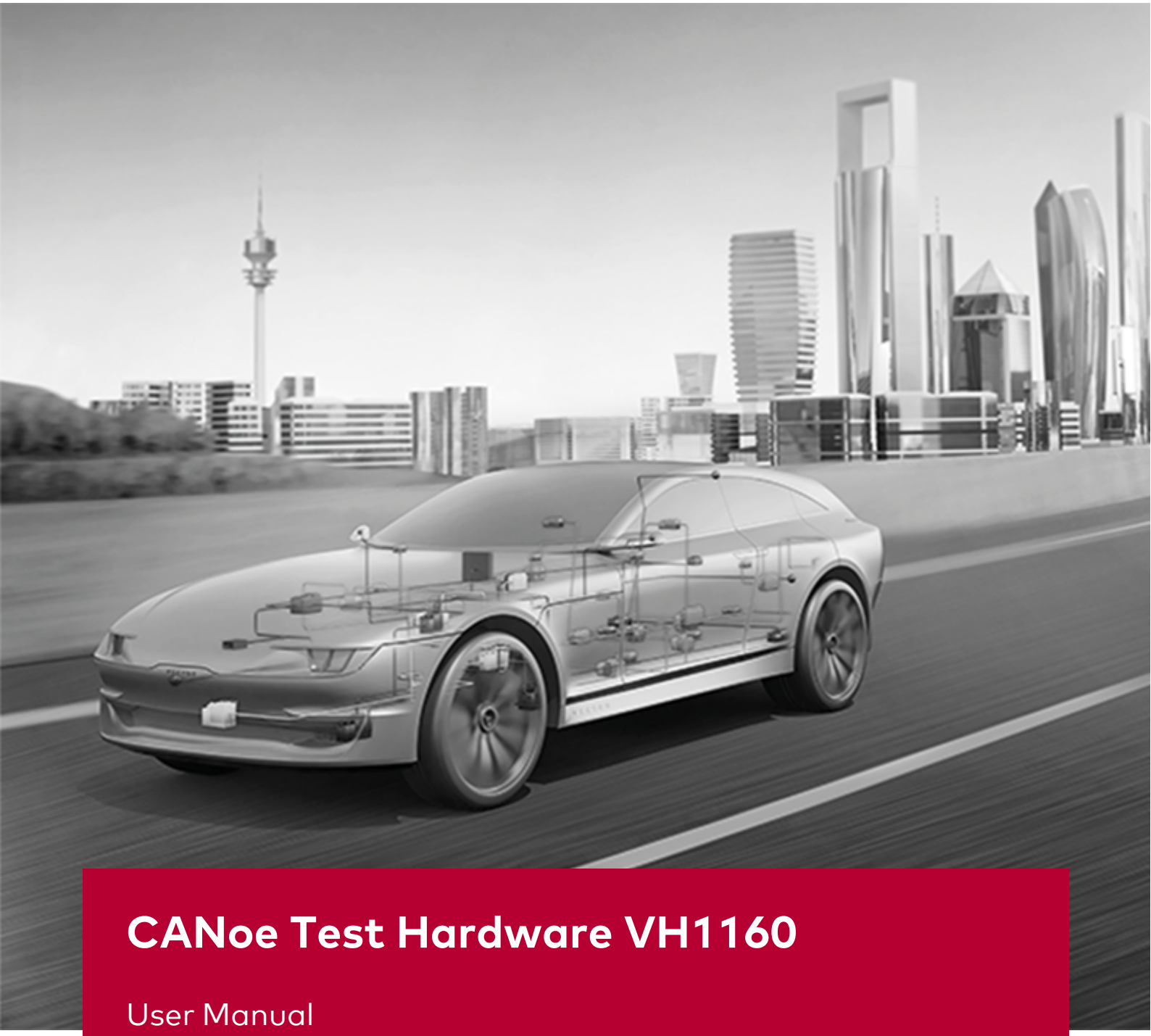
VECTOR



CANoe Test Hardware VH1160

User Manual

Version 1.3
English

Imprint

Vector Informatik GmbH
Ingersheimer Straße 24
D-70499 Stuttgart

Table of Contents

1 Introduction

In this chapter you find the following information:

1.1 Product Overview

The CANoe test hardware VH1160 is a USB-hardware for automating ECU conformance tests for CAN and LIN. It is typically used by CANoe-based test implementations to automatically hardware reset the ECU before each test case. Further applications include the programmed adjustment of the ECU's supply voltage and the automated detection of sleep and wake modes by measuring the ECU's current consumption.

Additionally the VH1160 supports stress features to produce a ground offset or to limit the recessive level of the LIN line. It is also possible to short circuit or interrupt CAN and LIN lines.

The VH1160 provides IO lines which allow to interface with external devices.

Currently this hardware is used by the Vector CANoe test package VAG for CAN high-speed and by the LIN conformance tests implementation provided with CANoe .LIN.

Requirements

CANoe version 8.1 SP3

Scope of Delivery

- USB-hardware VH1160 (external power supply not included)
- VH1160 installation manual
- Installation CD
- USB-cable (2 m)
- Y-cable

Optional Articles

- Y-cable: Additional Y-cables can be ordered if they are lost or defective
- Mean Well power supply GST120A24-R7B (can be attached to the VH1160's four pin connector)

1.2 Important Notes

1.2.1 Safety Instructions and Hazard Warnings

Caution!

In order to avoid personal injuries and damage to property, you have to read and understand the following safety instructions and hazard warnings prior to installation and use of this interface. Keep this documentation (manual) always near the interface.

1.2.2 Proper Use and Intended Purpose

Caution!

The interface is designed for analyzing, controlling and otherwise influencing control systems and electronic control units. This includes, inter alia, bus systems like CAN, LIN, K-Line, MOST, FlexRay, Ethernet, BroadR-Reach and/or ARINC 429.

The interface may only be operated in a closed state. In particular, printed circuits must not be visible. The interface may only be operated (i) according to the instructions and descriptions of this manual; (ii) with the electric power supply designed for the interface, e.g. USB-powered power supply; and (iii) with accessories manufactured or approved by Vector.

The interface is exclusively designed for use by skilled personnel as its operation may result in serious personal injuries and damage to property. Therefore, only those persons may operate the interface who (i) have understood the possible effects of the actions which may be caused by the interface; (ii) are specifically trained in the handling with the interface, bus systems and the system intended to be influenced; and (iii) have sufficient experience in using the interface safely.

The knowledge necessary for the operation of the interface can be acquired in work-shops and internal or external seminars offered by Vector. Additional and interface specific information, such as „Known Issues“, are available in the „Vector KnowledgeBase“ on Vector´s website at www.vector.com. Please consult the „Vector KnowledgeBase“ for updated information prior to the operation of the interface.

1.2.3 Hazards

Caution!

The interface may control and/or otherwise influence the behavior of control systems and electronic control units. Serious hazards for life, body and property may arise, in particular, without limitation, by interventions in safety relevant systems (e.g. by deactivating or otherwise manipulating the engine management, steering, airbag and/or braking system) and/or if the interface is operated in public areas (e.g. public traffic, airspace). Therefore, you must always ensure that the interface is used in a safe manner. This includes, inter alia, the ability to put the system in which the interface is used into a safe state at any time (e.g. by „emergency shutdown“), in particular, without limitation, in the event of errors or hazards.

Comply with all safety standards and public regulations which are relevant for the operation of

the system. Before you operate the system in public areas, it should be tested on a site which is not accessible to the public and specifically prepared for performing test drives in order to reduce hazards.

1.2.4 Disclaimer

Caution!

Claims based on defects and liability claims against Vector are excluded to the extent damages or errors are caused by improper use of the interface or use not according to its intended purpose. The same applies to damages or errors arising from insufficient training or lack of experience of personnel using the interface.

1.3 About This User Manual

1.3.1 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

1.3.2 Certification

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

1.3.3 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.

1.3.4 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.

1.3.5 Trademarks

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

2 Installation

In this chapter you find the following information:

2.1 Hardware Requirements

2.1.1 Connections

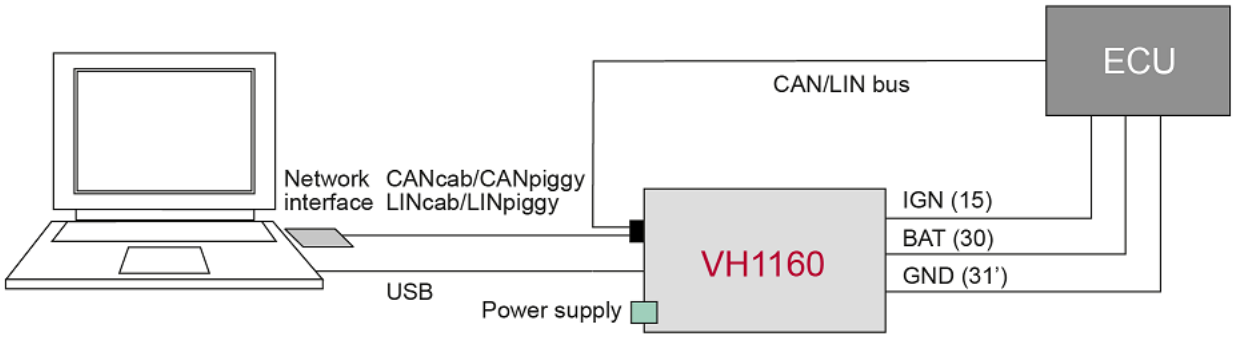


Figure 1: Hardware connections

- VH1160 shall be connected to PC or laptop using the USB cable provided.
- An external power supply (12-28 V) is required for the VH1160 and is not included in the scope of delivery. Recommended external power supply see chapter 1.1 Product Overview.
- The external power supply can be connected to the VH1160 via the 4 mm connectors or the four pin connector.
- The ECU supply lines shall be connected to the ECU's terminals for ignition (15), battery (30) and ground (31).
- Required bus lines must be connected to the CAN/LIN connector. A CAN line requires a 120 Ohm termination resistor.

2.1.2 Power Supply Requirements

Output Voltage	18 V – 24 V (+-0,3V)
Output Voltage (reduced VH1160 output current)	12 V – 18 V (+- 0,3V)
Output Current	5,0 A

2.2 VH1160 Overview

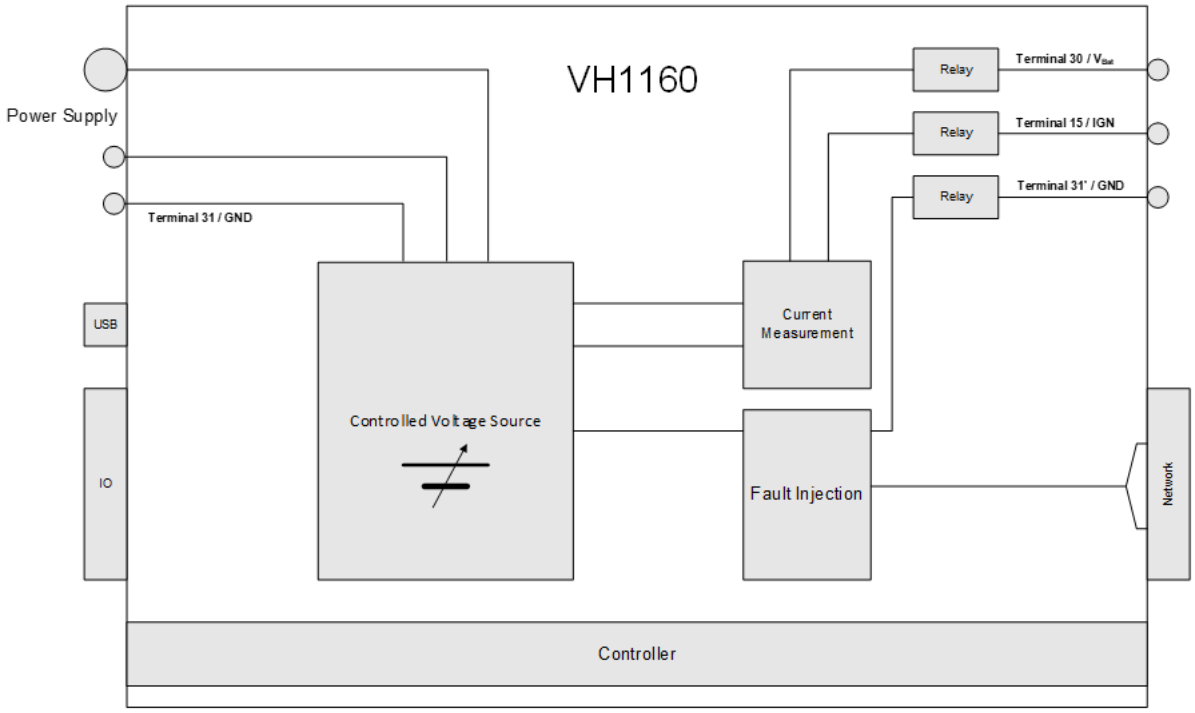


Figure 2: Block Diagram VH1160

- Controller: Control unit of the VH1160.
- Current measurement: Measures the ECU current consumption (e.g. detection of sleep/wake up of an ECU).
- Fault injection: Controls ground offset, recessive level limitation of LIN line and bus line disturbances.
- Relays: Switching the terminals 30,15, 31'.
- CAN/LIN: Connector to attach the bus lines.
- Voltage supply: Programmable voltage supply.

Figure 3: VH1160 terminals and network connector

Figure 4: VH1160 power supply, USB and IO connectors

- Power supply: Four pin connector for recommended power supply. 4 mm connectors for laboratory power supply.
- USB connector: Connection to the PC. An FTDI driver is required.
  Note: The device is referenced as a **USB serial port** in the Windows Device Manager (section COM and LPT connections).
- IO: Digital and analog input and output ports.

Caution!

Neither power supply nor terminal connections must be removed nor connected while the VH1160 operates!

Never supply the VH1160 device on both power supply connectors (four pin connector/ 4 mm connectors)!

2.3 Software Requirements

Operating System	Windows 7, 8.1, 10
CANoe Version	CANoe version 8.1 SP3
FTDI Driver	Version 2.08.30 or higher

2.4 Software Installation

Install

Please ensure that you have administrator access rights for your PC before following the installation steps below:

- Install or update your CANoe version to 8.1 SP3 or higher
- Close CANoe application
- Update VH1160 CANoe driver
- Install or update the FTDI driver
- Connect all hardware as described in chapter 2.1 Hardware Requirements
- After successful installation please connect the VH1160 via USB interface to your PC. Power up your VH1160. Windows should automatically recognize the VH1160.

Uninstall

FTDI and CANoe driver can be uninstalled using the standard Windows mechanism.

Downloads

- VH1160 setup: www.vector.com/vh1160-setup/
- FTDI driver: http://www.ftdichip.com/Drivers/VCP.htm

2.5 Compatibility with VH1100/VH1150

The VH1160 is backward compatible to the VH1100. This means, in configuration dialogs of older CANoe versions (e.g. LIN Conformance Test Module, VAG Test Package) the VH1100 has to be selected although a VH1160 is used as real device.

The VH1160 acts as a direct replacement of the VH1150. CANoe configurations which have been created with the VH1150 can be used with the VH1160 without any change.

2.6 Configuring VH1160

The API (see chapter 4 API) to control the VH1160 is implemented in the **VHDevDLL.dll**. The DLL can be found in the **EXEC32** directory of the CANoe installation. To use the API the VHDevDLL.dll has to be attached to a test module (see figure below).

Note

The VHDevDLL.dll must not be attached to a network node. The VHDevDLL.dll always requires the context of a test module.

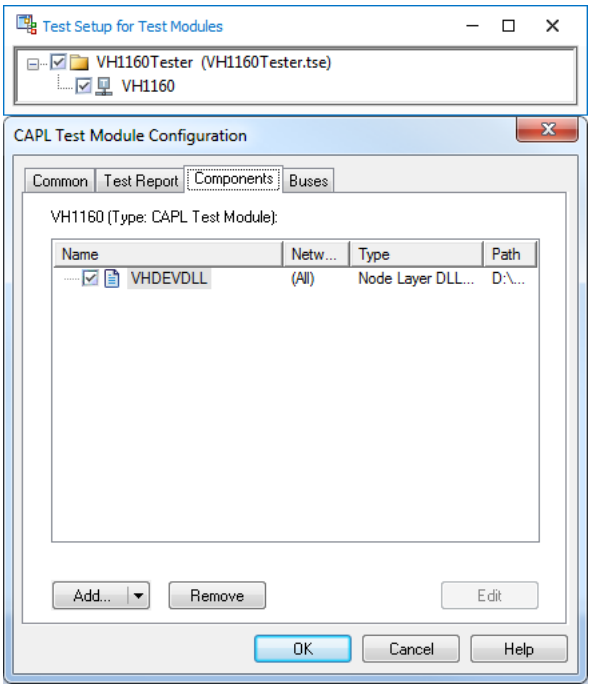


Figure 5: Configuration of VHDevDLL.dll

2.7 Connecting the VH1160 to a VN8900 Device

If the VH1160 is used with a device of the VN8900 interface family it must be connected to the USB host of the VN8900 device.

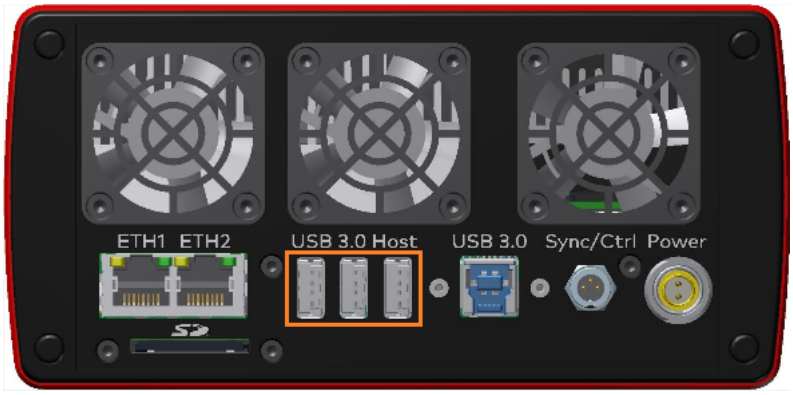


Figure 6: USB host of VN8900

Note

The VH1160 will not work if directly connected to the host PC!

3 Technical Data

In this chapter you find the following information:

3.1 Output Voltage Terminal 15/30

Parameter	Value Range
Voltage range	2 – 28 V
Output current @ input voltage >= 18V	Max 2.5 A
Output	Short-circuit proof
Accuracy output voltage	+- 60 mV
Typical settling time	<= 5 ms

3.2 Measurement of Output Current

Parameter	Value
Measurement location	Supply line to the ECU
Accuracy	+-2% or 0.5 mA

3.3 Relay for Terminals

Parameter	Value Range
Relay	terminal 30 terminal 15 terminal 31’
Typical switching time	7 ms

3.4 Ground Offset for DUT Supply

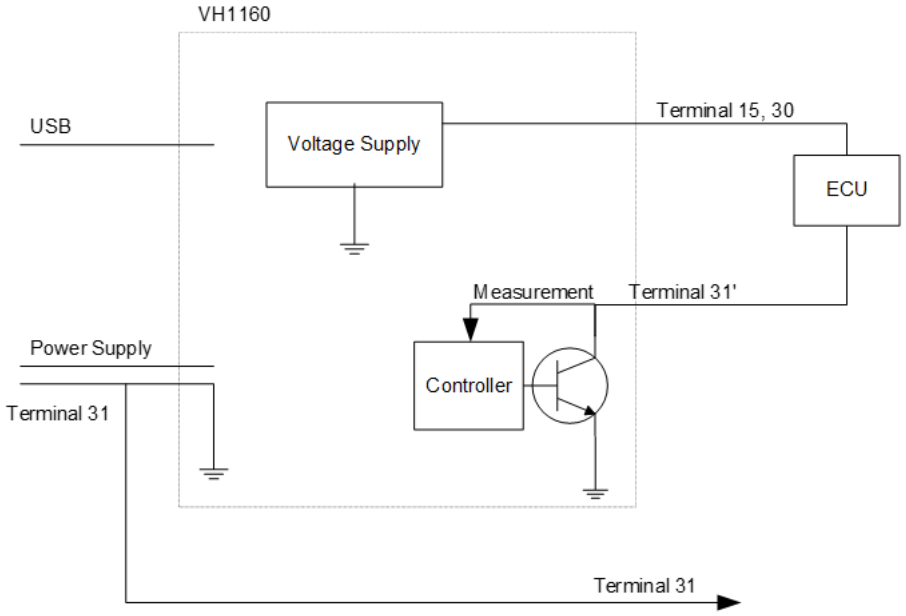


Figure 7: Block diagram ground offset

Parameter	Value Range
Offset Voltage Terminal 31’	0 V – 10 V
Terminal 31	Supply ground
Accuracy	+- 50 mV

3.5 Ground Offset LIN Line

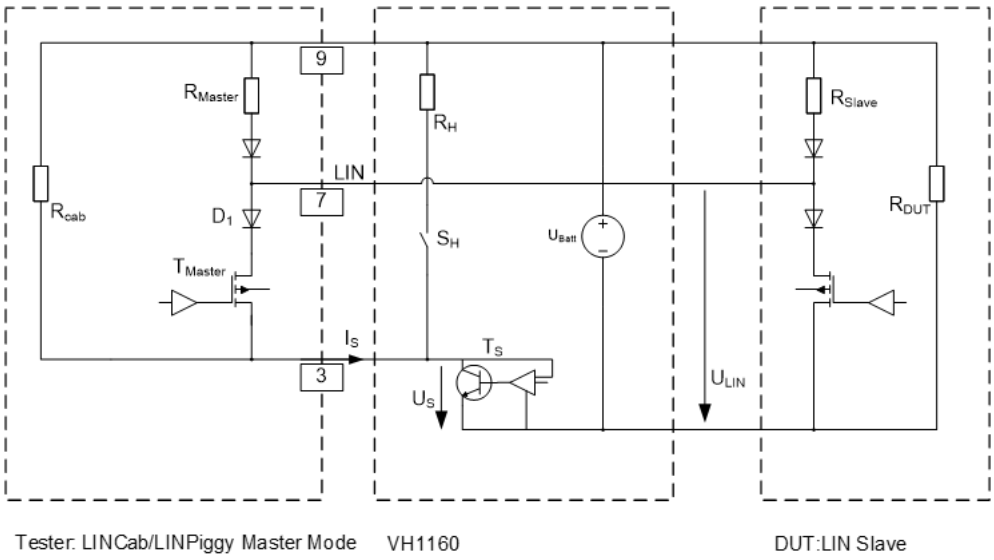


Figure 8: Principle ground offset LIN line

Parameter	Value Range
Offset range V_S	2.5 V – 17.5 V
Operation range V_{Batt}	6.0 V – 28.0 V
Accuracy	+- 50 mV
Settling time	100 ms or faster

3.6 LIN Recessive Level Limit

Parameter	Value Range
V_LIN_Rec_min	3.0 V
V_LIN_Rec_max	28.0 V
V_LIN_Rec Accuracy	+-70 mV in the range 3 V to (V_Batt-3V)
Settling time	100 ms
General	V_Batt > V_LIN_Rec

3.7 Device IOs

3.7.1 Analog Input Port

Parameter	Value Range
Measurement range	0 – 30 V
Accuracy	+-1 % of full scale
Input resistance	130 KOhm
Max voltage	+-40 V

3.7.2 Analog Output Port

Parameter	Value Range
Voltage range	0 – 15 V
Accuracy	+-0,5%
Output current	+- 5 mA Short circuit proof
Max voltage	+- 40V

3.7.3 Digital Input Port

Parameter	Value Range
Switching threshold	1.5 V – 2.5 V
Input resistance	> 10 kOhm
Max voltage	Max. 40 V

3.7.4 Digital Output Port

Parameter	Value Range
Output type	Open drain with 3.3 kOhm pull-up to V_{Batt}.
Current I_low max	100 mA @ V = 0.2 V **Caution!** The current must not exceed 100 mA! Do not attach a power supply to the port!
Max voltage	30 V

3.7.5 I/O Connector Pin Assignment

Pin - 9 pole Sub-D (female)	Description
1	Digital input
2	Not available
3	Terminal 31’
4	Not used
5	Analog input
6	Digital output
7	Terminal 30
8	Terminal 15
9	Analog output

Caution!

Do not draw over 0.5 A at any pin.

3.8 Short Circuit, Interruption and Crossing of Network Lines

The network lines are routed over the VH1160 to the DUT.

The network lines can be interrupted, short-circuited or crossed.

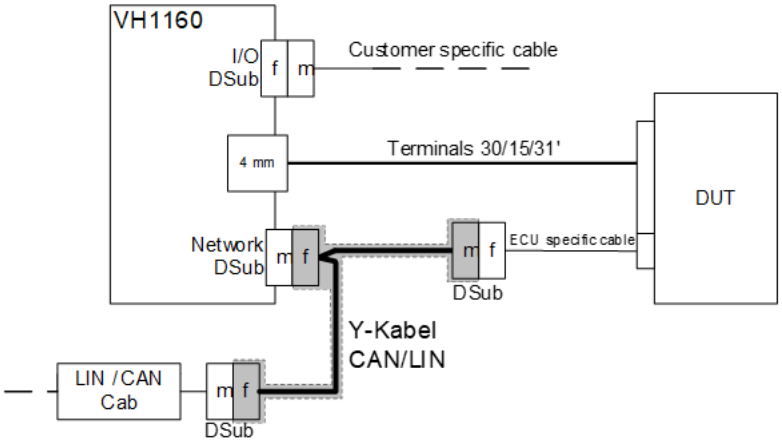


Figure 9: Connection network line for short circuit, interruption and crossing (specification of Y-Cable see chapter 3.8.1 Y-Cable)

Possible Faults
CAN_H – ground
CAN_L – ground
CAN_H – V batt
CAN_L – V batt
CAN_H – CAN_L
CAN_H interrupted
CAN_L interrupted
CAN_H – CAN_L
CAN_L – CAN_H

3.8.1 Y-Cable

Feature	Value Range
Connector Tester	Dsub 9 pole (female)
Connector DUT	Dsub 9 pole (male)
Connector VH11xx	Dsub 9 pole (female)

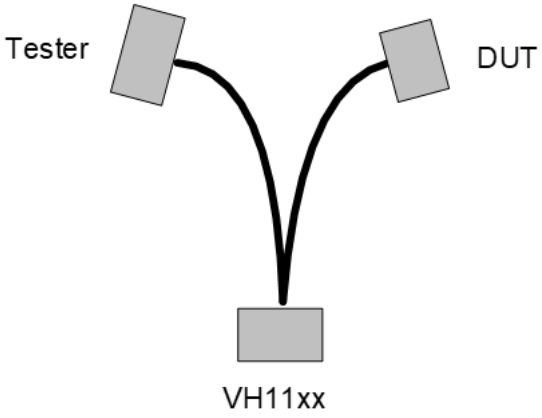


Figure 10: Y-Cable

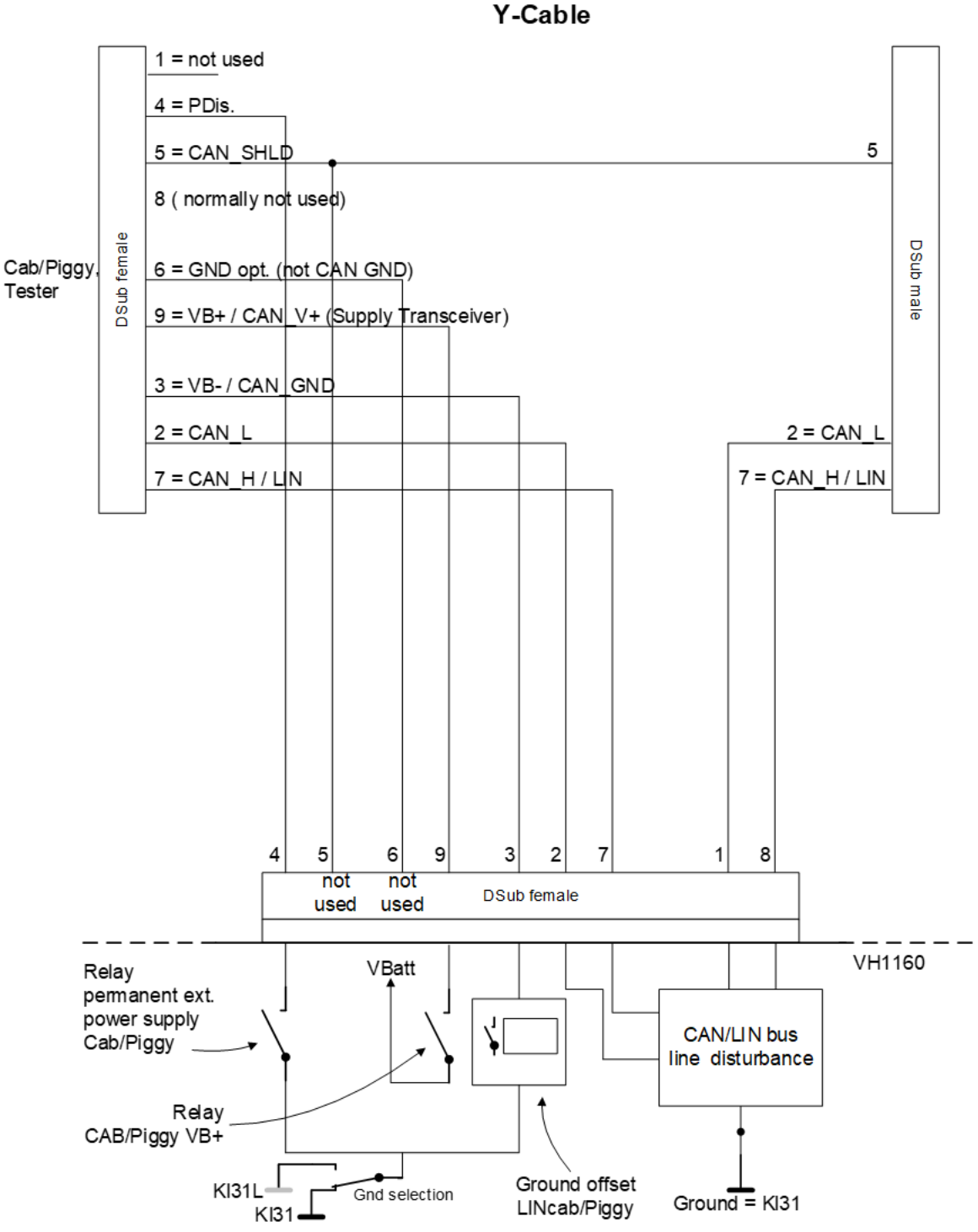


Figure 11: Y-Cable Wiring

3.8.2 Connector Network

Pin - 9 pole. DSub (male)	Name	Description
1	CAN_L DUT	CAN low connection of the device (DUT)
2	CAN_L Tester	CAN low connection of the CANcab/Piggy
3	CAN GND Cab	Connected to terminal 31
4	Pdis	LINcab/LINPiggy forced to external supply
5	—	Not used
6	—	Not used
7	CAN H / LIN Tester	CAN / LIN connection tester
8	CAN H / LIN DUT	CAN / LIN connection ECU (DUT)
9	CAN VB+	Can be switched with relay to V_{Batt}

Note

For CAN and LIN tests it is required that the DUT is also connected to the supply terminals 30 (V_{Batt}) or 15 (IGN) and 31' (GND).

4 API

In this chapter you find the following information:

4.1 General Functions

4.1.1 VHDevSearchOpen

Syntax	`long VHDevSearchOpen (long pHandle[1], char actualDevType[], char actualDevName[], int bufferSize);`		
Function	At first, the function tries to open a VH1100 device if National Instruments driver is installed. The function builds a list of compatible devices as known by the National Instruments driver. If this list contains more than one entry, no device will be opened and the function returns the error code ERR_OPEN_NO_REPLACEMENT_FOUND. If the list contains exactly one entry, the function opens this device. If no VH1100 device is found the function tries to open a VH1150/VH1160 device. The function opens the first compatible device. If no compatible device is found, the function returns the error code ERR_OPEN_NO_REPLACEMENT_FOUND.		
Parameters	**pHandle** Returns a handle. The handle is valid if the function returns zero, otherwise the handle must not be used.		
	actualDevType Returns the type of the current device. Either VH1100 or VH1150.		
	actualDevName VH1150/VH1160: The COM-PortID referring to the device as a one or two-digit number encoded as string. VH1100: The name of the device as specified in the Measurement and Automation Explorer (MAX). The name must not be a number (eg. "12"). Typical names are **dev1** or **dev2**.		
	bufferSize Size of the buffers **actualDevType** and **actualDevName**.		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter .4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.1.2 VHDevOpen

Syntax	`long VHDevOpen (char devName[], long pHandle[1]);`
Function	This function opens the specified VH device. The format of the parameter devName determines whether the function opens a VH1100 or VH1150/VH1160 device.

Parameters	**devName** VH1150/VH1160: COM-PortID encoded as string. The COM-PortID is a one or two digits number and identifies the COM port of the VH1150/VH1160. There can be entered any value from 0-99. VH1100: The name of the device as specified in the Measurement and Automation Explorer (MAX). Typical names are **dev1** or **dev2**.		
	pHandle Returns a handle. The handle is valid if the function returns zero, otherwise the handle must not be used.		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter .4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.1.3 VHDevOpen2

Syntax	`long VHDevOpen2 (char devName[], long pHandle[1], char actualDevName[], int bufferSize);`		
Function	This function opens the specified VH device. The format of the parameter devName determines whether the function opens a VH1100 or VH1150/VH1160 device. If the specified device is not found, the function tries to open the first available device of the specified type.		
Parameters	**devName** VH1150/VH1160: COM-PortID encoded as string. The COM-PortID is a one or two digits number and identifies the COM port of the VH1150/VH1160. There can be entered any value from 0-99. VH1100: The name of the device as specified in the Measurement and Automation Explorer (MAX). Typical names are **dev1** or **dev2**.		
	pHandle Returns a handle. The handle is valid if the function returns zero, otherwise the handle must not be used.		
	actualDevName VH1150/VH1160: The COM-PortID referring to the device as a one or two-digit number encoded as string. VH1100: The name of the device as specified in the Measurement and Automation Explorer (MAX). The name must not be a number (eg. "12"). Typical names are **dev1** or **dev2**.		
	bufferSize Size of the buffers **actualDevType** and **actualDevName**.		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter .4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.1.4 VHDevClose

Syntax	`long VHDevClose (long devHandle);`		
Function	Closes an opened device. Calling this function invalidates the handle.		
Parameters	**devHandle** A handle referencing the VH device.		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter .4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.1.5 VHDevSetSafeState

Syntax	`long VHDevSetSafeState (long devHandle);`		
Function	This function sets the device in a defined and secure state.		
Parameters	**devHandle** A handle referencing the VH device.		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter .4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	• (HW-Rev ≥ 2)	•

4.1.6 VHDevGetHWRevision

Syntax	`long VHDevGetHWRevision (long devHandle, long revision[1]);`
Function	Retrieves the hardware revision number. This number describes the features of the VH device. API functions may require a certain hardware revision number. For example the function VHDevSetCurrentMeasurementRange for a VH1150/VH1160 device requires a HRN of ≥ 5. If the application uses such a function, the application should first check the HRN and handle the case when the HRN is too low.  **Note** The HRN can not be used to distinguish between a VH1100 and VH1150/VH1160.

Parameters	**devHandle** A handle referencing the VH device.		
	revision The hardware revision number.		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.1.7 VHDevResetOverload

Syntax	`long VHDevResetOverload (long devHandle);`		
Function	This function terminates the overload state of the device. Calling this function has no effect when the device is not in the overload mode. If a overload condition (eg. short circuit on output) is pending, the device may go back into its overload state.		
Parameters	**devHandle** A handle referencing the VH device.		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

4.1.8 VHDevGetOverload

Syntax	`long VHDevGetOverload (long devHandle, long overloadState[1]);`		
Function	This function retrieves the overload state of the device.		
Parameters	**devHandle** A handle referencing the VH device.		
	overloadState Function returns 0: normal state, 1: device is in overload state.		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

4.1.9 VHDevSetDebugFlag

Syntax	`long VHDevSetDebugFlag (long debugflag);`		
Function	This function sets the debug option flag to enable debug output into the CANoe Write Window.		
Parameters	**debugflag** 0=disable, 1=enable		
Return Values	0 on success, handle is valid. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.2 Terminal Control and Measurement

4.2.1 VHDevSetRelay

Syntax	`long VHDevSetRelay (long devHandle, long selector, long state);`		
Function	Switches the specified relay. VH1150/VH1160: The ground line relay (GND, 31) cannot be controlled if the ground offset is enabled.		
Parameters	**devHandle** A handle referencing the VH device.		
	selector Selects the relay 0=V30, 1=V15, 2=V31.		
	state Relay contacts 0=open, 1=closed.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.2.2 VHDevSetRelayEx

Syntax	`long VHDevSetRelayEx (long devHandle, long selector, long state, long relaySettleTime);`

Function	Switches the specified relay. This function should be used for special applications only, in most applications `VHDevSetRelay` can be used instead. VH1150/VH1160: The ground line relay (GND, 31) cannot be controlled if the ground offset is enabled.		
Parameters	**devHandle** A handle referencing the VH device.		
	selector Selects the relay 0=V30, 1=V15, 2=V31.		
	state Relay contacts 0=open, 1=closed		
	relaySettleTime Relay settle time in ms. The function sends the relay command to the VH device and waits the specified time before it returns to the caller. A settleTime of zero is valid.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.2.3 VHDevGetVoltage

Syntax	`long VHDevGetVoltage (long devHandle, long selector, float voltage[1]);`		
Function	Retrieves a voltage value at the specified terminal.		
Parameters	**devHandle** A handle referencing the VH device.		
	selector 0=V30, 1=V15, VH1150/VH1160: 2=V31'. This value is valid only when ground offset is turned on.		
	voltage The measured voltage in volts.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.2.4 VHDevGetCurrent

Syntax	`long VHDevGetCurrent (long devHandle, long range, float current[1]);`

Function	Retrieves the measured current at the terminals 30/15. VH1100: The caller is responsible to set an appropriate range selector.		
Parameters	**devHandle** A handle referencing the VH device.		
	range 0=low up to 200 mA, 1=high up to 2A VH1150/VH1160: parameter is ignored		
	current The measured voltage in volts.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.2.5 VHDevSetVoltageVBatt

Syntax	`long VHDevSetVoltageVBatt (long devHandle, float voltage);`		
Function	Sets the output voltage V_{Batt} of the device.		
Parameters	**devHandle** A handle referencing the VH device.		
	voltage Output voltage in volts.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.2.6 VHDevSetVoltageVBattEx

Syntax	`long VHDevSetVoltageVBattEx (long devHandle, float voltage, long settleTime);`
Function	Sets the output voltage V_{Batt} of the device. This function should be used for special applications only, in most applications `VHDevSetVoltageVBatt` can be used instead.

Parameters	**devHandle** A handle referencing the VH device.		
	voltage Output voltage in volts.		
	settleTime Settle time in ms. The function sends the set voltage command to the device, waits the specified **settleTime**, and returns to the caller. A **settleTime** of zero is valid.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	•	•	•

4.2.7 VHDevSetCurrentMeasurementRange

Syntax	`long VHDevSetCurrentMeasurementRange(long devHandle, long range);`		
Function	Sets the current measurement range. **Note** It is not possible to select the low current measurement range.		
Parameters	**devHandle** A handle referencing the VH device.		
	range 0=automatic, device selects the current measurement range. This is the default behavior. 2=high range (2 is correct, not a typo)		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	• (HW-Rev ≥ 5)	•

4.3 Ground Offset and LIN Recessive Limit

4.3.1 VHDevSetGroundOffset

Syntax	`long VHDevSetGroundOffset (long devHandle, float offsetVoltage);`

Function	This function shifts the ECU ground voltage by the specified value. **Note** Enabling the ground offset will turn off the feature **LIN Recessive Level** and on the VH1150 also the **Analog Output**.		
Parameters	**devHandle** A handle referencing the VH device.		
	offsetVoltage Range 0.0 - 10.0 V. A value of <= 0.0 turns off the ground offset feature.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•
	Example If the ECU is supplied with a voltage of 14.4 V and the ground offset is set to 2.0 V, the actual ECU supply voltage measured between the pins IGN and GND' (connector 31') amounts to 12.4 V. Note that the battery voltage, measured between the pins IGN and GND (connector 31), remains unchanged (ie. 14.4 V). Depending on the ECU current and offset voltage, the ground offset feature may considerably increase the power dissipation of the VH device. This may cause the VH device to transit into its overload state.		

4.3.2 VHDevSetTesterBusGroundOffset

Syntax	`long VHDevSetTesterBusGroundOffset (long devHandle, float offsetVoltage);`		
Function	This function shifts the bus ground line on the tester side by the specified value. This feature is intended to be used in LIN networks. **Note** Enabling the ground offset will turn off the feature **Ground Offset** and **LIN Recessive Level**. On the VH1150 also the **Analog Output** will be switched off.		
Parameters	**devHandle** A handle referencing the VH device.		
	offsetVoltage Range 0.0 - 20.0 V. A value of <= 0.0 turns off the ground offset feature.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

	**Example** On a LIN network, the bus ground offset (on the tester side) is set to a value of 2.0 V. If the LIN line has a recessive level, the ground offset has no primary effect. Both, the tester and the ECU, measure the recessive level on the LIN bus. If the tester sets its dominant level, the ECU sees a LIN voltage of about 2.7 V. This value results from the 2.0 V offset and a 0.7 V voltage over a diode in the drain – source path of the tester transceiver transistor. Thus, the Tester Ground Offset feature allows to increase the dominant level recognized by the ECU. Typically, the LIN transceiver of the tester is supplied by the VH device. Even if the tester transceiver is setup to generate its own supply voltage, the Tester Ground Offset feature can be used.

4.3.3 VHDevSetLINRecessiveVoltageLimit

Syntax	`long VHDevSetLINRecessiveVoltageLimit (long devHandle, float recessiveVoltage);`		
Function	This function limits the recessive LIN voltage level to the specified value.		
Parameters	**devHandle** A handle referencing the VH device.		
	recessiveVoltage Range 0.0 - 28.0 V. A value of <= 0.0 turns the feature off.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•
	**Example** The ECU is supplied with a voltage of 14.4 V and the recessive voltage is set to 9.5 V. This limits a LIN recessive bit (high) to 9.5 V. If the LIN recessive voltage is set to a value larger than the master's battery voltage (in this example > 14.5 V) the recessive voltage remains unchanged.		

4.4 Device Input/Output

4.4.1 VHDevSetAnalogOutput

Syntax	`long VHDevSetAnalogOutput (long devHandle, float analogOutputVoltage);`

Function	This function sets the Analog Output voltage to the specified value.  **Note for VH1150** The output voltage is derived from the battery voltge V_{Batt}. This means the analog output voltage will never exceed V_{Batt}. Turning on the **Analog Output** will turn off the features **LIN Recessive Voltage** and **Ground Offset**.		
Parameters	**devHandle** A handle referencing the VH device.		
	analogOutputVoltage Range 0.0 - 15.0 V. A value of <= 0.0 turns off the feature.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

4.4.2 VHDevSetDigitalOutput

Syntax	`long VHDevSetDigitalOutput (long devHandle, long state);`		
Function	This function sets the digital output to the specified state.		
Parameters	**devHandle** A handle referencing the VH device.		
	state ▶ 0=false=low ▶ 1=true=high		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

4.4.3 VHDevSetLINCabSupplyMode

Syntax	`long VHDevSetLINCabSupplyMode (long devHandle, long mode);`
Function	This function sets the voltage supply mode of the LINcab connected to the Cab/Piggy connector of the VH1150/VH1160 Y-cable.

Parameters	**devHandle** A handle referencing the VH device.		
	mode ▶ 0=internal: LINcab produces its own 12V supply voltage. ▶ 1=external/internal: if an external supply voltage > 12V is available (pin 9) it will be used. Otherwise, the LINcab produces its own supply voltage. ▶ 2=external The LINcab uses the external supply voltage (pin 9). If there is none, the LINcab is not supplied!		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

4.4.4 VHDevGetAnalogInput

Syntax	`long VHDevGetAnalogInput (long devHandle, float analogInputVoltage[1]);`		
Function	This function retrieves the analog input voltage.		
Parameters	**devHandle** A handle referencing the VH device.		
	analogInputVoltage Measured voltage in volts.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

4.4.5 VHDevGetDigitalInput

Syntax	`long VHDevGetDigitalInput (long devHandle, long digitalInputState[1]);`		
Function	This function retrieves the digital input state.		
Parameters	**devHandle** A handle referencing the VH device.		
	digitalInputState Function returns 0=false (low voltage), 1=true (high voltage).		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

4.5 Bus Line Disturbance

4.5.1 VHDevSetBusFaults

Syntax	`long VHDevSetBusFaults (long devHandle, long passThru, long CanH, long CanL);`		
Function	This function interrupts or connects the CAN/LIN lines to ground or V_{Batt}. The mode is specified by three parameters. In the following description CAN_H corresponds to LIN.  **Note** Enabling this function will turn off the **LIN periodical short to ground**.		
Parameters	**devHandle** A handle referencing the VH device.		
	passThru ▶ 0=normal operation CAN_H and CAN_L are not interrupted ▶ 1=CAN_H interrupted ▶ 2=CAN_L interrupted ▶ 3=CAN_H and CAN_L are interrupted		
	CanH ▶ 0=nomal operation ▶ 1=connected to GND ▶ 2=connected to V_{Batt} ▶ 3=connected to CAN_L, parameter CanL is ignored		
	CanL ▶ 0=nomal operation ▶ 1=connected to GND 2=connected to V_{Batt}		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	•	•

4.5.2 VHDevSetPeriodicalLINBusFault

Syntax	`long VHDevSetPeriodicalLINBusFault (long devHandle, long mode, float frequency);`

Function	Switches the LIN line periodically to either GND or V_{Batt}. **Note** Enabling this function will turn off the **LIN bus faults**.		
Parameters	**devHandle** A handle referencing the VH device.		
	mode ▶ 0=normal operation, short connection disabled ▶ 1=fast periodic short connection to GND ▶ 2=fast periodic short connection to V_{Batt}		
	frequency Frequency of short connection in Hertz. 1/(2*f) = time span of connection to GND or V_{Batt}. Allowed range: 2 Hz - 50 kHz.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	• (HW-Rev ≥ 4)	•

4.6 CAN Network

4.6.1 VHDevMeasureCANTermResistor

Syntax	`long VHDevMeasureCANTermResistor (long devHandle, float termResPosMeasVoltage[1], float termResNegMeasVoltage[1]);`
Function	This function measures the CAN termination resistor between the CAN-H and CAN L lines. The termination resistor is measured twice by inverting the polarity of the measurement voltage. One measurement uses a positive CAN-H to CAN-L voltage, whereas the other measurement puts CAN-L on a higher voltage than CAN-H.
Parameters	**devHandle** A handle referencing the VH device.
	termResPosMeasVoltage The measured termination resistor value in Ohm. The measurement is carried out with a positive CAN-H to CAN-L measurement voltage.
	termResNegMeasVoltage The measured termination resistor value in Ohm. The measurement is carried out with a negative CAN-H to CAN-L measurement voltage.
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.

Support	VH1100	VH1150	VH1160
	—	• (HW-Rev ≥ 4)	•

4.6.2 VHDevSetCANCapacitors

Syntax	`long VHDevSetCANCapacitors (long devHandle, long enable);`		
Function	Switches capacitors to the CAN-H and CAN-L lines. The other ends of the capacitors are connected to Gnd. Each capacitor has capacitance of 1.6 nF.		
Parameters	**devHandle** A handle referencing the VH device.		
	enable ▶ 0=capacitors are switched off, i.e. they are not connected to the CAN lines. ▶ 1=capacitors are switch on, i.e. they are connected to the CAN lines.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	• (HW-Rev ≥ 4)	•

4.6.3 VHDevMeasureCANDiffVoltage

Syntax	`long VHDevMeasureCANDiffVoltage (long devHandle, long measurementPeriod, float vRecessive[1], float vDominant[1]);`		
Function	Measures the recessive and dominant voltage of the differential CAN signal.		
Parameters	**devHandle** A handle referencing the VH device.		
	measurementPeriod Duration of measurement in milliseconds. If **measurementPeriod** is <= 0, the default value of 1000 ms is used.		
	vRecessive Measured recessive (logic level H or 1) voltage of U_CAN_diff in volts.		
	vDominant Measured dominant (logic level L or 0) voltage of U_CAN_diff in volts.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	• (HW-Rev ≥ 4)	•

4.6.4 VHDevSetCANLinesCrossed

Syntax	`long VHDevSetCANLinesCrossed (long devHandle, long crossed);`		
Function	Sets whether the CAN_H and CAN_L bus lines are crossed or not.		
Parameters	**devHandle** A handle referencing the VH device.		
	crossed ▶ 0=normal operation CAN_H and CAN_L are not crossed. ▶ 1=CAN_H and CAN_L are crossed.		
Return Values	0 on success. Otherwise an error code, see chapter 4.7 Error Codes.		
Support	**VH1100**	**VH1150**	**VH1160**
	—	—	•

4.7 Error Codes

Error Code	Error No.	Comment
ERR_OK	0	
ERR_FAILED	-1	
ERR_INVALID_HANDLE	2	
ERR_NOT_IMPLEMENTED	3	
ERR_INVALID_PARAMETER	4	
ERR_DEVICE_INUSE	5	
ERR_VIA_NL	6	
ERR_VIA_CAST	7	
ERR_AINFAILED	8	
ERR_AINNODATA	9	
ERR_DRIVERDLL_NOT_LOADED	10	Unable to load driver DLL.
ERR_INTERNAL_NONL	11	Internal error, nodelayer not available.
ERR_VOLTAGEPROFILE_RUNNNING	12	Function call cot allowed at this time.
ERR_HARDWARE_REV_NUMBER_TOO_LOW	13	Newer hardware required.
ERR_CALIB_FAILED	14	
ERR_OPEN_NO_REPLACEMENT_FOUND	15	Open of specified device failed. Too many or no compatible devices found.
ERR_INCOMPATIBLE_DEVICE	16	This type of device can not handle the command.
ERR_NO_ANSWER_FROM_INTERFACE	17	Interface is not responding.

5 Index

V

W

Y

More Information

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

www.vector.com