

User Manual LabVIEW Interface

Version 2.1
English

Imprint

Vector Informatik GmbH
Ingersheimer Straße 24
D-70499 Stuttgart

Vector reserves the right to modify any information and/or data in this user documentation without notice. This documentation nor any of its parts may be reproduced in any form or by any means without the prior written consent of Vector. To the maximum extent permitted under law, all technical data, texts, graphics, images and their design are protected by copyright law, various international treaties and other applicable law. Any unauthorized use may violate copyright and other applicable laws or regulations.

© Copyright 2018, Vector Informatik GmbH. Printed in Germany.
All rights reserved.

Table of Contents

1 Introduction	5
1.1 Product Overview	6
1.2 About this User Manual	7
1.2.1 Conventions	7
1.2.2 Certification	8
1.2.3 Warranty	8
1.2.4 Support	8
1.2.5 Trademarks	8
2 CANalyzer	9
2.1 CarSpeed – CANMainDemo.vi Example	10
2.2 User Library	11
2.2.1 Signal	11
2.2.2 Signal Read	12
2.2.3 Signal Write	12
2.2.4 Unique Signal Name	12
2.2.5 Start	13
2.2.6 Stop	13
2.2.7 CAPL Function	13
2.2.8 CAPL Function Call	14
3 CANoe	15
3.1 Easy.vi Example	16
3.2 More Examples	17
3.3 User Library	18
3.3.1 EnvVar	18
3.3.2 EnvVar Read	19
3.3.3 EnvVar Write	19
3.3.4 EnvVar Value	20
3.3.5 Signal	20
3.3.6 Signal Read	21
3.3.7 Signal Write	21
3.3.8 Unique Signal Name	21
3.3.9 SysVar	22
3.3.10 SysVar Read	22
3.3.11 SysVar Write	23

3.3.12 Start	23
3.3.13 Stop	23
3.3.14 CAPL Function	24
3.3.15 CAPL Function Call	24

1 Introduction

In this chapter you find the following information:

1.1 Product Overview	6
1.2 About this User Manual	7
1.2.1 Conventions	7
1.2.2 Certification	8
1.2.3 Warranty	8
1.2.4 Support	8
1.2.5 Trademarks	8

1.1 Product Overview

The **CANalyzer/CANoe** COM server can be activated by means of the ActiveX library of **LabVIEW**. In this way all objects (as well as their methods, properties and events) supplied by the COM server are available in **LabVIEW**. The following examples have been created with **LabVIEW** 9.0 and show the fundamental procedure to activate the **CANalyzer/CANoe** COM server.

**Note**









The **LabVIEW** interface 2.0 requires **CANalyzer/CANoe** 8.0 or newer.





1.2 About this User Manual

1.2.1 Conventions

In the two tables below you will find the notation and icon conventions used throughout the manual.

Style	Utilization
bold	Fields/blocks, user/surface interface elements, window- and dialog names of the software, special emphasis of terms [OK] Buttons in brackets File Save Notation for menus and menu commands
Microsoft	Legally protected proper names
Source Code	File and directory names, source code, class and object names, object attributes and values
Hyperlink	Hyperlinks and references
<CTRL>+<S>	Notation for key combinations

Symbol	Utilization
	Dangers that could lead to damage
	Notes and tips that facilitate your work
	More detailed information
	Examples
	Step-by-step instructions
	Text areas where changes of the currently described file are allowed or necessary
	Files you must not change
	Multimedia files e.g. video clips

Symbol	Utilization
	Introduction into a specific topic
	Text areas containing basic knowledge
	Text areas containing expert knowledge
	Something has changed

1.2.2 Certification

Vector Informatik GmbH has ISO 9001:2008 certification. The ISO standard is a globally recognized standard.

1.2.3 Warranty

We reserve the right to modify the contents of the documentation or the software without notice. Vector disclaims all liabilities for the completeness or correctness of the contents and for damages which may result from the use of this documentation.

1.2.4 Support

You can get through to our hotline at the phone number

+49 (711) 80670-200

or you send a problem report to the Vector Informatik GmbH Support.

1.2.5 Trademarks

All brand names in this documentation are either registered or non registered trademarks of their respective owners.

2 CANalyzer

In this chapter you find the following information:

2.1 CarSpeed – CANMainDemo.vi Example	10
2.2 User Library	11
2.2.1 Signal	11
2.2.2 Signal Read	12
2.2.3 Signal Write	12
2.2.4 Unique Signal Name	12
2.2.5 Start	13
2.2.6 Stop	13
2.2.7 CAPL Function	13
2.2.8 CAPL Function Call	14

2.1 CarSpeed – CANMainDemo.vi Example

This VI starts/stops the measurement and accesses a signal object of the **CANalyzer** (CarSpeed). It reads this signal cyclically and displays its value in a chart. Please use it together with the **CANMainDemo** configuration in the **CANALYZER** folder of the **LabVIEW AddOn** installation.

CANalyzer signals are identified by their symbolic name. A fully qualified name consists of a database name, a sender node name, a message name and the name of the signal itself.



Example

mostbus::ABS::ABSdata::CarSpeed

It is possible to leave out the left most parts of the name as long as the signal name remains unique in the current configuration.



Example

- ▶ ABS::ABSdata::CarSpeed
- ▶ ABSdata::CarSpeed
- ▶ CarSpeed

To ease configuration the following XControl should be used to select the signal name at design time:

- ▶ USER.LIB\SymbSelBtn CL.xctl

It is also available from a user control palette named **CANoe/CANalyzer**. Drop the XControl on the front panel of a VI and press the **[Select...]** button. A selection dialog will be opened containing the signals of the current **CANalyzer** configuration. After selecting a signal inside the dialog, the shortest unique signal name will be taken over by the XControl.

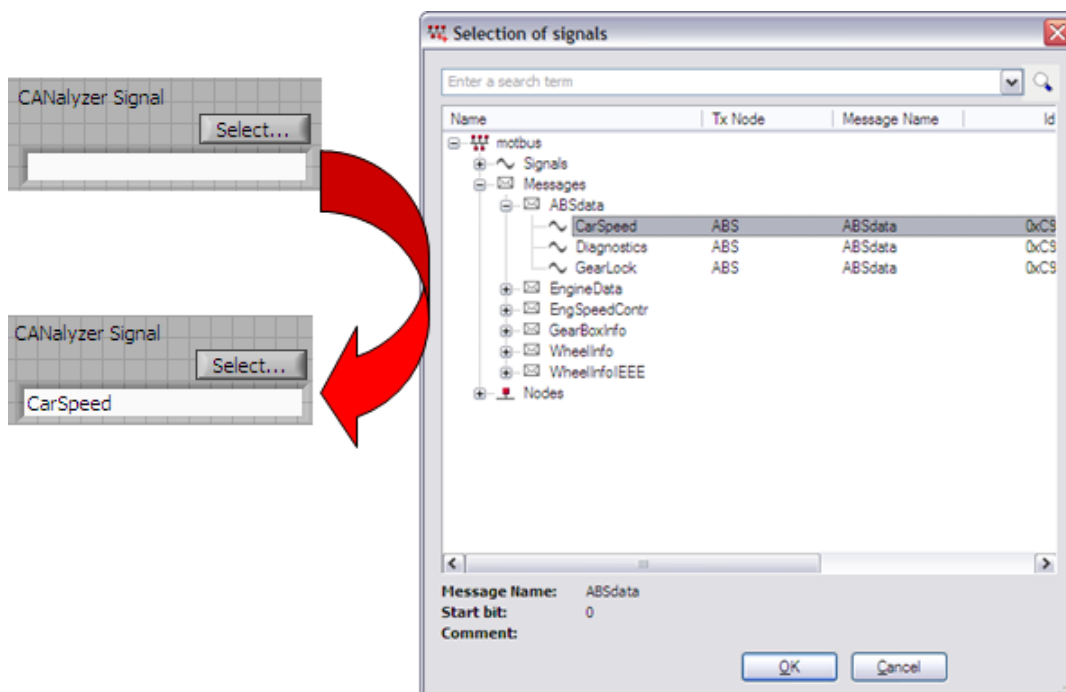


Figure 1: How to add a signal

2.2 User Library

The user library for **CANalyzer** contains reusable sample VIs for common tasks. You can find these VIs after installation of the **LabVIEW** interface in the following library file inside the **LabVIEW** installation path:

► USER.LIB_CANalyzer.llb

There is also a user library named **CANalyzer** which contains the VIs used for accessing the **CANalyzer**:

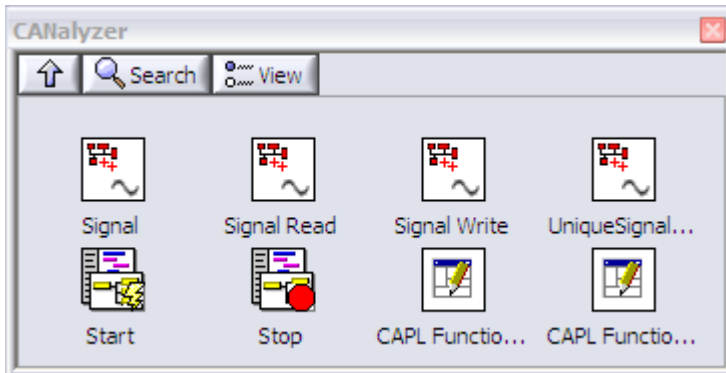




Figure 2: CANalyzer library

2.2.1 Signal

Symbol	
VI Name	Signal CL.vi
Description	Returns the CANalyzer COM signal object which belongs to the given unique signal name.
Inputs	Unique signal name (String) Refer to the example in 2.1.
Outputs	Signal (Variant) COM reference of a CANalyzer signal object.

2.2.2 Signal Read

Symbol	
VI Name	Signal Read CL.vi
Description	Returns the current physical value of a signal object.
Inputs	Signal (Variant) COM Reference of a CANalyzer signal object.
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the signal object is not needed any longer.
	Value (Double) Current physical value of the signal.

2.2.3 Signal Write

The value of signal objects currently can't be changed in **CANalyzer**. As an alternative, you can use the **CAPL** Function Call VI to send a message. The function can be given the signal value(s) as parameter(s).




Example


```
void SendMessageA(double sig1, double sig2)
{
    message A msg;
    msg.sig1 = sig1;
    msg.sig2 = sig2;
    output(msg);
}
```

You can also send a message cyclically with a CAPL timer, and store the signal values inside the CAPL program as variables.


2.2.4 Unique Signal Name

Symbol	
VI Name	UniqueSignalName.vi
Description	Converts a fully qualified signal name into its shortest unique name.
Inputs	Full signal name (String)
Outputs	Unique signal name (String)


2.2.5 Start

Symbol	
VI Name	Measurement Start CL.vi
Description	Starts the measurement. If no instance of CANalyzer is running, the latest installed version will be started.
Inputs	—
Outputs	—


2.2.6 Stop

Symbol	
VI Name	Measurement Stop CL.vi
Description	Stops the measurement.
Inputs	—
Outputs	—

2.2.7 CAPL Function

Symbol	
VI Name	CAPL Function CL.vi
Description	Returns the CANalyzer COM object of a CAPL function. This VI does only work inside the OnInit callback of CANalyzer , i.e. during measurement start. Please refer to 3.2 for correct usage.
Inputs	CAPL function name (String)
Outputs	CAPL function (Reference) COM reference of a CAPL function.

2.2.8 CAPL Function Call

Symbol	
VI Name	CAPL Function Call CL.vi
Description	Calls a CAPL function.
Inputs	CAPL Function (Variant) COM Reference of a CAPL function
	p1 ... p10 (Variant) Up to 10 optional parameter values of the function. The first unconnected parameter defines the total number of parameters. The number of parameters as well as their types must match with that of the CAPL function in CANalyzer .
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the CAPL function object is not needed any longer.
	Return Value (Integer) Return value of the function. It is only valid for functions whose CAPL programs are configured in the evaluation branch.

3 CANoe

In this chapter you find the following information:

3.1 Easy.vi Example	16
3.2 More Examples	17
3.3 User Library	18
3.3.1 EnvVar	18
3.3.2 EnvVar Read	19
3.3.3 EnvVar Write	19
3.3.4 EnvVar Value	20
3.3.5 Signal	20
3.3.6 Signal Read	21
3.3.7 Signal Write	21
3.3.8 Unique Signal Name	21
3.3.9 SysVar	22
3.3.10 SysVar Read	22
3.3.11 SysVar Write	23
3.3.12 Start	23
3.3.13 Stop	23
3.3.14 CAPL Function	24
3.3.15 CAPL Function Call	24

3.1 Easy.vi Example

This VI starts/stops the measurement and accesses signal, environment variable and system variable objects of **CANoe**. It reads and writes their value. Please use it together with the **Easy** configuration in the **CANOE** folder of the **LabVIEW AddOn** installation.

CANoe object are identified by their symbolic name. A fully qualified signal name consists of a database name, a sender node name, a message name and the name of the signal itself.



Example

```
easy::LightSwitch::LightState::OnOff
```

It is possible to leave out the left most parts of the name as long as the signal name remains unique in the current configuration.



Example

- ▶ LightSwitch::LightState::OnOff
- ▶ LightState::OnOff
- ▶ OnOff (not unique since there is another signal named OnOff in the configuration: MotorState::OnOff)

Environment variables are identified by their name. This name has to be unique throughout all databases.

The names of System variables contain a namespace name and the variable name. No part is optional.



Example

- ▶ LightSystem::OnOff
- ▶ WiperSystem::Active

To ease configuration the following XControl should be used to select the symbolic names at design time:

- ▶ USER.LIB\SymbSelBtn.xctl

It is also available from a user control palette named **CANoe/CANalyzer**. Drop the XControl on the front panel of a VI, choose the correct type (signal, environment or system variable) and press the **[Select...]** button. A selection dialog will be opened containing objects of the current **CANoe** configuration. After selecting a signal inside the dialog, the shortest unique signal name will be taken over by the XControl.

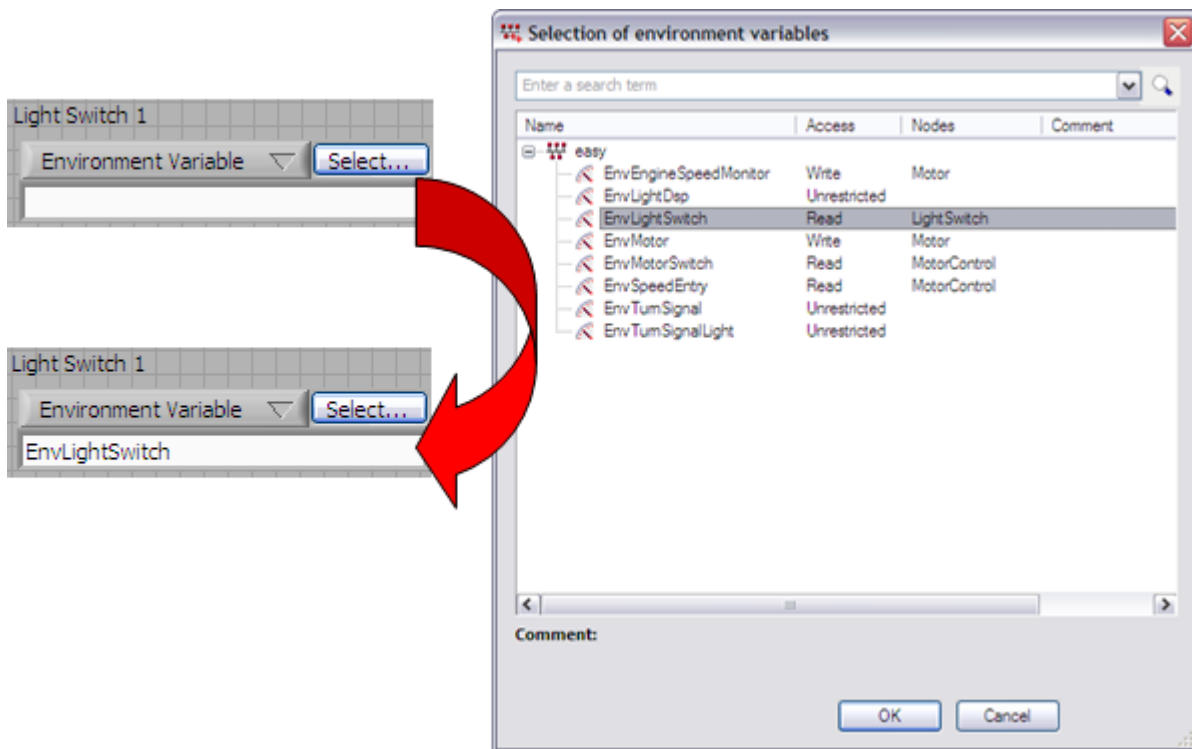


Figure 3: How to add an environment variable

3.2 More Examples

Use the **More Examples** CANoe configuration in the CANOE\More Examples folder of the **LabVIEW AddOn** installation for the following sample VIs. They show some less common or more complicated use cases.

► CAPL Functions.vi

This example shows how to call CAPL functions from **LabVIEW**. It is only possible to connect to a CAPL function during the **OnInit** callback at measurement start of **CANoe**. The example VI registers another VI named **CAPL Functions – OnInitCallback.vi** for this COM callback. It also makes use of a global variable (CAPL Functions – GlobalVariable.vi) for data exchange with the callback VI. The global variable contains a reference to the CAPL function COM object and a Boolean value to indicate that the **OnInit** callback was executed.

► SystemVariableTypes.vi

This example VI shows how to use all types of system variables correctly. It covers reading as well as writing of all value types. In contrast to other **CANoe** objects the measurement doesn't have to be started to access the values of system variables. If the measurement is running, the changes of values done in LabVIEW can be observed in **CANoe**.

3.3 User Library

The user library for **CANoe** contains reusable sample VIs for common tasks. You can find these VIs after installation of the **LabVIEW** interface in the following library file inside the **LabVIEW** installation path:

► USER.LIB_CANoe.lib

There is also a user library named **CANoe** which contains the VIs used for accessing the **CANoe**:

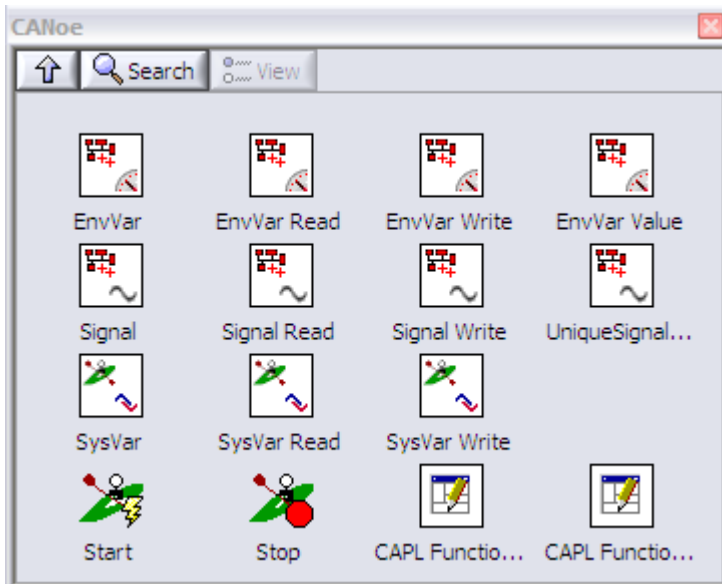




Figure 4: CANoe library


3.3.1 EnvVar

Symbol	
VI Name	Environment Variable.vi
Description	Returns the CANoe COM environment variable object which belongs to the given name.
Inputs	Name (String) Name of the environment variable.
Outputs	Variable (Variant) COM reference of an environment variable object.


3.3.2 EnvVar Read

Symbol	
VI Name	Environment Read.vi
Description	Returns the current value of an environment variable object.
Inputs	Variable (Variant) COM reference of an environment variable object
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the environment variable object is not needed any longer.
	Value (Variant) Current value of the environment variable. The value must be converted using a Variant To Data VI according to the type defined by the database. Possible data types: Integer, Float, String or Array of Byte.


3.3.3 EnvVar Write

Symbol	
VI Name	Environment Write.vi
Description	Sets the value of an environment variable object.
Inputs	Variable (Variant) COM reference of an environment variable object.
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the environment variable object is not needed any longer.


3.3.4 EnvVar Value

Symbol	
VI Name	Environment Value.vi
Description	Returns and sets the value of an environment variable object using only one COM call to CANoe .
Inputs	Variable (Variant) COM reference of an environment variable object.
	New Value (Variant)
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the environment variable object is not needed any longer.
	Value (Variant) Current value of the environment variable.


3.3.5 Signal

Symbol	
VI Name	Signal.vi
Description	Returns the CANoe COM signal object which belongs to the given unique signal name.
Inputs	Unique signal name (String) Refer to the example in 3.1.
Outputs	Signal (Variant) COM reference of a CANoe signal object.


3.3.6 Signal Read

Symbol	
VI Name	Signal Read.vi
Description	Returns the current physical value of a signal object.
Inputs	Signal (Variant) COM reference of a CANoe signal object.
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the signal object is not needed any longer.
	Value (Double) Current physical value of the signal.


3.3.7 Signal Write

Symbol	
VI Name	Signal Write.vi
Description	Sets the physical value of a signal object.
Inputs	Signal (Variant) COM reference of a CANoe signal object.
	New Value (Double) New physical value of the signal.
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the signal object is not needed any longer.


3.3.8 Unique Signal Name

Symbol	
VI Name	UniqueSignalName.vi
Description	Converts a fully qualified signal name into its shortest unique name.
Inputs	Full signal name (String)
Outputs	Unique signal name (String)


3.3.9 SysVar

Symbol	
VI Name	System Variable.vi
Description	Returns the CANoe COM system variable object which belongs to the given name.
Inputs	Name (String) Name (including namespace) of the system variable.
Outputs	Variable (Variant) COM reference of a system variable object.


3.3.10 SysVar Read

Symbol	
VI Name	System Variable Read.vi
Description	Returns the current value of a system variable object.
Inputs	Variable (Variant) COM Reference of a system variable object.
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the system variable object is not needed any longer.
	Value (Variant) Current value of the system variable. The value must be converted using a Variant To Data VI according to the type defined by the database. Possible data types: Integer, Float, String, Array of Integer or Array of Float.


3.3.11 SysVar Write

Symbol	
VI Name	System Variable Write.vi
Description	Sets the value of a system variable object.
Inputs	Variable (Variant) COM reference of a system variable object.
	New Value (Variant)
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the system variable object is not needed any longer.


3.3.12 Start

Symbol	
VI Name	Measurement Start.vi
Description	Starts the measurement. If no instance of CANoe is running, the latest installed version will be started.
Inputs	—
Outputs	—


3.3.13 Stop

Symbol	
VI Name	Measurement Stop.vi
Description	Stops the measurement.
Inputs	—
Outputs	—

3.3.14 CAPL Function

Symbol	
VI Name	CAPL Function.vi
Description	Returns the CANoe COM object of a CAPL function. This VI does only work inside the OnInit callback of CANoe , i.e. during measurement start. Please refer to 3.2 for correct usage.
Inputs	CAPL Function Name (String)
Outputs	CAPL Function (Reference) COM reference of a CAPL function.

3.3.15 CAPL Function Call

Symbol	
VI Name	CAPL Function Call.vi
Description	Calls a CAPL function.
Inputs	CAPL Function (Variant) COM Reference of a CAPL function. p1 ... p10 (Variant) Up to 10 optional parameter values of the function. The first unconnected parameter defines the total number of parameters. The number of parameters as well as their types must match with that of the CAPL function in CANoe .
Outputs	Automation RefNum (Reference) COM reference which should be freed using a Close Reference VI if the CAPL function object is not needed any longer. Return Value (Integer) Return value of the function. It is only valid for functions whose CAPL programs are configured in the measurement branch.



More Information

- ▶ News
- ▶ Products
- ▶ Demo Software
- ▶ Support
- ▶ Training Classes
- ▶ Addresses

www.vector.com