CANoe .A429

Product Information
Table of Contents

1 Introduction to CANoe................................................................................................................................................. 4
1.1 Bus Systems and Protocols................................................................................................................................. 4
1.2 Product Concept and Variants.............................................................................................................................. 6
1.3 Product Components.............................................................................................................................................. 6
1.4 System Requirements............................................................................................................................................. 6
1.5 Further Information............................................................................................................................................... 6
2 Functions .................................................................................................................................................................... 7
2.1 Database Support.................................................................................................................................................. 7
2.2 Special Functions.................................................................................................................................................. 7
3 Analysis.................................................................................................................................................................... 8
3.1 Measurement Setup............................................................................................................................................. 8
3.2 Trace Window ..................................................................................................................................................... 9
3.3 Graphics Window ................................................................................................................................................ 10
3.4 Data Window....................................................................................................................................................... 11
3.5 Bus Statistics Window ....................................................................................................................................... 11
3.6 Write Window .................................................................................................................................................... 12
3.7 Triggers and Filters ........................................................................................................................................... 12
3.8 Logging/Replay .................................................................................................................................................. 12
4 Stimulation/Simulation.............................................................................................................................................. 13
4.1 Variables and Generators................................................................................................................................. 13
4.1.1 Interactive Generator ................................................................................................................................. 13
4.1.2 Signal Generator ....................................................................................................................................... 14
4.2 MATLAB/Simulink ......................................................................................................................................... 15
4.2.1 Further Information ................................................................................................................................. 15
5 Test ....................................................................................................................................................................... 15
5.1 Testing ECUs and Networks............................................................................................................................... 15
5.2 Further Information........................................................................................................................................... 16
6 Programming.......................................................................................................................................................... 17
6.1 CAPL Interface ................................................................................................................................................ 17
6.1.1 C-Like Syntax ........................................................................................................................................... 17
6.1.2 Event-oriented Control ............................................................................................................................. 17
6.1.3 Symbolic Access ....................................................................................................................................... 18
6.1.4 Application-Specific Language Extensions............................................................................................... 18
6.2 CAPL Browser ................................................................................................................................................. 20
6.3 Debugging ........................................................................................................................................................ 21
7 Panels .................................................................................................................................................................. 22
8 Hardware Interfaces................................................................................................................................................ 23
9 Interfaces to Other Applications............................................................................................................................ 24
9.1 COM Interface ................................................................................................................................................. 24
9.1.1 Further Information ............................................................................................................................... 24
9.2 FDX .................................................................................................................................................................. 24
9.3 ASAM XIL API ................................................................................................................................................ 24
10 Training .............................................................................................................................................................. 24
This document presents the CANoe .A429 application areas of analysis, stimulation/simulation, testing and their individual functions. The document also contains a brief overview of programming in CANoe .A429, supplemental programs as well as hardware and software interfaces.

Product information and technical data on CANoe in general are available in a separate document.
1 Introduction to CANoe

CANoe is a versatile tool for the development, testing and analysis of entire ECU networks as well as individual ECUs. Specific problems can be detected and corrected early in the development process. Graphic and text based evaluation windows are provided for evaluating the results.

CANoe contains the Test Feature Set for easy and automated test execution. It is used to model and execute sequential test sequences and automatically generate a test report.

![CANoe configuration for analysis of a ARINC 429 system](image)

**Figure 1:** CANoe configuration for analysis of an ARINC 429 system

1.1 Bus Systems and Protocols

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

CANoe supports the following bus systems: CAN, CAN FD, LIN, MOST, FlexRay, J1708, Ethernet, WLAN, ARINC 429 and AFDX.\(^1\)

\(^1\) AFDX® is a registered trademark of Airbus
Option CAN is the basis for these supported CAN-based protocols: J1939, CANopen, ISO 11783, GMLAN, CANaerospace, ARINC 825.
1.2 Product Concept and Variants

CANoe is available in the following variants:

- With full range of functionality (full)
- As a Runtime (run) variant with unmodifiable simulations, full range of analysis functions and easy activation/deactivation of individual network nodes. This variant is intended for users who wish to quickly and simply test their ECU in interaction with a prescribed remaining bus simulation.
- As a Project Execution (pex) variant with just a graphic user interface. The test cases and results are very easy to control without requiring special evaluation of the underlying messages.

The CANoe/CANalyzer compatibility mode lets you use both programs, e.g. within a project or an organization, by exchanging uniform configurations. Then the appropriate program can be used in the optimal variant for each use case. During ECU development, the full variant of CANoe is used, while system integrators and testers can use the same configuration in CANalyzer to test the bus communications.

1.3 Product Components

The product contents depend on the selected product variant. The full version contains the following components in addition to CANoe itself:

- Numerous sample configurations of the overall system, on all installed bus system options and on special use cases such as testing and diagnostics
- Editors and display programs for various database formats, for panels and for CAPL programming
- Installation instructions, manuals and online help functions

Other modules, such as OEM-specific TP or IL, are not included with the standard product, but they can be obtained from Support at no extra charge.

1.4 System Requirements

<table>
<thead>
<tr>
<th>Component</th>
<th>Recommended</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel Core i5</td>
<td>3.0 GHz</td>
</tr>
<tr>
<td>Memory (RAM)</td>
<td>4 GB</td>
<td>1 GB</td>
</tr>
<tr>
<td>Hard drive capacity</td>
<td>≥ 2.0 GB (depending on options used and required operating system components)</td>
<td></td>
</tr>
<tr>
<td>Screen resolution</td>
<td>1280 x 1024</td>
<td>1024 x 768</td>
</tr>
<tr>
<td>Graphics card</td>
<td>DirectX 9.0c or higher and Shader Model 1.1 or higher</td>
<td></td>
</tr>
<tr>
<td>Operating system</td>
<td>Windows 10/8.1/8/7/Vista (Vista nur 32 Bit)</td>
<td></td>
</tr>
</tbody>
</table>

1.5 Further Information

- **Vector Download Center**
- **Demo versions**
  Various demo versions are available on the web for CANoe. They contain sample configurations for the various application areas as well as a detailed online help function, in which all CANoe 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.
- **CANoe Feature Matrix**
  More information on variants, channels and bus system support is presented in the feature matrix.
2 Functions

Basic CANoe functions include:

- Simulation of entire systems and remaining bus simulations
- Analysis of the bus communications
- Testing of entire networks and/or individual ECUs
- Creating customized user interfaces to control the simulation and tests or to display analysis data
- Integration of additional I/O hardware and/or special test hardware (VT System)
- Intuitive user interface with flexible docking concept and user-friendly menu structures
- Support of new Vector bus hardware VN0601 (4 TX & 4 RX ARINC 429 channels and I/O)
- User programmability using the CAPL programming language to support simulation, analysis and testing

2.1 Database Support

CANoe supports system descriptions via the DBC format for CAN, ARINC 429 and AFDX®. Information of a database can be symbolically displayed and used in CANoe.

2.2 Special Functions

CANoe highlights include:

- For critical, real-time relevant simulations and tests, CANoe operates in a distributed mode on two PCs
- CAPL-on-Board makes it possible to execute CAPL nodes directly on the interface hardware
- The Vector VT System enables comprehensive ECU tests in which I/O lines are used in addition to bus access
- Test cases may be linked to requirements using commonly used requirements tools such as Telelogic DOORS
- CANoe supports integration of MATLAB/Simulink models
- Control of digital and analog I/O modules as well as measurement hardware permits processing of real signal values in simulations and test environments
- Open software interfaces, such as Microsoft COM, FDX or ASAM XIL API, enable integration in existing system environments.

3 Analysis

The basis for analysis in CANoe 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 considered in an analysis and which data should not.

Highlights:

- Easy to configure the analysis window by drag & 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.

CANoe supplies the user with windows and blocks such as those described below.
3.1 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. VN0601) 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](image)
3.2 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 list of signals opened](image-url)
3.3 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.

- **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 used to display the values transmitted in the messages over a time axis](image-url)
3.4 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.

![Data Window with various representation types for incoming values](image)

3.5 Bus Statistics Window

The Statistics Window shows statistical information about bus activities (ARINC 429) during a measurement. This includes such information as bus load, counters/rates for ARINC Words and errors.

![Bus Statistics for ARINC429](image)

Certain ARINC 429 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 Bus Statistics Window.
3.6 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.

![Write Window with system messages and CAPL outputs](image)

3.7 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.

3.8 Logging/Replay

Data can be logged in CANoe 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.
4 Stimulation/Simulation

CANoe offers many different ways to stimulate ECUs with bus traffic and I/O data.

4.1 Variables and Generators

System variables are available for all simulation and analysis blocks, panels and for the integrated I/O hardware. They are used system-wide to exchange configuration parameters, measurement values or to link external programs via the COM interface.

To stimulate the remaining bus simulation or connected I/O hardware, signals, environment or system variables can be directly introduced by Signal Generators. This makes it easy to inject signal curves such as ramps or sinusoidal waveforms into the system. It is also possible to extract logged signal waveforms from logging files and use them as the generator type.

4.1.1 Interactive Generator

The Interactive Generator (IG) can be used to set signal values, define signal waveforms and send specific messages. This gives users an easy way to stimulate the bus.

> Send messages
  The messages that 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, individual signal values can be modified in the signal list, and signal waveforms (signal curves) may be defined with the integrated Signal Generator. These signal values can then easily be sent on the bus in the associated message, e.g. to check the reaction of an ECU.

> Generate high load situations
  A maximum bus load is reached when one message directly follows another on the bus. This means that after successful message transmission, the next Tx request for resending of the same message is delayed. The continual message volume can simply be configured in the queue.

![Interactive Generator with configured messages and their signals](image)

Figure 8: Interactive Generator with configured messages and their signals
4.1.2 Signal Generator

The Signal Generator can be used to define a waveform for signals and variables (sine, ramp, pulse, value list, etc.).

- **Send values**
  Here, sending of the relevant messages is handled according to a defined send model. In LIN and FlexRay, a schedule table handles sending of messages. In CAN, the interaction layer (IL) handles sending of messages in conjunction with function libraries (DLLs).
  The Signal Generator can be started and stopped during the measurement.

- **Define value waveforms**
  There are two ways to define Signal Generators. A waveform may be defined for a single signal/variable, or it may be loaded from a logging file.

- **Support in panels**
  To automatically stimulate ECUs from panels over a longer time period, Signal Generators may be assigned to the signals/variables. The assigned Signal Generators are visualized on the panel and any errors are shown.

![Figure 9: Signal Generator with user-defined signal waveform](image-url)
4.2 MATLAB/Simulink

Development engineers use the CANoe MATLAB/Simulink integration for functional and application prototyping, integrating complex MATLAB models in CANoe tests and simulations and for developing control algorithms in real-time applications. CANoe and the Simulink models communicate directly via signals and system or environment variables.

The CANoe MATLAB/Simulink integration supports three different execution modes:

- **In the HIL or online mode**, code is generated from the Simulink model that is added as a DLL at a simulated node in CANoe. The model is calculated in real time with CANoe. Automatically generated system variables can be used to make post-run changes to model parameters without recompiling.
- **In offline mode**, the two programs are coupled. Simulink provides the time base, and CANoe is in Slave mode. The entire system operates in simulated mode. It is not possible to access real hardware here.
- **The synchronized mode** is similar to offline mode in its operation. However, in synchronized mode CANoe provides the time base that is derived from the connected hardware. This enables access to real hardware in this mode. One limitation is that the Simulink model must be computed faster than in real time, because in this mode the Simulink simulation is slowed down to adapt the overall simulation to the CANoe simulation time.

4.2.1 Further Information

The application note AN-IND-1-007_Using_MATLAB_with_CANoe describes the use of MATLAB/Simulink together with CANoe. It presents the fundamentals of the CANoe/MATLAB Interface and provides an overview of different use cases.

5 Test

5.1 Testing ECUs and Networks

One of the primary use cases of CANoe is to test ECUs and networks. CANoe services the “System under Test” at all interfaces, in order to attain the most complete test coverage possible.

![Diagram](image)

**Figure 10:** Testing with full access to the ECU

To assure that your testing tasks can be implemented simply and flexibly, the Test Feature Set consists of the following components:

- **In CANoe**, sequential test flows are implemented as test modules or test units, which are subclassified into test groups and test cases. The individual modules can be executed at any time during a measurement.
- **Test modules** may be implemented in XML, CAPL or .NET.
  - In XML modules, tests are assembled from predefined test patterns, and it is easy to parameterize them via input and
output vectors. CAPL and .NET test modules, on the other hand, are programmed and therefore exhibit very flexible test flow control. The .NET test modules can be conveniently developed in C# or VB.NET. The different description forms may be combined according to requirements.

Test modules are managed in test environments. These test environments contain test modules as well as further function blocks for test execution. Test environments are saved separately from the CANoe system configuration. Therefore they are easy to reuse them in different projects.

> **Test units** (only for CAN) contain a collection of data which contain the implementation of the tests (e.g. CAPL/C# files, test tables, test diagrams or parameter files). A test unit may contain for example the implementation of test for a specific ECU function.

Test units are managed within test configurations. A test configuration may contain 1...n test units which are sequentially executed in a consecutive manner. A CANoe system configuration on the other hand may contain any number of different test configurations which may be executed parallelly. Executable test units are developed with vTESTstudio which is a separate product.

> In parallel to test execution, other system states can be checked such as conformance to the cycle times of individual messages. These constraints are automatically added to the test evaluation.

> The **Test Service Library** contains a collection of prepared test functions that simplify the process of setting up tests. They are used in the test modules and are parameterized via the database. For example, it is possible to monitor the cycle times of messages, an ECU’s reaction time from the time a message is received until it sends the response message or the validity of signal values and diagnostic parameters. To evaluate the quality of the tested ECUs, different statistical values of the tests are output, such as the number of reported deviations over the test time period.

> When a test module or test unit is executed, an extensive test report is generated. Along with the names of the executed test cases and the individual test results, user-defined information or automatic screenshots of various analysis windows can also be recorded, for example. CANoe writes the results to a flexibly further processed XML file. The output format for the test report is adapted via an XSLT stylesheet.

> Direct control of I/O hardware in CANoe makes it possible to use analog and digital ECU interfaces in addition to bus communications. Along with standard I/O components, the Vector **VT System** represents a modular hardware system for comprehensively testing ECUs in the automotive field.

![Image of test report](image.png)

**Figure 11:** Test flow with associated test report in HTML

### 5.2 Further Information

Fundamental concepts of CANoe’s Test Feature Set are described in application note **AN-IND-1-002_Testing_with_CANoe**.
6 Programming

6.1 CAPL Interface

The CAPL (Communication Access Programming Language) programming language extends the functional scope of CANoe 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. CANoe assumes control.
> 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 (simulation, testing, diagnostics and analysis of various bus systems)
> Flexible extension by external libraries

6.1.1 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.

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

6.1.2 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 – or, if time controlled, then triggered by the hardware or internal to CANoe.

Here are just a few examples of these event handlers:

<table>
<thead>
<tr>
<th>Event Handler</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>On timer seconds cycle</td>
<td>Time controlled</td>
</tr>
<tr>
<td>On a429word FlapStatus</td>
<td>Message input or output</td>
</tr>
<tr>
<td>On signal update</td>
<td>Rewrites signal value</td>
</tr>
<tr>
<td>On sysvar</td>
<td>Modifies system variable</td>
</tr>
<tr>
<td>On a429worderror</td>
<td>Detects ARINC 429 errors</td>
</tr>
</tbody>
</table>
6.1.3 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.

> Physical access to signal values:
  // Definition of the representation in the database
  $\text{EnergyMgmt::BatteryVoltage} = 14.1;

> Access to raw value of a signal:
  // 8 to 18 Volt with 12bit resolution, without range check
  $\text{EnergyMgmt::BatteryVoltage.raw} = (14.1 - 8) / (18 - 8) * 4096;

> Access on a message base:
  a429word EnergyMgmt msg;
  // Most significant bytes Intel coded
  // with 12bit only the lower 4 bits are used
  // details are specified via database
  msg.BatteryVoltage = 14.1;
  output(msg);

6.1.4 Application-Specific Language Extensions

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

> Stimulation
  CAPL can also be used to generate messages for the purpose of stimulating ECUs. This relieves the developer of routine work tasks. Signals and messages of the buses are defined in the database (e.g. in DBC file).

> Testing
  CAPL also offers convenient control options for programming automated tests that support both test execution and evaluation. Just a few lines of code is all it takes to create a basic structure that utilizes the test flow and automated reporting. So, with just a small program, a CAPL test node is able to provide a well-organized summary of the test flow with the standard report.

testcase MyTestCase()
{
  const char* MyTestCaseTitle = "myTC", MyTest Case"
  const char* MyTestCaseDescription = "A test of mine"
  const char* MyTestCaseComment = "first take a short break"

  TestWaitForTimeout(200);
  if ( MyTestExecution () > 0 )
    TestStepPass("myTC successful");
  else
    TestStepFail("myTC failed");
}

long MyTestExecution ()
{
  /* Own code */
  return 1;
}

MyTest() {
  MyTestCase();
  TestSetVerdictModule(TestGetVerdictLastTestCase());
}
CAPL can also be used in the analysis of measurement results - 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.

```cpp
// On a429word Brake {
    long TempCounter = 0;
    $BRECounter++;
    // Take a look only at the last 10000 values
    TempCounter = $BRECounter;
    if (BREZCounter > 100000)
        TempCounter = 100000;

    $Brake::Pressure = $Brake::Pressure * TempCounter + @AveragePressure;

    output ( this );
}
```
6.2 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 CANoe 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 in what is known as the Symbol Explorer:

> Network symbols such as nodes, messages and signals
> Environment data, i.e. database-specific environment variables and system variables that are used CANoe-wide
6.3 Debugging

The Vector debugger is available for debugging CAPL and .NET programs. It can be used to insert breakpoints in the source text of the programs and to check the values of variables.

It is possible to debug all CAPL and .NET programs in the simulated mode, because the simulation is stopped for this purpose. When real hardware is used, it is only possible to debug in the programs of test modules, because the events sent by the hardware still need to be evaluated.

Figure 14: Debugger with breakpoints
7 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. Different types of panels are available in CANoe.

> Signal panel
A signal panel offers a simple way to modify signal values at measurement time. A distinction is made between node panels and network panels among the signal panels. When a node panel is used, the Tx signals of the related node are automatically configured. When network panels are used, the Tx signals of the entire network are automatically configured.

> Symbol panel
The symbol panel can be used to display and/or modify the values of signals and variables during the simulation.

> User-defined panels
User-defined panels are user interfaces for special use cases. Such panels might be used to control the simulation and test environment, for example, or to display the analysis data from CAPL 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 & 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.

> User-programmable ActiveX and .NET panels
These panels that are created with programming languages such as Visual Basic 6.0, Visual Basic.NET or C# can be integrated in CANoe.

![User-defined panels for displaying signal and variable values](image15.png)

Figure 15: User-defined panels for displaying signal and variable values
8 Hardware Interfaces

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

![Figure 16: Overview of Vector hardware for CAN, ARINC 429 and AFDX/Ethernet](image16)

![Figure 17: The compact VN0601 USB interface for the ARINC 429 bus system can operate up to four RX and four TX channels simultaneously](image17)
9 Interfaces to Other Applications

9.1 COM Interface

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 CANoe 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 Simulation Setup
- Control of automated tests, start test execution, add test modules
- Access to signals and system variables, access to CAPL functions, compiling of CAPL nodes

Visual Basic script example for starting the measurement:

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

Visual Basic script example for opening a configuration:

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

9.1.1 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.

9.2 FDX

CANoe FDX (Fast Data eXchange) is a protocol, with which data can be exchanged between CANoe and other systems simply, quickly and with minimal delay over an Ethernet connection. The protocol gives other systems read and write access to system variables, environment variables and bus signals of CANoe. It is also possible to send control commands to CANoe via FDX (e.g. for starting and stopping the measurement) or receive status information.

The other system might be an HIL system in a test bench, for example, or a PC that should display data from CANoe. The protocol is based on the widely used standard protocol UDP (based on IPv4).

9.3 ASAM XIL API

ASAM XIL is an API standard for the communication between test automation tools and test benches. All types of test benches are supported, e.g. SIL, MIL and HIL.

Data between CANoe and the test benches is exchanged by utilizing prepared .NET assemblies with a C# API.

10 Training

As part of our training program, we offer various classes and workshops on CANoe in our classrooms at Vector and on-site at our customers.
You will find more information on individual training courses and a schedule online at: www.vector-academy.com
Get More Information

Visit our website for:
> News
> Products
> Demo software
> Support
> Training classes
> Addresses

www.vector.com