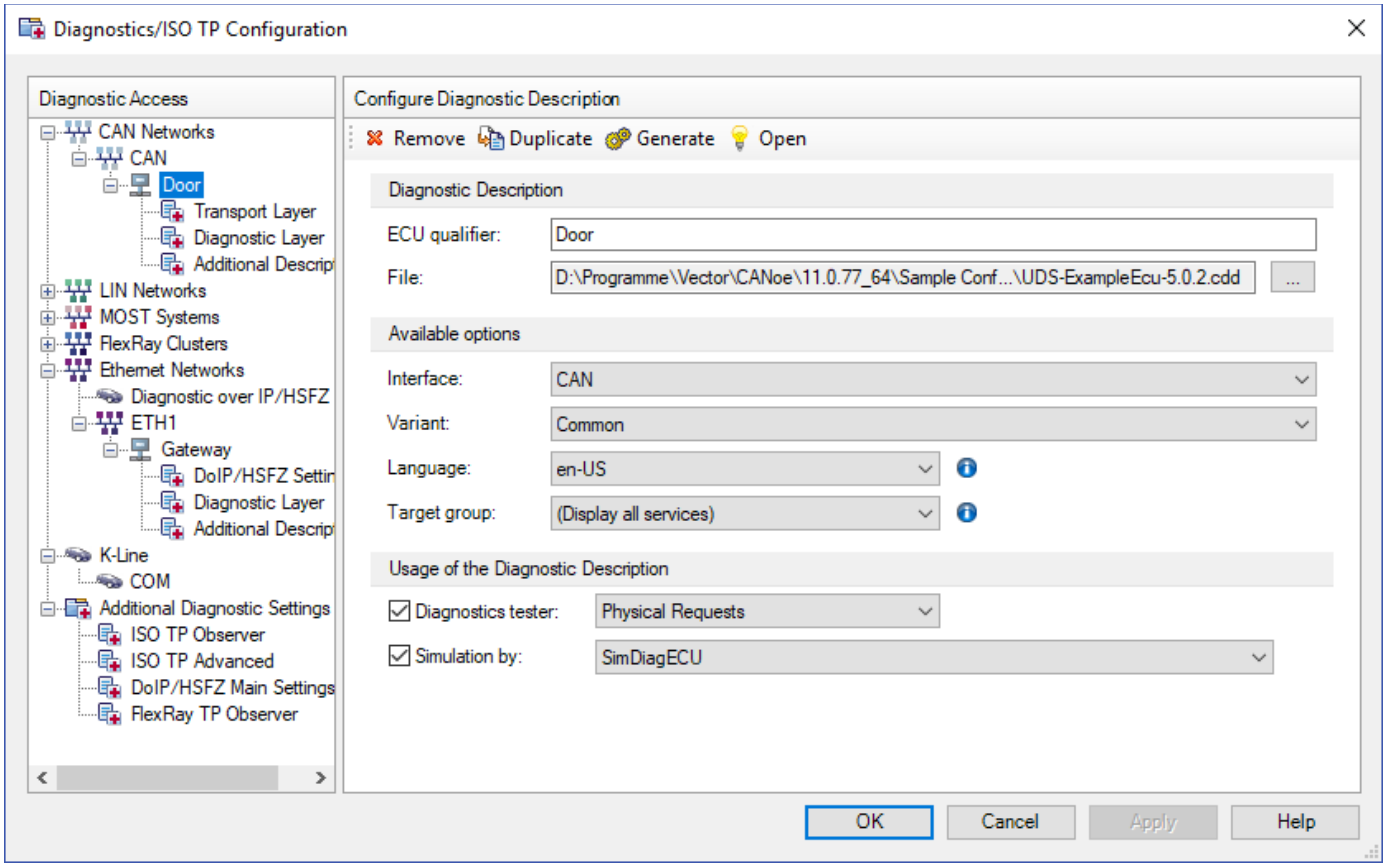



**Figure 18:** Diagnostics/ISO-TP Configuration dialog

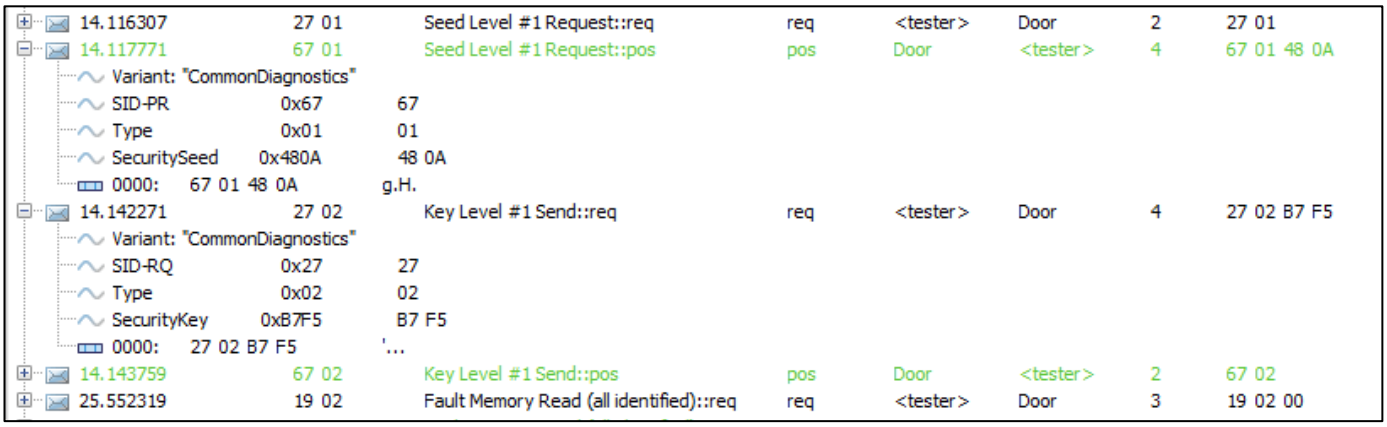



**Figure 19:** Representation of diagnostic communication in the Trace Window

## 4.1 Further Information

The application note **AN-IND-1-001_CANoe_CANalyzer_as_Diagnostic_Tools** describes a general introduction to working with diagnostics in CANoe/CANalyzer. Fundamental technical aspects and options are presented with the Diagnostic Feature Set. This document supplements CANalyzer's help and can be used as a tutorial.

# 5 Programming

## 5.1 CAPL Interface

The CAPL (Communication Access Programming Language) programming language extends the functional scope of CANalyzer tremendously. Special characteristics of CAPL include:

- Can be learned quickly since it is based on the C programming language
- Fully event-controlled in its operation. CANalyzer takes over control from the user.
- Supports symbolic access to all database information such as messages and signals. Signal values can be used directly in their physical form.
- The language has been extended with special functions for quick implementation of problem solutions in various use scenarios
- Flexible extension by external libraries

### 5.1.1 C-Like Syntax

The usual scalar data types and arrays are provided (1, 2, 4 and 8 byte long whole number types as well as an 8 byte long floating point type). Assignments, arithmetic operators and loop flow control conform to C-syntax.

```
myFunction {
  int counter;
  for ( counter = 0; counter < 8; counter++ ) {
    doSomethingWithCounter ( counter );
  }
}
```

### 5.1.2 Event-Oriented Control

CAPL is an event-controlled programming language. In contrast to C, special predefined event handlers (event procedures) are available in CAPL, which are always executed whenever a specific event occurs (if time controlled then triggered by the hardware or internal to CANalyzer).

Here are just a few examples of these event handlers:

| Event Handler | Event |
|---|---|
| On timer seconds cycle | Time controlled |
| On message ESPStatus | Message input or output |
| On signal update | Rewrites signal value |
| On sysvar | Modifies system variable |
| On diagRequest | Diagnostic request |
| On FRError | Detects FlexRay bus errors |

### 5.1.3 Symbolic Access

Signal values are generally accessed as physical values, regardless of the scaling of message transmission. This is set in the database and is taken from there.

> Access on a message base:

```
message EnergyMgmt msg;
// Most significant bytes Motorola, with 12bit only the lower 4 bits are used
msg.byte(0) = (msg.byte(0) & 0xF0) | (byte)((14.1 - 8) / (18 - 8) * 4096 / 256) & 0xF;
// Least significant byte
msg.byte(1) = (byte)((14.1 - 8) / (18 - 8) * 4096) & 0xFF;
output(msg);
```

### 5.1.4 Application-Specific Language Extensions

For all use cases of CANalyzer there are numerous functions that are specially tailored to everyday problems related to these topics.

> **Analysis**
CAPL can be used in the analysis of measurement results – either online and offline. One simple task might be to count the occurrences of a specific event or perform computations with the contents of certain signals.

```
On message Brake {
  long TempCounter = 0;
  $BRECounter++;
  // Weighing the average
  TempCounter = $BRECounter;
  if ($BRECounter > 1000)
    TempCounter = 1000;
  @AveragePressure = @AveragePressure * TempCounter + $Brake::Pressure;
  @AveragePressure = @AveragePressure / (TempCounter + 1);
  output ( this );
}
```

> **Stimulation**
You can also use CAPL to generate messages to stimulate ECUs. This relieves the developer of routine work tasks. Signals and messages of the bus are defined in the database (e.g. in DBC, LDF or FIBEX files). These files are often managed, maintained and updated centrally.

### 5.1.5 CAPL Browser

The functionality of the CAPL Browser goes beyond that of an editor for CAPL programs. It offers functions of an advanced development environment, such as:

- Code auto-completion and syntax checking while writing code
- Configurable syntax highlighting
- Syntax-sensitive tabs
- Folding function blocks and functional references in a tree view for quicker navigation
- Find and replace in individual or multiple files
- Online help with references to functions
- Calling of the compiler with preselected source text lines in case of error
- Hierarchical function list with search function for direct copying into the source text

Objects of the CANalyzer database are available in the CAPL Browser as well, and they are also displayed in a tree view. The following database contents can be accessed from what is known as the **Symbol Explorer**:

- Network symbols such as nodes, messages and signals
- System variables that are used CANalyzer-wide
- All diagnostic symbols such as requests, responses and fault memory

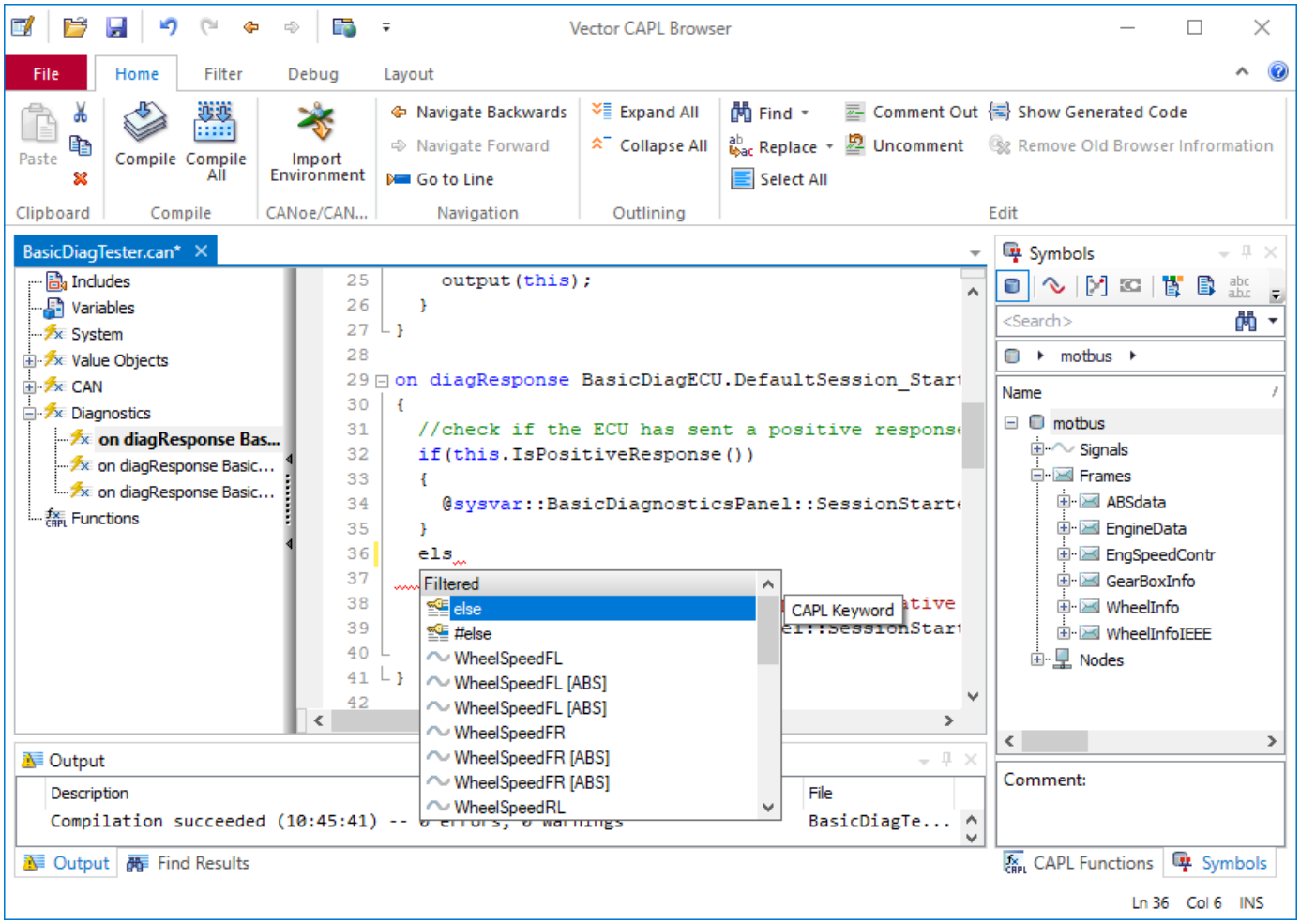



**Figure 20:** CAPL Browser with opened CAPL program, contained event procedures and network symbols from the database

## 5.2 Visual Sequencer

This is a quick way to graphically configure flow sequences without requiring programming. Variables and signals may be set within such sequences. Frames and diagnostic commands can also be sent. In addition, it is possible to wait for certain events, check values or define repetitions with control structures (repeat...until). These sequences are therefore ideal for simple tests of heterogeneous systems or for stimulating ECUs.

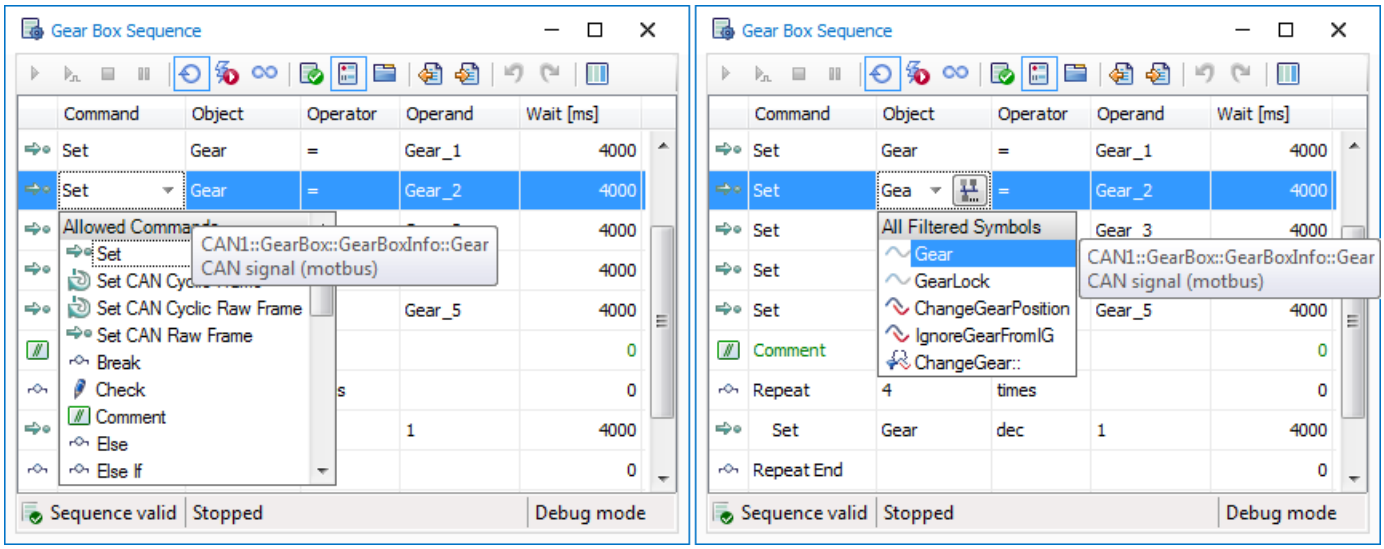



**Figure 21:** CANalyzer Visual Sequencer for creating test and stimulation sequences. Makes it easy to select commands and database objects with auto-complete support and to display detailed database information.

# 6 Panels

Panels are graphical elements that can be used to modify signal and variable values and display them with controls such as sliders or pointer instruments. Panels are used to display the analysis data from CAPL programs, for example, or to control CAPL programs and transfer values into the programs.

The **Panel Designer** can be used to conveniently create such panels. For example, it is easy to link a symbol to a control by drag and drop. The individual panels and controls are configured via the constantly open Properties Window, and a whole series of useful alignment functions ensure an optimal layout.

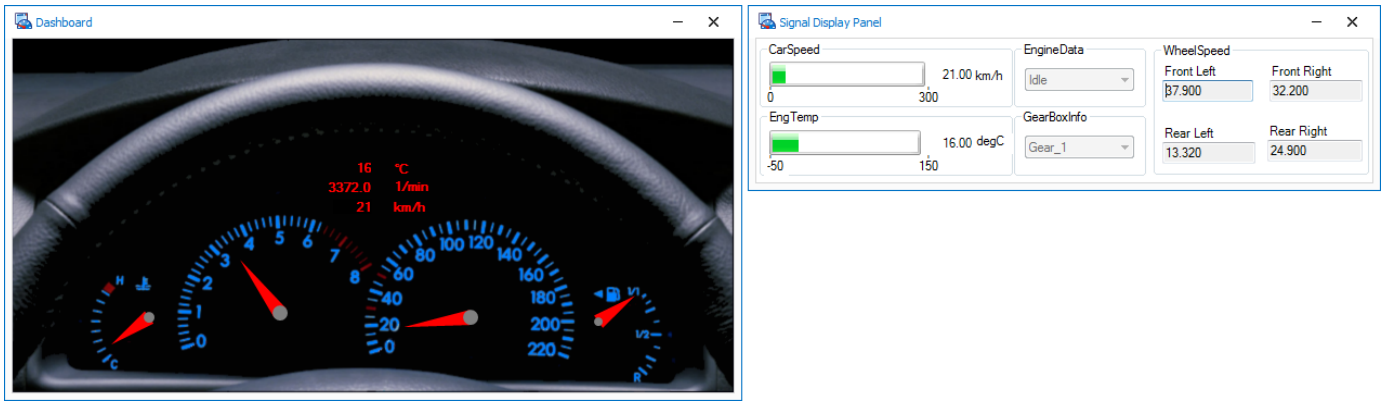



**Figure 22:** User-defined panels for displaying signal and variable values

# 7 Software Interfaces

The integrated COM Server (Component Object Model) enables control of the measurement sequence by external applications and convenient data exchange with standard software, e.g. for measurement data analysis or in-depth evaluation of the observed bus traffic. Frequently used programming/script languages here are Visual Basic or Visual Basic for Applications. C++/C# are also frequently used. The functionality that CANalyzer offers over the COM interface covers such aspects as:

- Control of the simulation, starting and stopping the measurement
- Loading existing configurations, generating new configurations, adding databases and blocks to the Tx branch
- Access to signals and system variables, access to CAPL functions, compiling of CAPL nodes

Visual Basic Script example for starting the measurement:

```
set app = createobject( "canalyzer.application")
set measurement = app.measurement
measurement.start
set app = nothing
```

Visual Basic Script example for opening a configuration:

```
set app = createobject( "canalyzer.application")
app.open "D:\PathToMyConfig\myconfig.cfg"
set app = nothing
```

## 7.1 Further Information

A general introduction to COM Server functionality of CANoe/CANalyzer is described in application note **AN-AND-1-117_CANalyzer_CANoe_as_a_COM_server**. Fundamental technical considerations and options are presented, and they are illustrated as Microsoft Visual Basic examples.

# 8 Hardware Interfaces

CANalyzer supports all bus system interfaces available from Vector. Optimal bus access is possible for every use case thanks to a large selection of different computer interfaces (PCMCIA, USB 2.0, PCI, PCI-Express, PXI) and bus transceivers.

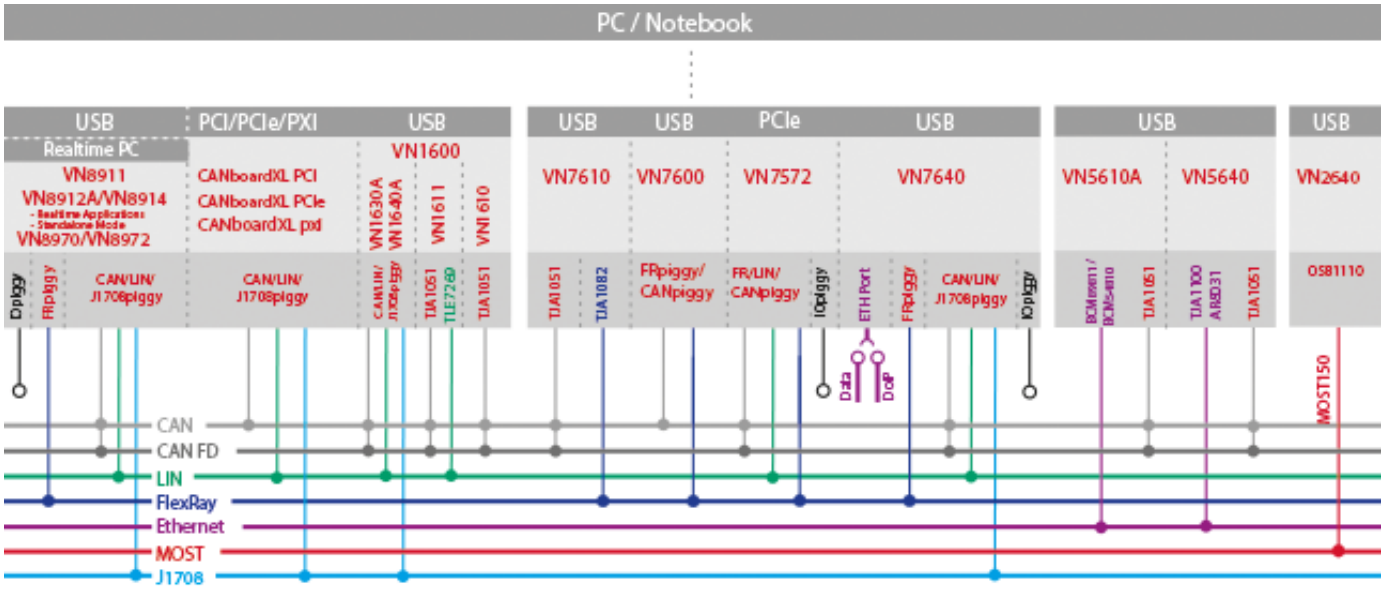



**Figure 23:** Overview of Vector hardware

# 9 Option .Scope

The option .Scope is an integrated oscilloscope solution for CANalyzer based on a very powerful USB oscilloscope hardware. This CANalyzer option appears as a further analysis window with views for configuration, bus level and protocol decoding. The supported hardware has up to 4 input channels for 2 CAN/CAN FD/FlexRay or 4 LIN/IO and is triggered by the sync line of Vector interface hardware (e.g. VN1630/40, VN8900, VT System). The option .Scope is available for all CANalyzer variants.

## 9.1 Application Areas

The powerful combination of the USB oscilloscope and CANalyzer offers many new ways to analyze protocol errors. With bus-specific trigger conditions and CANalyzer time synchronization, the causes of protocol errors can be found significantly faster than with any traditional oscilloscope.

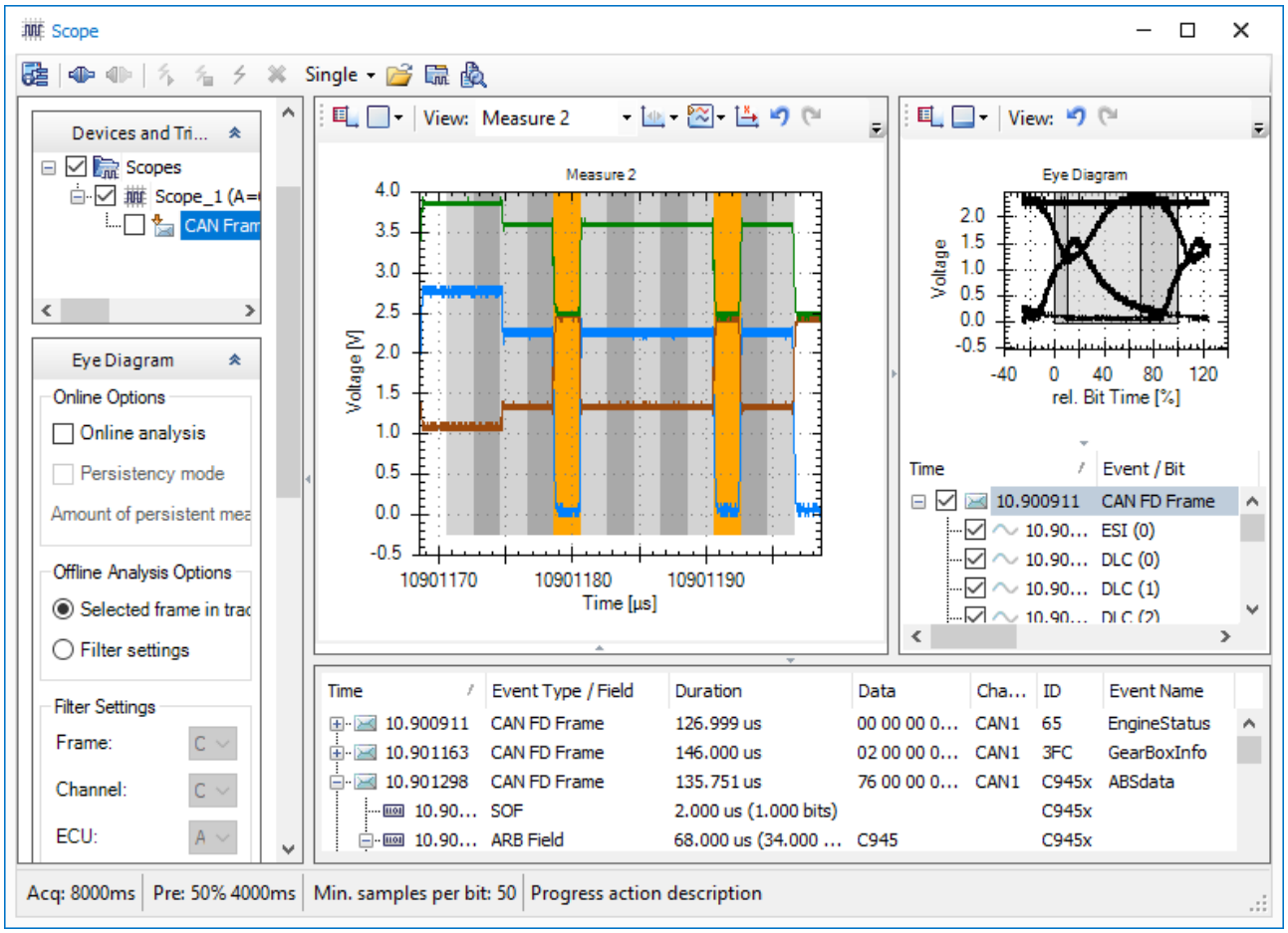



**Figure 24:** Detailed analysis of a CAN FD frame/error on physical and logical level with the Scope Window

## 9.2 Overview of Advantages

- Extremely compact and portable oscilloscope solution based on an USB oscilloscope hardware
- Easy bus connection via Vector Scope Bus Probe
- Oscilloscope functionality specially developed for bus analysis, e.g.:
  - Predefined trigger conditions for bus errors and events
  - Synchronous recording of bus and I/O signals with the CANalyzer time base
  - Complete decode of protocol errors (e.g. CAN Error Frames)

## 9.3 Supported Protocols

- Bus systems: CAN, CAN FD, FlexRay und LIN

## 9.4 Supported Oscilloscope Hardware

- Vector PicoScope 5444B-034:
  - USB precision oscilloscope with 200 MHz bandwidth and 500 MS/s sampling rate
  - 4 input channels for bus signals (2 CAN/CAN FD/FlexRay or 4 LIN/IO)
- Vector PicoScope 5242B-034:
  - USB precision oscilloscope with 60 MHz bandwidth and 500 MS/s sampling rate
  - 2 input channels for bus signals (1 CAN/CAN FD/FlexRay or 2 LIN/IO)
- Bus connection via Vector Scope Bus Probe with D-Sub connector
- Vector Scope Y-Trigger Cable for internal and external triggering via the sync line of a Vector interface

## 9.5 Oscilloscope Software

Option .Scope appears as a new analysis window in CANalyzer with views for configuration/measurements, bus levels and protocol decoding.

### 9.5.1 Configuration Functions

- Connection of several bus signals to input channels of oscilloscope. For each bus system a default configuration for scope hardware is created
- Easy configuration of sampling rate (independent of bus baud rate) in min. sampling points per bit
- Automatic adjustment of acquisition time depending on the bus baud rate
- Pre-/post-trigger time adjustable from 10% to 90%
- Support of up to multiple scope devices of the same type

### 9.5.2 Trigger Functions

- **Single** and **Repeated** trigger modes
- Manual triggering via the toolbar or CAPL function
- Simple trigger conditions:
  - Trigger on frames (ID or ID range)
  - Trigger on protocol errors (e.g. CAN Error Frames)
  - All conditions may be combined in a logical OR relationship
  - Triggering on external signals with edge or pulse triggers (IO trigger)
- Advanced trigger conditions programmable by CAPL:
  - Trigger on change in a frame signal or in a system variable (e.g. I/O)
  - Complex trigger conditions may be defined as AND/OR relationships
  - Establishment of trigger conditions via the COM interface of CAPL

### 9.5.3 Analysis Functions

- Detailed decoding of the bus levels on the bit level, even in case of protocol errors
- Display of the time stamp and the voltage value per sampling value
- Bidirectional synchronization of Trace view (logical values of the data link layer) and the diagram (physical values). The diagram shows the signal encoding of the Trace view.
- Time-Synchronization with other CANalyzer windows, e.g. Trace, Graphics and State Tracker

### 9.5.4 Offline Functions

Offline functions of the Scope Window are available to users even without the license for option .Scope, e.g. to view and analyze measurements made by colleagues.

- Overview and easy management of scope measurements
- Automatic display of new measurement data
- Eye diagram analysis with single bitmask accessible via user interface
- Complete export and import of scope measurements (binary format *.csf)
- Export of scope measurement data in ASCII (*.csv) or MATLAB formats (*.mat)
- Export/Import of eye diagrams (*.ceye).
- Comparison mode for scope signals and for entire scope measurements
- Marking significant points of a measurement with global markers

## 10 Training

As part of our training program, we offer various classes and workshops on CANalyzer in our classrooms at Vector and on-site at our customers.

You will find more information on individual training courses and a schedule **online**.

VECTOR

## Get More Information

**Visit our website for:**

- News
- Products
- Demo software
- Support
- Training classes
- Addresses

**www.vector.com**