



Manual

Vector CAN Driver

for LabVIEW

Version 1.6

English

Imprint

Vector Informatik GmbH
Ingersheimer Straße 24
D-70499 Stuttgart

The information and data given in this user manual can be changed without prior notice. No part of this manual may be reproduced in any form or by any means without the written permission of the publisher, regardless of which method or which instruments, electronic or mechanical, are used. All technical information, drafts, etc. are liable to law of copyright protection.

© Copyright 2013, Vector Informatik GmbH
All rights reserved.

Table of contents

1	Introduction	3
1.1	About this User Manual	4
1.1.1	Certification	5
1.1.2	Warranty	5
1.1.3	Registered trademarks	5
2	CAN Driver for LabVIEW	6
2.1	General Information	7
2.1.1	Liability	7
2.1.2	Hardware and Software Requirements for PC	7
2.1.3	Supported Vector Hardware	7
2.1.4	Included in Delivery	7
2.1.5	Installation	8
2.1.6	Executable LabVIEW Projects	10
2.2	Basics	11
2.2.1	Principle of Vector CAN Driver	12
2.2.2	Data stream, blocks and time slots	12
2.2.3	Assigning signal values	13
2.2.4	Example 1 – Assigning of multiple signals	14
2.2.5	Example 2 – Small block size and block frequency	15
2.2.6	Example 3 – Small block size and high block frequency	15
2.2.7	Example 4 – Dimensioning block size and block frequency	16
3	Vector Virtual Instruments (VIs)	17
3.1	Access in LabVIEW	18
3.1.1	Open Project	18
3.1.2	Close Project	19
3.1.3	Update Send Buffer	19
3.1.4	Trigger Send	20
3.1.5	Trigger Send Single	20
3.1.6	Get Block Size	20
3.1.7	Get Wait MS	21
3.1.8	Read	21
3.1.9	Error Code to String	21
3.1.10	Get Status	22
4	Vector LabVIEW Configuration Tool	23
4.1	Short Description	24
4.2	Tree View	24
4.2.1	General	24
4.2.2	CAN Channels	25
4.2.3	Signal Configuration	26
5	Example SendReceiveLoop	30
5.1	Short Description	31
5.2	Running the example	31
6	Error Codes	34

6.1	Overview	35
-----	----------	----

1 Introduction








In this chapter you find the following information:

1.1	About this User Manual	page 4
	Certification	
	Warranty	
	Registered trademarks	

1.1 About this User Manual

Conventions

In the two following charts you will find the conventions used in the user manual regarding utilized spellings and symbols.

Style	Utilization
bold	Blocks, surface elements, window- and dialog names of the software. Accentuation of warnings and advices. [OK] Push buttons in brackets File Save Notation for menus and menu entries
Microsoft	Legally protected proper names and side notes.
Source Code	File name and source code.
Hyperlink	Hyperlinks and references.
<CTRL>+<S>	Notation for shortcuts.
Symbol	Utilization
	Here you can obtain supplemental information.
	This symbol calls your attention to warnings.
	Here you can find additional information.
	Here is an example that has been prepared for you.
	Step-by-step instructions provide assistance at these points.
	Instructions on editing files are found at these points.
	This symbol warns you not to edit the specified file.

1.1.1 Certification

Certified Quality Management System

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

1.1.2 Warranty

Restriction of warranty

We reserve the right to change the contents of the documentation and the software without notice. Vector Informatik GmbH assumes no liability for correct contents or damages which are resulted from the usage of the documentation. We are grateful for references to mistakes or for suggestions for improvement to be able to offer you even more efficient products in the future.

1.1.3 Registered trademarks

Registered trademarks

All trademarks mentioned in this documentation and if necessary third party registered are absolutely subject to the conditions of each valid label right and the rights of particular registered proprietor. All trademarks, trade names or company names are or can be trademarks or registered trademarks of their particular proprietors. All rights which are not expressly allowed are reserved. If an explicit label of trademarks, which are used in this documentation, fails, should not mean that a name is free of third party rights.

→ **Windows, Windows XP, Windows Vista, Windows 7, Windows 8** are trademarks of the Microsoft Corporation.

2 CAN Driver for LabVIEW

In this chapter you find the following information:

2.1	General Information	page 7
	Liability	
	Hardware and Software Requirements for PC	
	Supported Vector Hardware	
	Included in Delivery	
	Installation	
	Executable LabVIEW Projects	
2.2	Basics	page 11
	Principle of Vector CAN Driver	
	Data stream, blocks and time slots	
	Assigning signal values	
	Example 1 – Assigning of multiple signals	
	Example 2 – Small block size and block frequency	
	Example 3 – Small block size and high block frequency	
	Example 4 – Dimensioning block size and block frequency	

2.1 General Information

2.1.1 Liability



Caution: Using this driver can be dangerous. Please use it with care.

Since you can influence or operate a control system with this driver, your actions when using this could lead to serious damage to persons or property. For this reason, only those persons may use this driver who understand the possible consequences of their actions with this driver or who have been specially trained in how to handle this driver.

If other persons use this driver, the Vector Informatik GmbH Company assumes no warranty or liability over and above troubleshooting or refunding the purchase price.

2.1.2 Hardware and Software Requirements for PC

CPU	Pentium III
Working memory	512 MB RAM or more
Operating system	Windows XP or Windows 7
Software	LabView V7.0 or higher (32 bit)
Hardware	Vector Hardware with LabVIEW license

2.1.3 Supported Vector Hardware

- CAN Interfaces**
- CANcardXL
 - CANcardXLc
 - CANboardXL / PCIe / pxi
 - CANcaseXL / log
 - VN16xx
 - VN8910 mit VN8950/VN8970

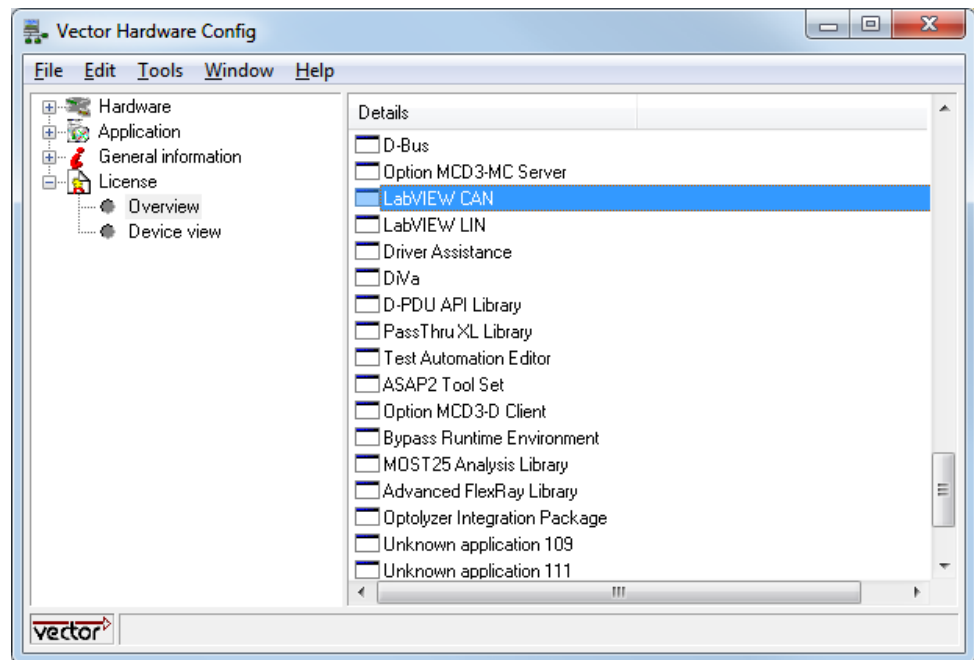
2.1.4 Included in Delivery

- Vector CAN Driver `VectorLabVIEW.dll`
- Vector LabVIEW Library `VectorLabVIEW.llb`
- Example `SendReceiveLoop.vi`
- Project editor `vlvconf.exe`
- Database editor `CANdb++` (separate setup)
- This manual (PDF)

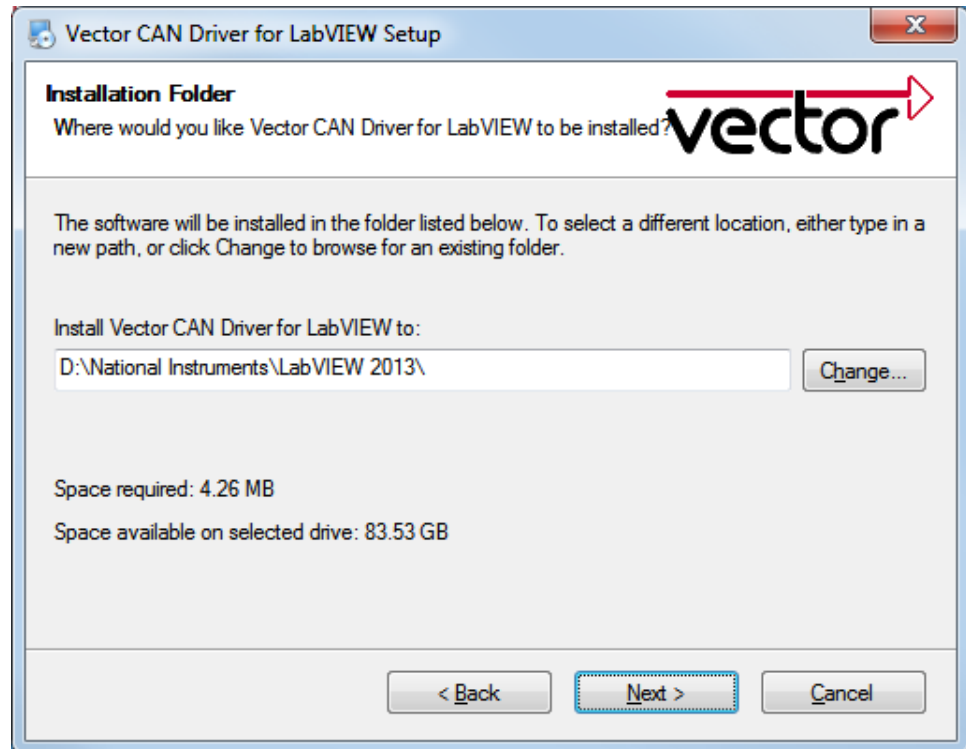
2.1.5 Installation



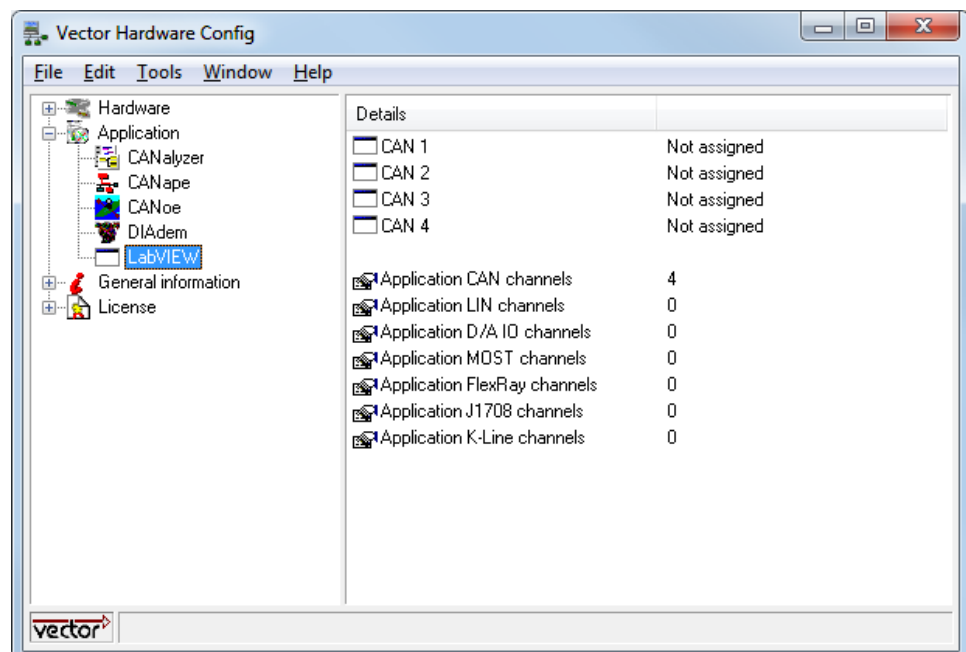
1. Ensure that LabVIEW 7.0 or higher (32 bit) is installed on your PC.
2. Make sure that the driver of your Vector hardware (e.g. CANcardXL) is installed on your PC. The driver is found in the delivery of the hardware.
3. Check if your Vector hardware is licensed for LabVIEW. Click **Start | Settings | Control Panel | Vector Hardware** and open section **License | Overview**. The entry **LabVIEW CAN** must be found in the right part of the window.



4. Please run the installation from the Vector Driver Disk in
`\Tools\CAN Driver for LabVIEW\SetupVectorLabVIEW.exe`.
5. Follow the instructions of the setup and click the **[Next >]** button.
6. The setup will determine the path of the current LabVIEW installation. You can also change the path by clicking the **[Change...]** button. The driver DLL and the examples are installed into the following LabVIEW sub folder
`...user.lib_express\Vector Informatik`.



7. Click the **[Next >]** button to continue the installation setup.
8. Finish the installation by clicking the **[Finish]** button.
9. Open the Vector Hardware Configuration (**Start | Settings | Control Panel | Vector Hardware**) and assign your desired Vector Hardware to the application **LabVIEW**.



10. Ensure that the software time synchronization is switched on. Click **Start | Settings | Control Panel | Vector Hardware** and open section **General information | Settings** and switch **Software time synchronization** to **YES**.

2.1.6 Executable LabVIEW Projects

Embedding Vector CAN Driver

If the Vector CAN Driver is used in an executable LabVIEW project, the following files have to be embedded:

- sLabCloseProject.vi
- sLabConvertErrCode2String.vi
- sLabGetBlockSz.vi
- sLabGetWaitMs.vi
- sLabOpenProject.vi
- sLabRead.vi
- sLabTrigSendSingle.vi
- sLabUpdateSendBuffer.vi
- CANdt2.dll
- cbdmslfho.dll
- smbsk.dll
- smbsk01.dll
- VectorLabView.dll
- xlLogAGui.dll
- xlLogAGuiSym.dll
- xlLogApi.dll

2.2 Basics

LabVIEW and Vector Hardware

In LabVIEW, the Vector CAN Driver provides indirect access to the Vector Hardware and enables sending and receiving of CAN signals.

Access to CAN signals is exclusively granted by symbolic names, which are defined in CANdb databases. The conversion of raw CAN data into physical values and vice versa is done internally by corresponding conversion rules in the database.













Info: If the signal configuration changes (e.g. bit position, conversion rules), only the database has to be edited.



Info: The setup for the database editor CANdb++ can be found on the Vector Driver Disk.

Vector VIs

The CAN Library **VectorLabVIEW.lib** offers the following Vector VI's:

-  **Open Project**
-  **Close Project**
-  **Read**
-  **Update Send Buffer**
-  **Trigger Send**
-  **Trigger Send Single**
-  **Get Wait MS**
-  **Get Block Size**
-  **Convert Error Code to String**
-  **Get Status**

A description of the Vector VI's is found in chapter **Vector Virtual Instruments (VIs)** on page 17.

2.2.1 Principle of Vector CAN Driver

Data stream with fixed sampling rate

The Vector CAN Driver simulates a data stream from an asynchronously received CAN signal with a fixed sampling rate. The received signal is integrated into this data stream according to its time stamp. In the following the principle of the data transmission from the CAN bus to LabVIEW is described in detailed.

2.2.2 Data stream, blocks and time slots

Data transmission driver/LabVIEW

A data stream is a continuous flow of blocks, which is defined by the **Block frequency** (see chapter General on page 24). Each block consists of at least one time slot where a signal value can be held. The amount of time slots in a block is defined by the **Block size** (see chapter General on page 24). The higher the block size, the more values can be held in one block.



Info: The length of a time slot decreases, the higher the block size is set at a constant block frequency.

Data stream

The following drawings depicts different data streams depending on their block frequency and block size:

Starting example

Block size = 4
(four time slots)

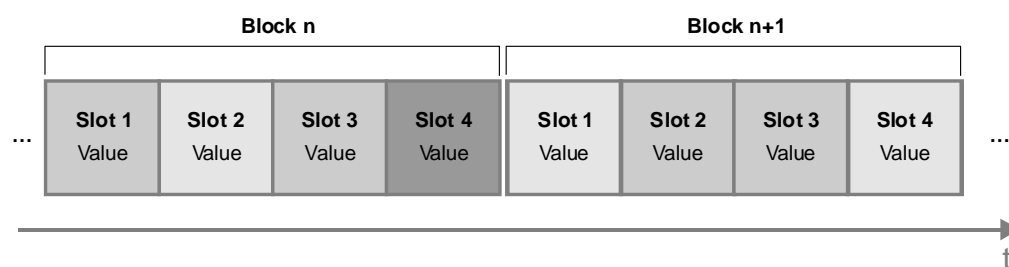


Figure 1: Data stream with block size 4.

Variation 1:

Doubled
block frequency

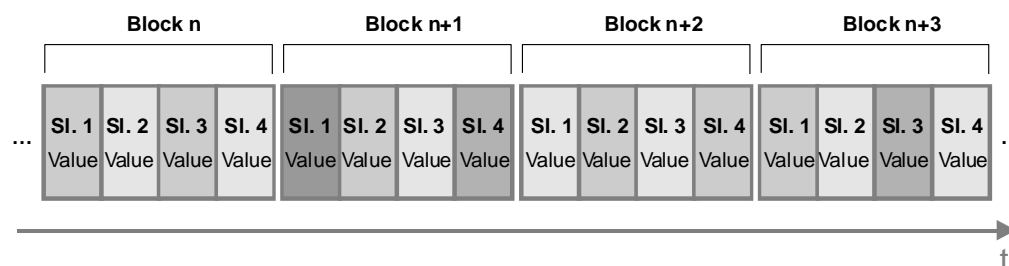


Figure 2: Data stream with block size 4, but with doubled block frequency.

Variation 2:

Doubled
block size

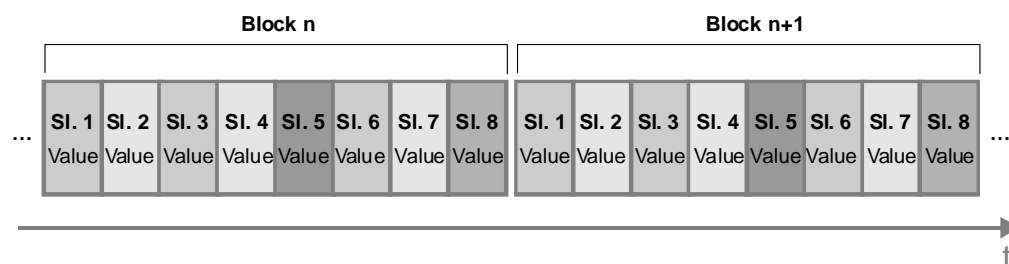


Figure 3: Data stream with block size 8. Time slot length decreases.

In order to reduce the amount of the data transmission from the CAN Driver to LabVIEW, the signal values (single time slots) are passed as a block (see section Read on page 21). The data is available as an array of type double.

2.2.3 Assigning signal values

One data stream per signal

A data stream processes value changes for only one signal. For each configured signal (see chapter [Signal Configuration](#) on page 26) there is an own data stream.

Initialized measuring values

If there are no signal values in the beginning of a measurement (CAN message was not received), the data stream is initialized before with an invalid value (zero).

Assigning

An incoming signal from the CAN bus is integrated into the data stream. The signal value is assigned to the current time slot of the current block. If there is no further signal, the last measured value is used to fill up the unused time slots.



Example: The following drawings depicts different data streams depending on their block size:

Signal changes over two blocks

Figure 4 depicts a changing signal value from the CAN bus. Due to the small block size in this example, two read cycles are necessary.

Block size = 2

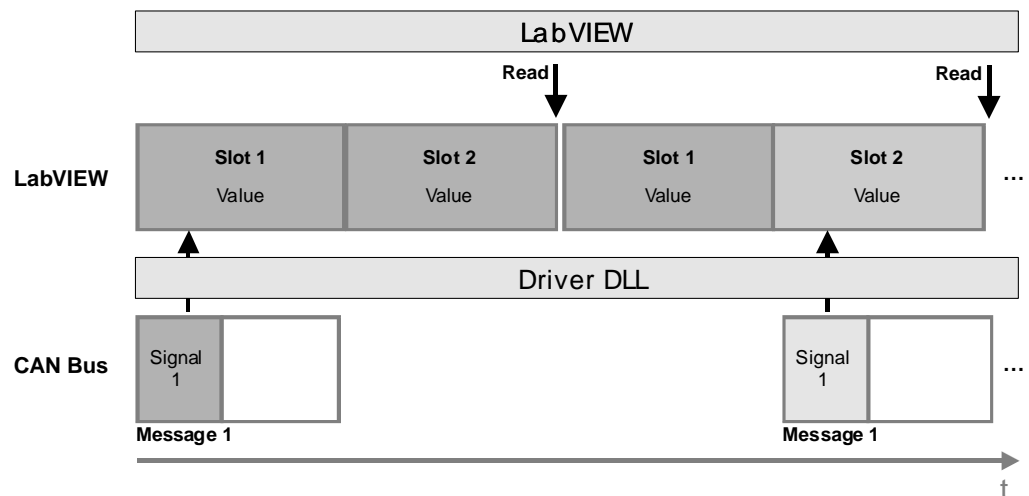


Figure 4: Changing signal value over two blocks.



Info: The less the block size, the bigger the time slot length.

Signal changes over one block

Figure 5 shows the same value change but with fourfold block size and halved block frequency.

Block size = 8

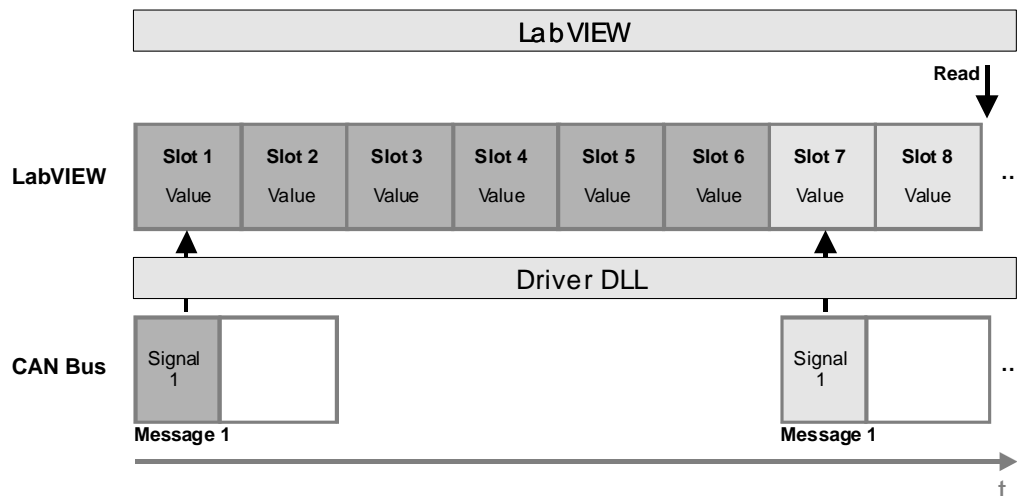


Figure 5: Read changes in one cycle.

2.2.4 Example 1 – Assigning of multiple signals

Description

Figure 6 depicts the situation of two different signals which are sent via the CAN bus. Each signal has its own data stream. Unused time slots are filled up with the last measured value.

Block size = 4

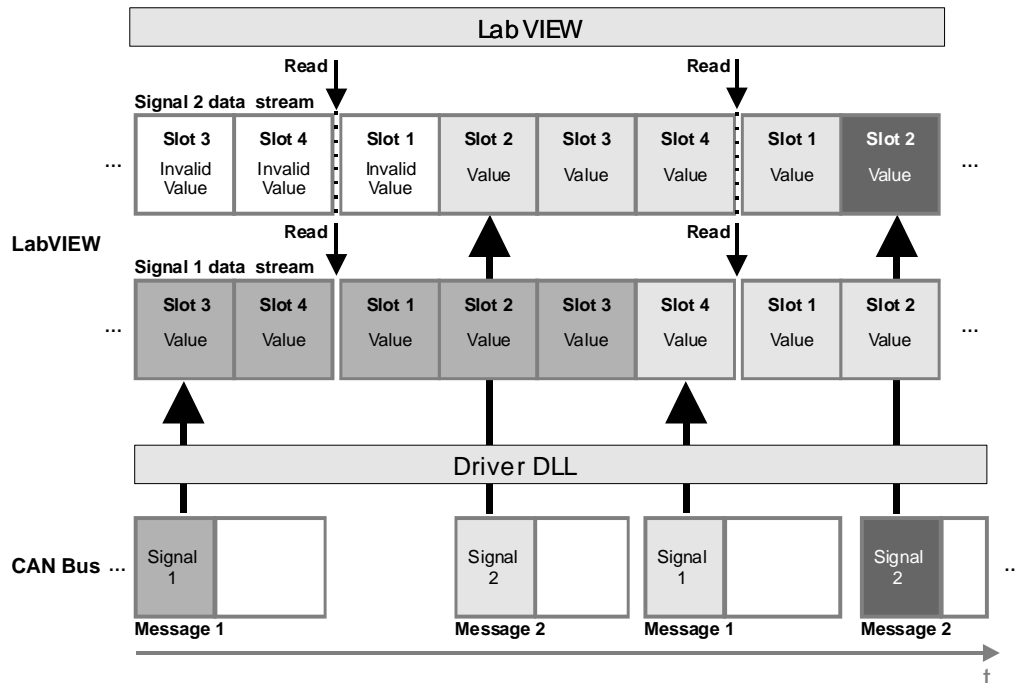


Figure 6: Different signals from two CAN messages.

2.2.5 Example 2 – Small block size and block frequency

Description

Figure 7 demonstrates a smaller block size, whereby the slot time increases. This means, that signal values can be overwritten inside the same time slot if signals are sent too fast. Only the last measured value is valid.

Block size = 2

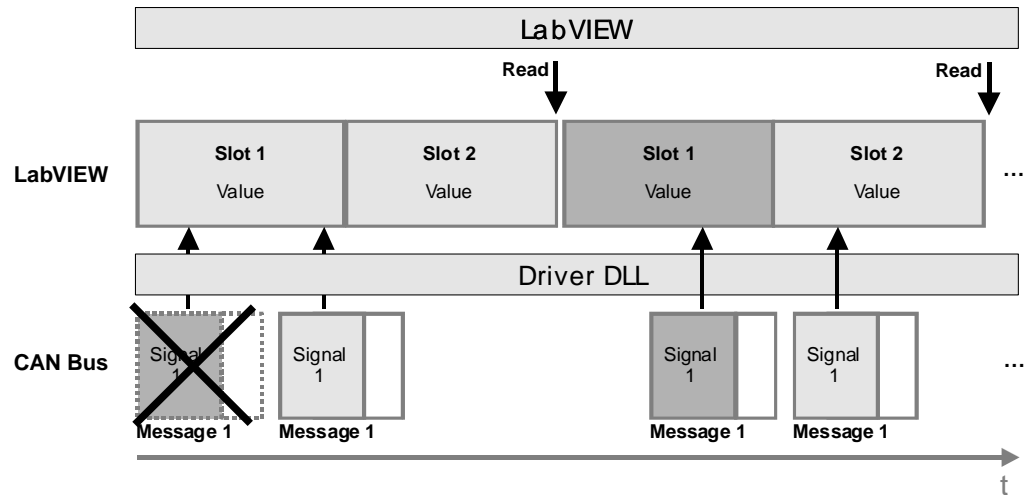


Figure 7: Too small block size. Values are overwritten.

2.2.6 Example 3 – Small block size and high block frequency

Description

A bad dimensioned block size and block frequency is shown in Figure 8. Due to the high block frequency a higher system load could arise, because more read cycles are required for all possible value changes. In the following example, five additional read cycles are called, though no value change is available.

High system load

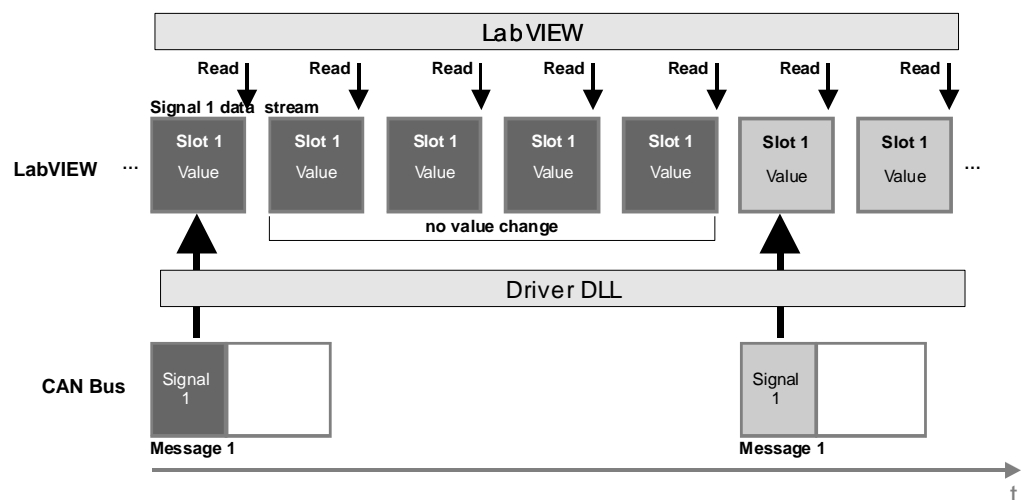


Figure 8: Needless system load by read cycles.

Optimization

In order to reduce the read cycles, the block size is set higher and the block frequency is set lower. In this example, the value change can be read in two read cycles (which means a third of the previous system load).

Balanced solution

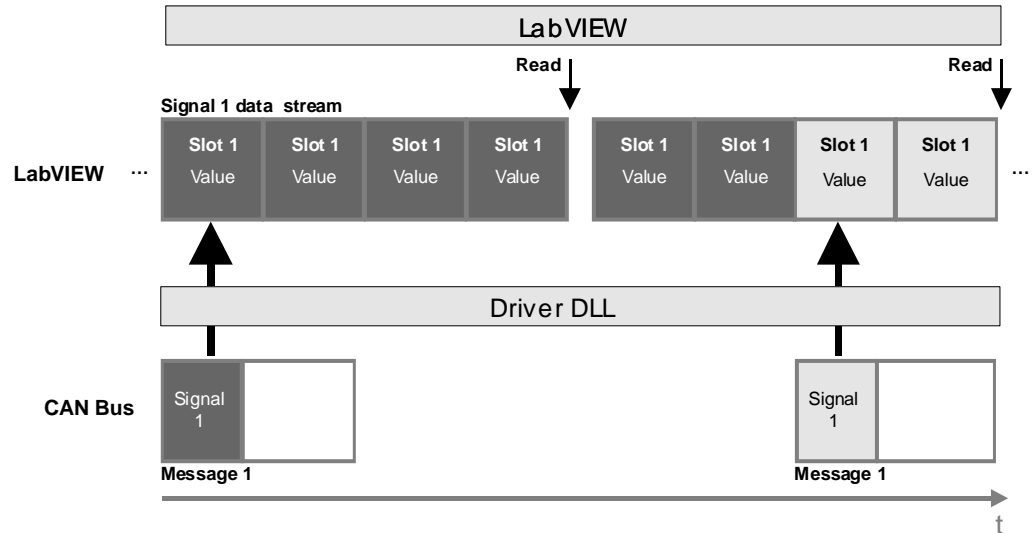


Figure 9: Alternative solution.

2.2.7 Example 4 – Dimensioning block size and block frequency

Calculation tips

- The base for calculation is the fastest signal/CAN message on the bus.
- The sampling frequency (block size * block frequency) is chosen three times faster than the fastest signal.



Example: An ECU transmits the actual outdoor temperature ten times per second (= 10 Hz). In order to sample all value changes, the sampling is done three times faster (= 30 Hz).

The sampling frequency is *block size * block frequency*.

Possible settings:

Block size = 1, Block frequency = 30 Hz or

Block size = 3, Block frequency = 10 Hz



Example: A signal in a network is transmitted 100 times per second (= 100 Hz). The sampling is done 300 times per second (= 300 Hz).

Possible settings:

Block size = 30, Block frequency = 10 Hz or

Block size = 15, Block frequency = 20 Hz or

Block size = 10, Block frequency = 30 Hz or

Block size = 6, Block frequency = 50 Hz.

3 Vector Virtual Instruments (VIs)

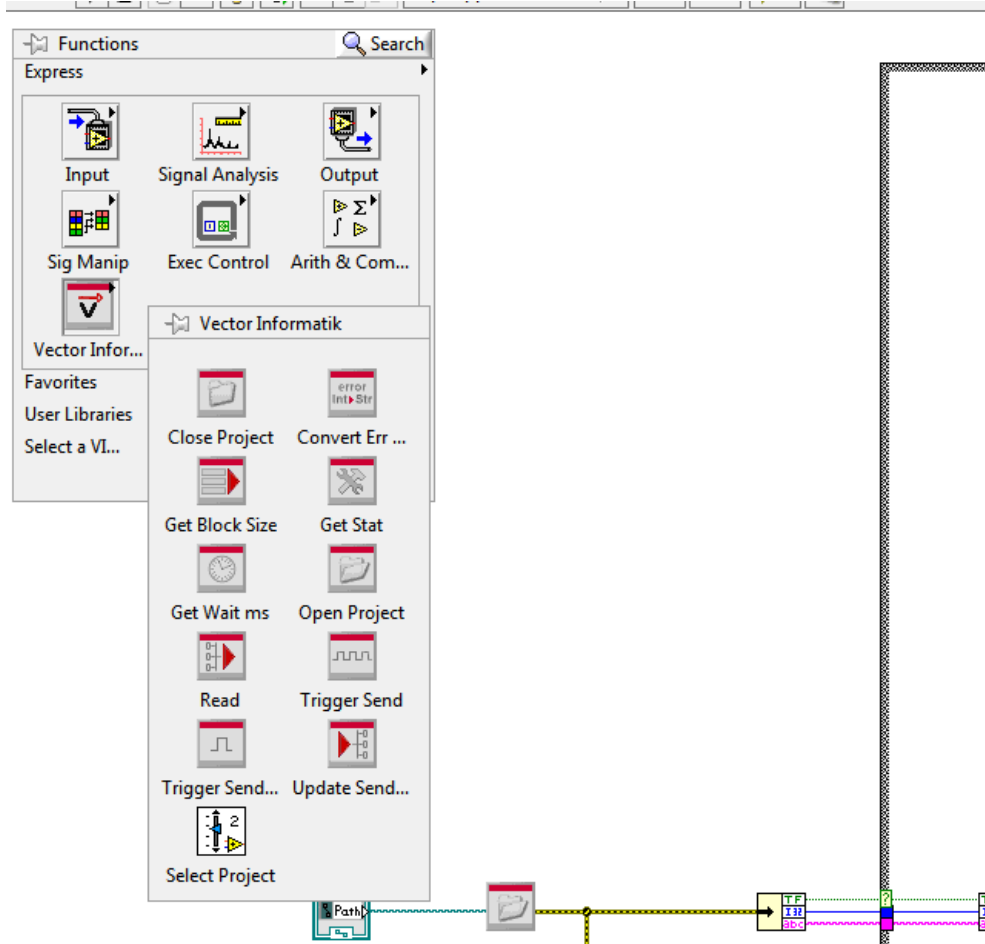
In this chapter you find the following information:

3.1	Access in LabVIEW	page 18
	Open Project	
	Close Project	
	Update Send Buffer	
	Trigger Send	
	Trigger Send Single	
	Get Block Size	
	Get Wait MS	
	Read	
	Error Code to String	
	Get Status	

3.1 Access in LabVIEW

User Libraries

After installation of the Vector CAN Driver, the Vector VI's can be used in the block diagram of LabVIEW. Click functions palette **Express | Vector Informatik**.



3.1.1 Open Project

Icon



Description

Opens a Vector LabVIEW project (.vlv).

Vector LabVIEW projects are created with the the project editor **Vector LabVIEW Configuration tool**. Further information can be found in chapter **Signal Configuration** on page 26.

Inputs

- ➔ **Project File**
Path to a Vector LabVIEW project (.vlv).

Outputs

- ➔ **Error**
Output error code (see section **Error Codes** on page 34).

3.1.2 Close Project

Icon



Description

Closes the opened Vector LabVIEW project.

Inputs

→ **Error**
Input error code (see section [Error Codes](#) on page 34).

Outputs

→ **Error**
Output error code (see section [Error Codes](#) on page 34).

3.1.3 Update Send Buffer

Icon



Description

Writes the value into the send buffer of the selected CAN signal. Each signal has its own send buffer; for this, the signal has to be defined as **OUT** or **OUT Single** (see chapter [Signal Configuration](#) on page 26).

Update Send Buffer can be used in a while-loop for example, to update values in the send buffer cyclically (see example `SendReceiveLoop.vi`).

Inputs

- **Alias**
Selected signal from Vector LabVIEW project (its send buffer to be updated). Aliases can be configured for each signal (see chapter [Signal Configuration](#) on page 26).
- **Error**
Input error code (see section [Error Codes](#) on page 34).
- **Value (double)**
Value to be written.

Outputs

→ **Error**
Output error code (see section [Error Codes](#) on page 34).

3.1.4 Trigger Send

Icon



Description

Builds **several** CAN messages from all **OUT** marked (not Out Single!) send buffers as defined in the data base. For further information see chapter [Update Send Buffer](#) on page 19.

Trigger Send can be used in a while-loop for example to transmit all values from OUT signals cyclically to the CAN bus.

Inputs

- **Error**
Input error code (see section [Error Codes](#) on page 34).

Outputs

- **Error**
Output error code (see section [Error Codes](#) on page 34).

3.1.5 Trigger Send Single

Icon



Description

Builds only **one** CAN message from the **selected send buffer** (see chapter [Update Send Buffer](#) on page 19) as defined in the data base. The signal has to be marked as **Out Single**.

Trigger Send Single can be used for example to send a selected signal on the CAN bus, if a certain condition is fulfilled.

Inputs

- **Alias**
Selected signal from Vector LabVIEW project (see section [Signal Configuration](#) on page 26).

Outputs

- **Error**
Output error code (see section [Error Codes](#) on page 34).

3.1.6 Get Block Size

Icon



Description

This is just for user's information and reads the block size setting in the Vector LabVIEW project.

Inputs

- **Error**
Input error code (see section [Error Codes](#) on page 34).

Outputs

- **Blocksize**
Numerical value of block size.

3.1.7 Get Wait MS

Icon



Description

Calculates the waiting time from block size and block frequency, which can be used in while-loops, e.g. in LabVIEW function "Wait Until Next ms Multiple".

Outputs

- **WaitMsec**
Numerical value in milli seconds.

3.1.8 Read

Icon



Description

Triggers a cycle where the actual filled block is read. The content of the block (all time slots) is available as a one dimensional array of type double. Detailed information can be found in chapter [Basics](#) on page 11.

Inputs

- **Alias**
Selected signal from Vector LabVIEW project (see section [Signal Configuration](#) on page 26).
- **Error**
Input error code (see section [Error Codes](#) on page 34).

Outputs

- **Error**
Output error code (see section [Error Codes](#) on page 34).
- **Value (double)**
Output of received signal values (array).

3.1.9 Error Code to String

Icon



Description

Converts a numerical error code of a Vector VI into clear text.

Inputs

- **Error**
Input error code (see section [Error Codes](#) on page 34).

Outputs

- **Error**
Output error code in text form.

3.1.10 Get Status

Icon



Description

For bus control. The following status can be obtained depending on the input values (Selector, SubSelector):

→ **External overrun (Value 0)**

Selector = 0,
SubSelector = x

→ **Internal overrun (Value 1)**

Selector = 0,
SubSelector = x

→ **Number of received CAN messages (Value 0)**

Selector = 1 und
SubSelector = 1..16 (application channel), 0 (all application channels)

→ **Number of received error frames (Value 1)**

Selector = 1 und
SubSelector = 1..16 (application channel), 0 (all application channels)

x = dont' care

Inputs

→ **Selector**

Selects the function in the VI, see description above.

→ **SubSelector**

SubSelector depending on selected function, see description above.

Outputs

→ **Value 0**

Output value depending on selected function, see description above.

→ **Value 1**

Output value depending on selected function, see description above.

→ **Error**

Output error code (see section **Error Codes** on page 34).

4 Vector LabVIEW Configuration Tool

In this chapter you find the following information:

4.1	Short Description	page 24
4.2	Tree View	page 24
	General	
	CAN Channels	
	Signal Configuration	

4.1 Short Description

Project editor

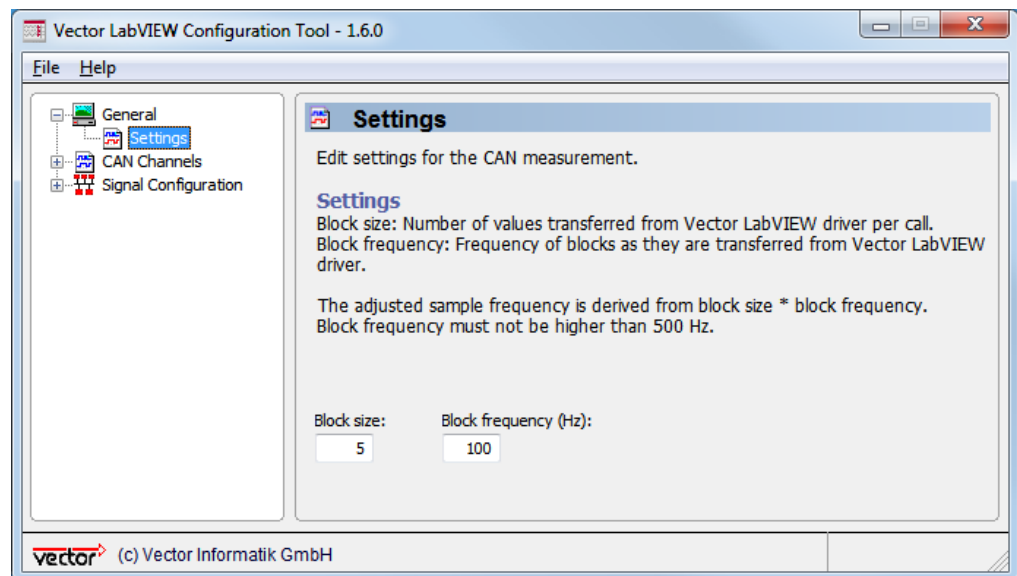
Vector LabVIEW Configuration is a project editor which allows the creation of Vector LabVIEW projects. The created project configuration is opened in LabVIEW by the Vector VI **Open Project**.

Execute the project editor by **Start | Programs | Vector CAN Driver for LabVIEW | Vector LabVIEW Configuration Tool**.

4.2 Tree View

4.2.1 General

Settings



- **Block size**
Number of samples per block (time slots). Each time slot can hold a signal value.
- **Block frequency (Hz)**
Block frequency in Hz.



Info: The sampling rate arises from

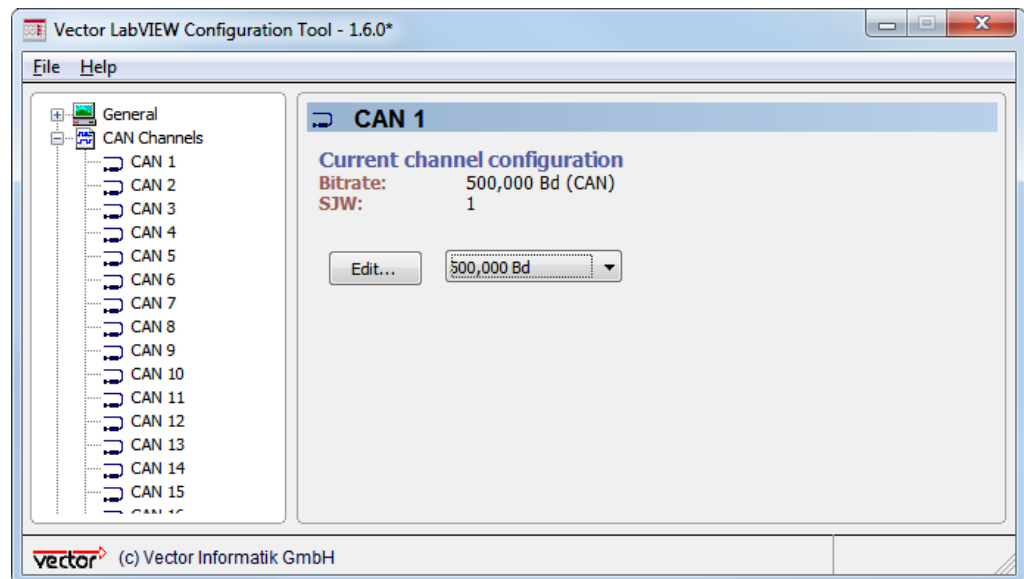
$$\text{Block size} * \text{Block frequency}$$

Detailed information about block size and block frequency can be found in chapter Basics on page 11.

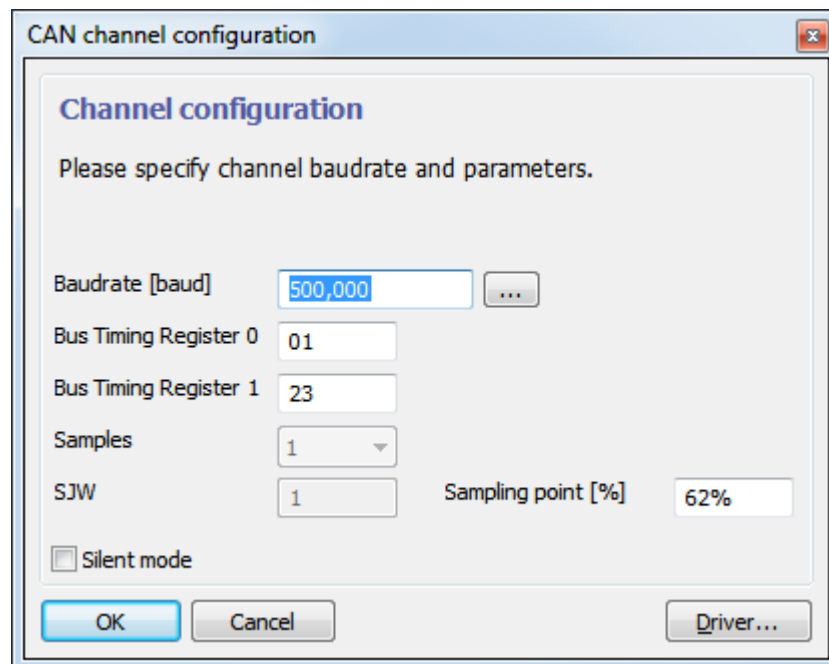
Calculation tips are described in chapter Example 4 – Dimensioning block size and block frequency on page 16.

4.2.2 CAN Channels

CAN 1...CAN 16



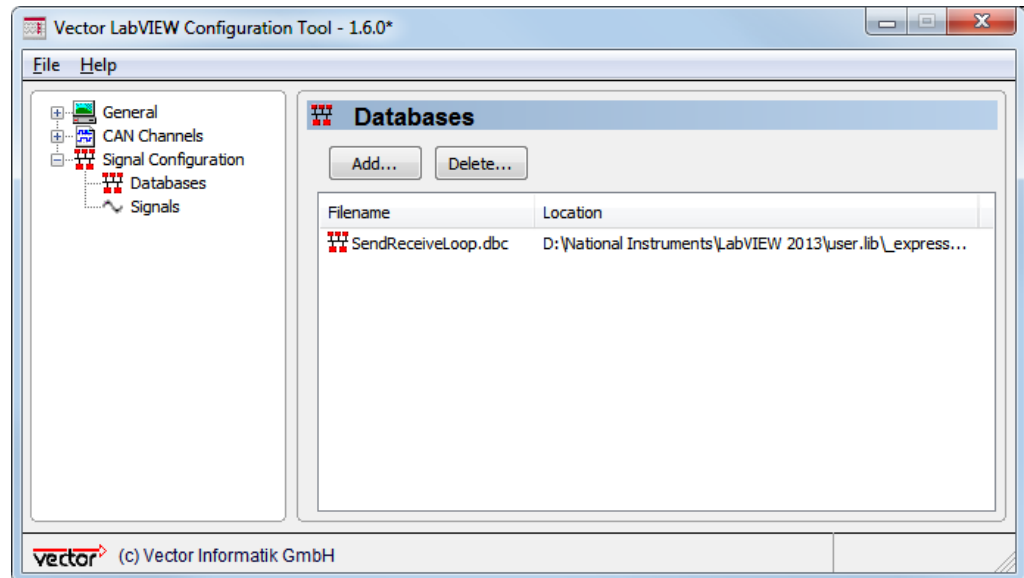
The Vector CAN Driver supports up to sixteen CAN channels, which can be configured in section **CAN Channels**. Standard baud rates are available in a combo box. With the **[Edit...]** button the CAN channel configuration dialog can be opened to enter own baud rates.



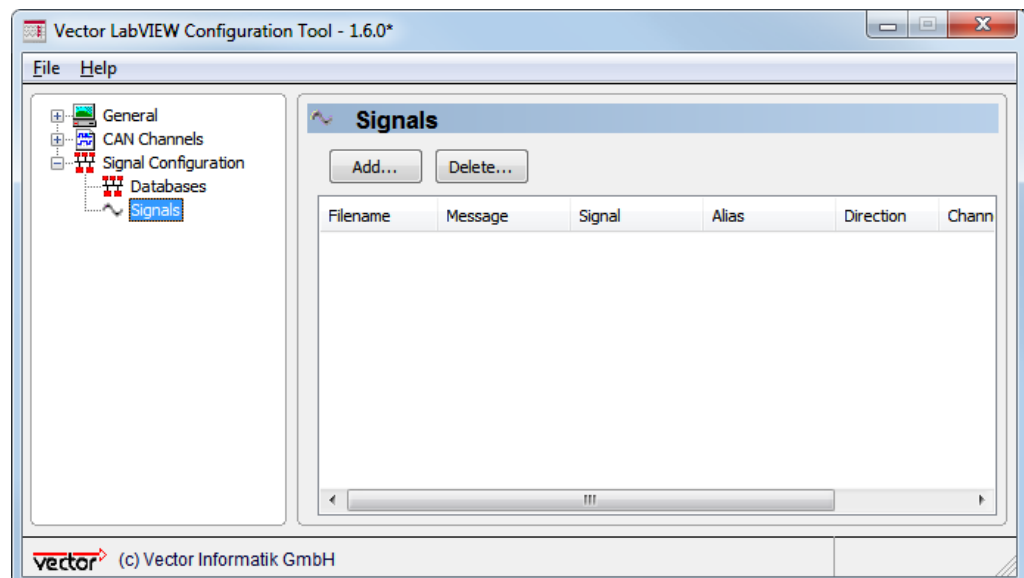
- **Baudrate [baud]**
Baudrate of the selected CAN channel.
- **Bus Timing Register 0/1**
Expert settings for baudrates via timing register. Affects the sampling point. Further information can be found in the datasheet of CAN Controller SJA1000.
- **Sampling point [%]**
Expert setting. Ratio between bit length at sampling time and whole bit time in percent. Further information can be found in the datasheet of CAN Controller SJA1000.

Silent mode

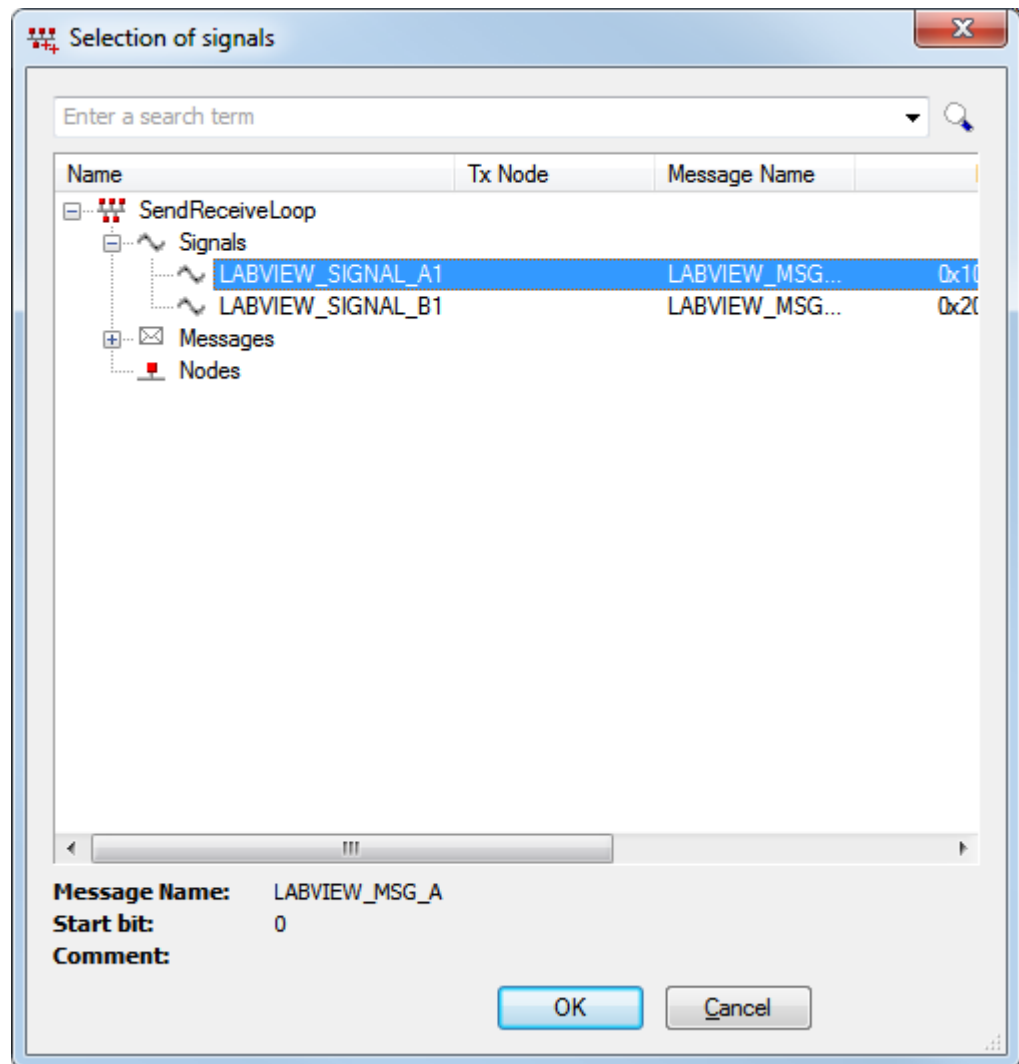
This switch allows the analysis of received CAN messages, without disturbing the CAN bus (e.g. by acknowledge).

4.2.3 Signal Configuration**Databases**

The signal configuration enables adding and removing of CAN signals from existing databases (*.dbc). First, add one or more databases in subsection **Databases** by clicking the **[Add...]** button. Afterwards, one or more signals can be added in subsection **Signals**, which should be accessible in LabVIEW.

Signals

Click the **[Add...]** button to open the selection of signals dialog.



The top node in the tree view represents the database. The signals and messages are displayed below.

Signals and messages can be found in a more complex database with the search function. Search strings can be entered the input box above (lens icon).

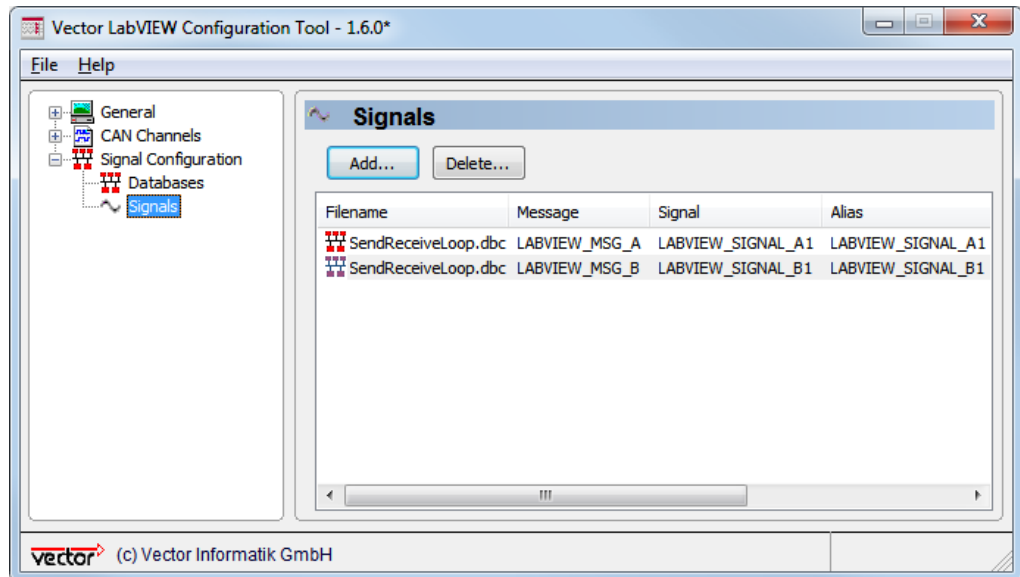


Info: The asterisk '*' will also be used as search criteria. In order to find similar signals only a text fragment is needed, e.g. "labview_".

Select signals

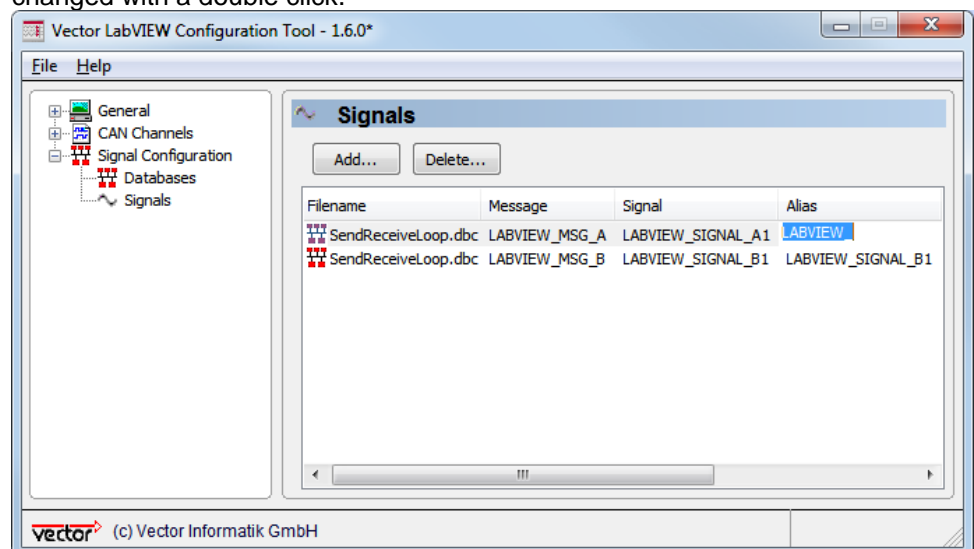
The signal selection is done in the right part of the window and completed with the **[OK]** button.

Added signals



The signal list contains the following columns:

- ➔ **Filename**
Name of the database.
- ➔ **Message**
Displays the message containing the signal.
- ➔ **Signal**
Displays the signal name.
- ➔ **Alias**
Signal reference for Vector VI's in LabVIEW (see Inputs „Alias“). The alias can be changed with a double click.



→ **Direction**

Defines a signal for input or output. The direction can be changed with a double click.

In

The signal is received in LabVIEW (Vector VI **Read**).

Out

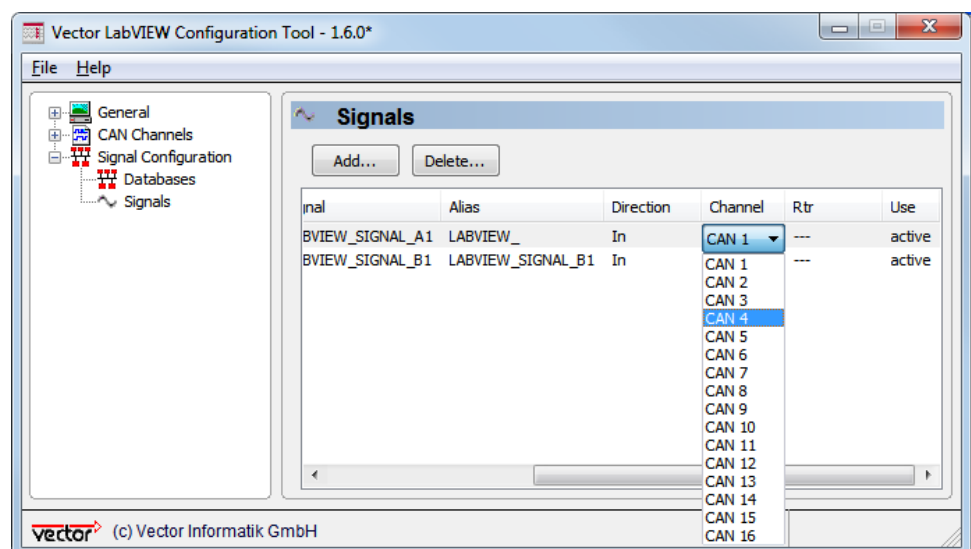
This signal and all other OUT marked signals are transmitted together when Vector VI **SendTrigger** is called.

Out Single

Only the selected signal is transmitted (Vector VI **SendTriggerSingle**).

→ **Channel**

CAN channel on which the signals should be received or sent. This channel is related to one (of maximum sixteen) channels in the Vector Hardware Configuration in section **Application | LabVIEW**.



→ **Rtr**

Remote transmission. This switch affects all signals which belong to the same CAN message. This means, if one signal is set to Rtr, a remote frame is generated on the bus.

→ **Use**

Enables/disables signals to reduce system resources. Each signal has its own data stream. These data streams can be reduced if the database is too large and only few signals are needed.

An unused signal is **not** available in LabVIEW and arise an error message, if it is accessed.

5 Example SendReceiveLoop

In this chapter you find the following information:

5.1	Short Description	page 31
5.2	Running the example	page 31

5.1 Short Description

Content

After installation of the Vector CAN Driver you can find an example in the following sub folder of LabVIEW **...\express\Vector Informatik\examples**. The example consisting of:

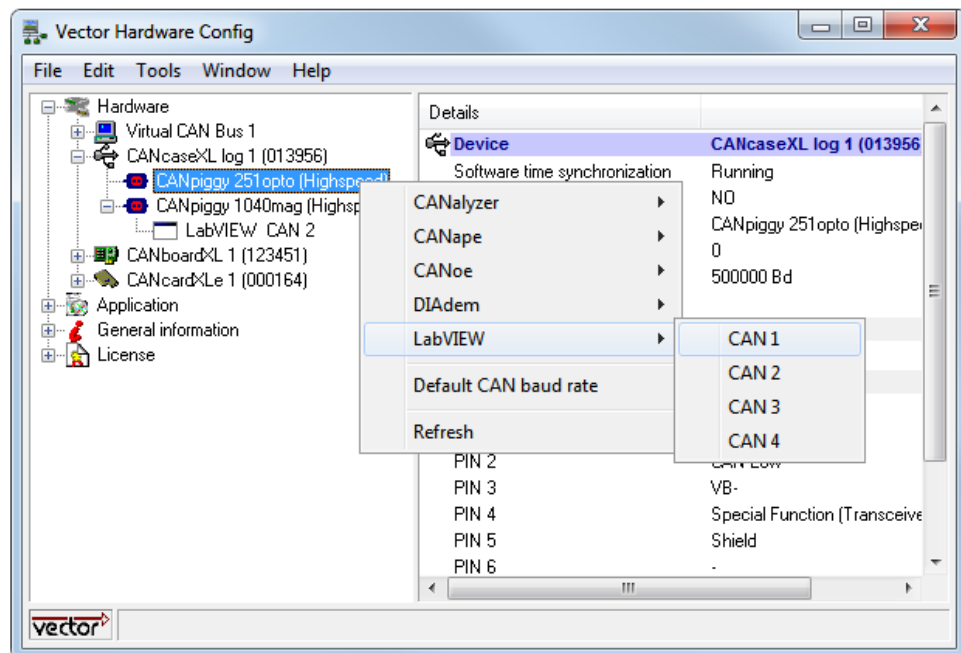
- Database `SendReceiveLoop.dbc` with defined signals
- Vector LabVIEW project `SendReceiveLoop.vlv`
- LabVIEW VI `SendReceiveLoop.vi`

The example is a loop application, where simulated signals are sent from LabVIEW to the first CAN channel and received by the second CAN channel of a previous defined Vector Hardware. It is shown, how signals can be sent cyclically and manually with Vector VIs. For this purpose the database offers two messages each containing one signal.

5.2 Running the example

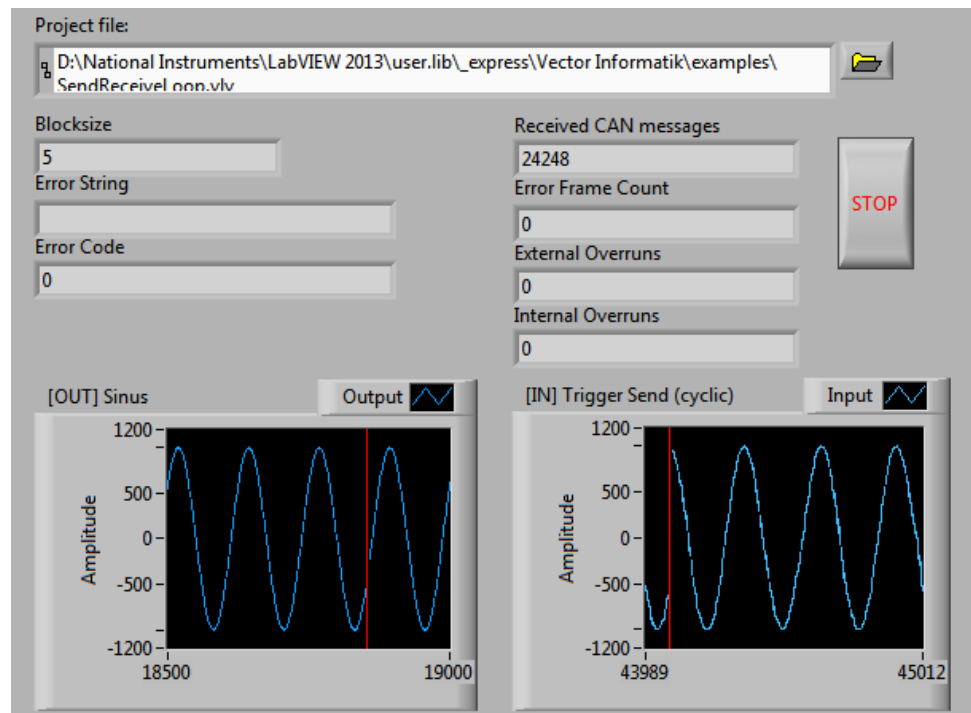


1. Connect both CAN channels of your Vector Hardware with a suitable cable. Ensure that the Vector Hardware is licensed for LabVIEW.
2. Open the Vector Hardware Configuration (**Start | Preferences | System | Vector Hardware**) and assign **LabVIEW CAN1** and **CAN2** to the CAN channels of your Vector Hardware.

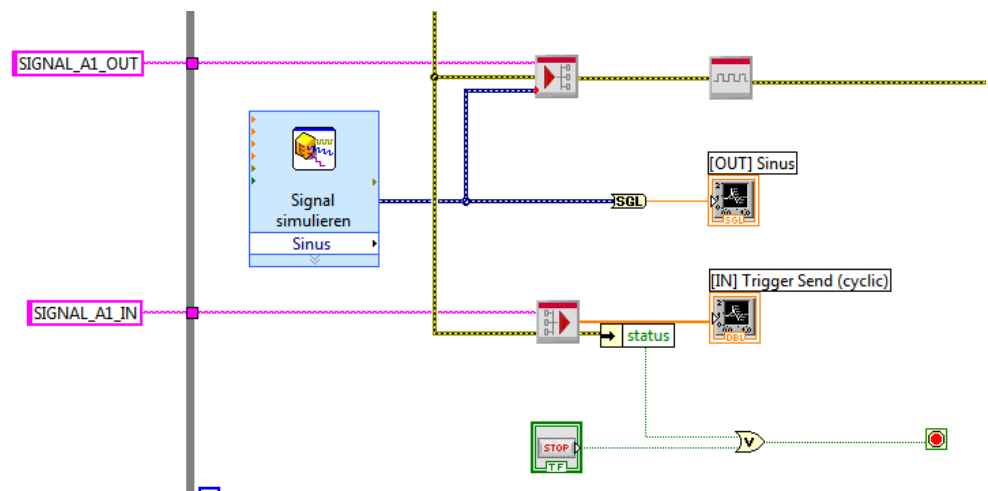


3. Open the VI **SendReceiveLoop.vi** in LabVIEW. You can run this example by clicking **Start | Programs | Vector CAN Driver for LabVIEW**.
4. Ensure that the project path (**Project file**) to the Vector LabVIEW project is correct.
5. Start measurement with **<STRG+R>** or in main menu **Operate | Run**.

Cyclically updates
of multiple signals
(Trigger Send)



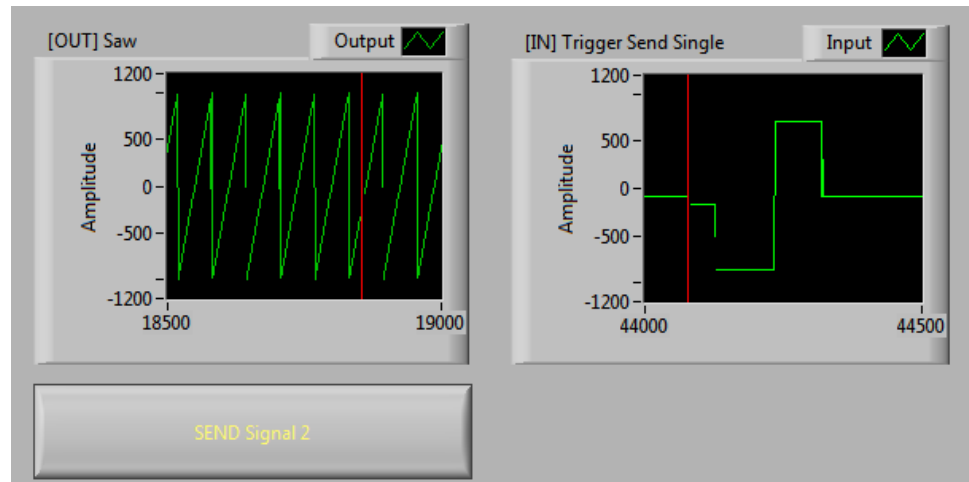
Block scheme



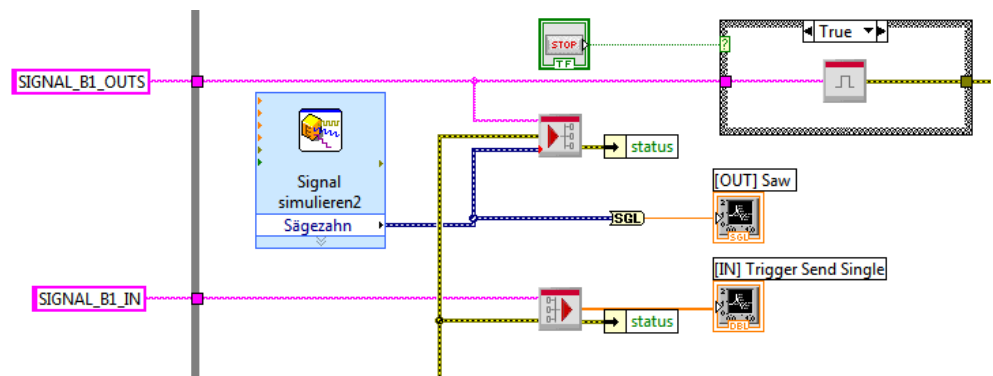
The signal curve **[OUT] Sinus** displays a generated signal in LabVIEW, which is permanently written into the send buffer of the selected signal (Vector VI **Update Send Buffer**). The current values of the send buffer are sent to the CAN bus cyclically by the VI **Trigger Send** in the set frequency and through the defined CAN channel.

[IN] Trigger Send displays the signal curve, which is cyclically received by the second CAN channel.

Event triggered
updates of selected
signal
(Trigger Send Single)



Block scheme



The signal curve **[OUT] Saw** displays a generated signal in LabVIEW, which is permanently written into the send buffer of the selected signal (Vector VI **Update Send Buffer**). The current value of the send buffer is sent to the CAN bus using the VI **Trigger Send Single** and the **[Send Signal 2]** button through the defined CAN channel.

[IN] Trigger Send displays the current value, which is received by the second CAN channel.

6. Stop the measurement by clicking the **[Stop]** button, to close and reset Vector LabVIEW Driver.

6 Error Codes

In this chapter you find the following information:

6.1	Overview
-----	----------

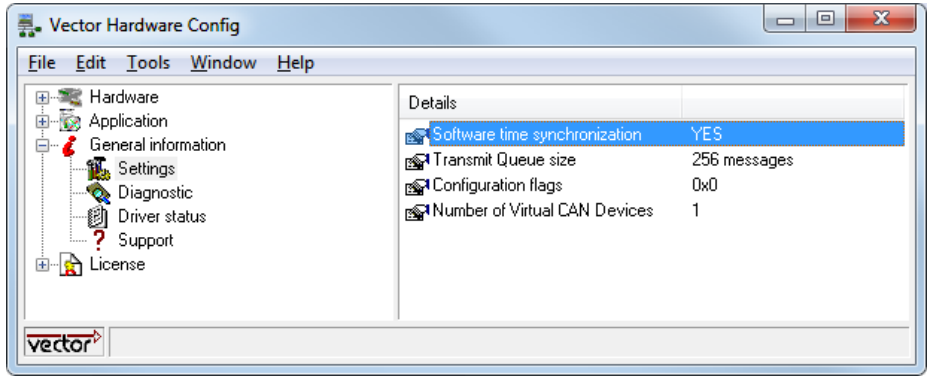
page 35

6.1 Overview



Info: For clear text output on runtime use VI **Error Code To String** (see section **Error Code to String** on page 21).

Code	Meaning
6	DRV_ERR_DIRECTION_NOT_SUPPORTED
9	DRV_ERR_HW_MAPPING
10	DRV_ERR_ACCESS_PROJECTFILE
11	DRV_ERR_LOAD_PROJECTFILE_G
12	DRV_ERR_LOAD_PROJECTFILE_P
13	DRV_ERR_SIGNAL_CONF_LINE_TOO_LONG
14	DRV_ERR_SIGNAL_CONF_PATH2LONG
15	DRV_ERR_SIGNAL_CONF_CANDBACCESS
16	DRV_ERR_SIGNAL_CONF_2MANYCANDB
17	DRV_ERR_SIGNAL_CONF_ADDDBTOGUILIB
18	DRV_ERR_SPLITPATH
19	DRV_ERR_READ_SIGSUBSTRING
20	DRV_ERR_PARAM2LONG
21	DRV_ERR_INVALID_DIRECTION
22	DRV_ERR_INVALID_APPCHANNEL
23	DRV_ERR_APPCHANNEL_NOT_CONFIGURED
24	DRV_ERR_BUS_PARAMETERES_MISSING
25	DRV_ERR_RESOLVE_SIGNAL
26	DRV_ERR_XLGETAPPCONFIG
27	DRV_ERR_BUSCHANNEL_CAPABILITIES
28	DRV_ERR_NO_SIGNAL_CONFIGURED
29	DRV_ERR_OPEN_CAN_PORT
30	DRV_ERR_OPEN_LIN_PORT
31	DRV_ERR_INITACCESS_CAN_PORT
32	DRV_ERR_INITACCESS_LIN_PORT
33	DRV_ERR_SET_CAN_PARAMS
34	DRV_ERR_SET_LIN_PARAMS
35	DRV_ERR_SET_NOTIFICATION
36	DRV_ERR_ACTIVATE_CHANNEL
37	DRV_ERR_SIGNAL_READ_NOT_FOUND
38	DRV_ERR_SIGNAL_BUFFER_NOT_ALLOCATED
39	DRV_ERR_DUPLICATE_OUT_SIGNAL

Code	Meaning
40	DRV_ERR_LIN_OUT_UNSUPPORTED
41	DRV_ERR_OUT_SIG_NOTFOUND
42	DRV_ERR_COLLECT_SIGNAL_VALUES
43	DRV_ERR_SENDMSG
44	DRV_ERR_OUT_MSG_NOTFOUND
45	DRV_ERR_OUT_MSG_NOTSINGLE
46	DRV_ERR_NO_SIGNALS
47	DRV_ERR_DUPLICATE_SIGNR
48	DRV_ERR_UNKNOWN
50	DRV_ERR_ADD_ACCRANGE
51	DRV_ERR_SET_ACCEPTANCE
52	DRV_ERR_NO_LICENSE
53	DRV_ERR_HW_NOT_SUPPORTED
54	<p>DRV_ERR_CHANNEL_CAPABILITIES</p> <p>Note:</p> <p>Error 54 arises mostly due to the switched off software time synchronization. Click Start Settings Control Panel Vector Hardware and open section General information Settings and switch Software time synchronization to YES.</p> 
55	DRV_ERR_SIGNALALIAS_NOTFOUND
56	DRV_ERR_PROJECT_NOT_OPEN
57	DRV_ERR_SIGNALALIAS_NOTFOUND
58	DRV_ERR_UNKNOWN_PARAM
59	DRV_ERR_BFREQ_LIMIT
60	DRV_ERR_CONFIG_FILE_VERSION
99	DRV_ERR_IN_ERRORCONVERSION
100	DRV_ERR_NO_MEM

Get more Information!

Visit our Website for:

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

www.vector-worldwide.com