Manual

XL Driver Library

.NET Wrapper Description

Version 7.5

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 2010, Vector Informatik GmbH. Printed in Germany.
All rights reserved.

Table of contents

1 Introduction

In this chapter you find the following information:

1.1 About this User Manual

1.1.1 Access Help and Conventions

To find information quickly

The user manual provides you the following access help:

- At the beginning of each chapter you will find a summary of the contents,
- In the header you can see in which chapter and paragraph you are ((situated)),
- In the footer you can see to which version the user manual replies.

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
Windows	Legally protected proper names and side notes.
`Source code`	File name and source code.
Hyperlink	Hyperlinks and references.
<STRG>+<S>	Notation for shortcuts.

Symbol	Utilization
	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.

1.1.2 Certification

Certified Quality Management System

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

1.1.3 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 user manual. 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.

1.1.4 Support

You need support?

You can get through to our support at the phone number

+49 711 80670-200 or by fax

+49 711 80670-111

E-Mail: support@vector-informatik.de

1.1.5 Registered Trademarks

Registered trademarks

All trademarks mentioned in this user manual 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 user manual, fails, should not mean that a name is free of third party rights.

➔ Windows, Windows XP, Windows Vista, Windows 7 are trademarks of the Microsoft Corporation.

2 .NET Wrapper

In this chapter you find the following information:

2.1 Overview

Caution: THE INCLUDED WRAPPER IS PROVIDED “AS-IS”. NO LIABILITY OR RESPONSIBILITY FOR ANY ERRORS OR DAMAGES.

Info: The Vector Driver Disk includes a separate .NET wrapper (assembly) for the XL Driver Library: `\Drivers\XL Driver Library\bin\vxlapi_NET20.dll`.

Framework .NET 2.0 or higher is required.

Info: In order to run the .NET wrapper in your application the general library `vxlapi.dll` must also be placed in the execution folder of your application.

Info: The .NET wrapper only supports CAN, LIN, and DAIO.

Overview

Vector offers an XL Driver .NET wrapper, which allows an easy integration of the XL Driver Library in any .NET environment. This means that Vector XL Interfaces can be accessed from any .NET programming language, like C#, Visual Basic .NET or Delphi .NET (Delphi 8). Examples for these languages are also available.

The XL Driver .NET wrapper consists of the single .NET assembly ***vxlapi_NET20.dll*** and offers the complete functionality of the XL Driver Library. The usage is similar to the XL API. There is also a “comfort layer” included, which allows a very easy access to XL Interfaces by only using a few code lines.

Easy and Advanced Examples

The XL Driver Library setup contains a few examples in different .NET languages in order to explain the usage in each environment, split by **Easy** (describing the “comfort layer”) and **Advanced** (describing the detailed wrapper methods).

 © Vector Informatik GmbH

2.2 Comfort Layer

Info: The comfort layer only supports CAN.

Quick introduction

In order to get a quick idea of the wrapper, we recommend using the comfort layer methods, which keeps the whole initialisation and the handle management in the background.

The comfort layer can be accessed by the class `xlSingleChannelCAN_Port`, which is part of the wrapper.

Class xlSingleCAN_Port

```
public xlSingleChannelCAN_Port(String appName,
                               uint appChannel)
```

Description

This object creates a single port with a single channel, which is opened then. The driver configuration is read from Vector Hardware Config, therefore the application name and the channel number is needed at call.

Info: On the very first call, the application name does not exist in the Vector Hardware Config, but it is automatically created with two CAN channels. Afterwards it is necessary to assign these logical channels to physical channels in the Vector Hardware Config. One channel is used to transmit messages, the other one to receive them.

Input Parameters

- **appName**
  Name of the application, registered in Vector Hardware Conf.
- **appChannel**
  Channel of the named application.

List of methods — The class `xlSingleChannelCAN_Port` contains the following methods:

.xlCheckPort

```
bool xlCheckPort()
```

Description — Checks if the opened port by `xlSingleChannelCAN_Port` object is accessible.

Return value — TRUE if accessible, otherwise FALSE.

.xlActivate

```
bool xlActivate()
```

Description — Activates an accessible port.

Return value — TRUE if accessible, otherwise FALSE.

.xlResetAcceptanceFilter

```
bool xlResetAcceptanceFilter()
```

Description — Resets all acceptance filters of the opened port.

Return value — TRUE if accessible, otherwise FALSE.

.xlCanAddAcceptanceRange

```
bool xlCanAddAcceptanceRange()
```

Description — Adds acceptance filters.

Input Parameters
- **first_id**
  First CAN ID of the range.
- **last_id**
  Last CAN ID of the range.

Return value — TRUE if accessible, otherwise FALSE.

 © Vector Informatik GmbH

.xlTransmit

```
bool xlTransmit(uint id, ushort dlc, UInt64 data)
```

Description: Transmits a single CAN message with a given ID, DLC and the data (8 byte value).

Input Parameters:

- **id**
  CAN ID.
- **dlc**
  Data length code.
- **Data**
  8 byte value to be transmitted.

Return value: TRUE if accessible, otherwise FALSE.

.xlReceive

```
bool xlReceive(ref XLClass.xl_event rxEvent)
```

Description: Reads CAN events (if available) from hardware queue of the XL Interface.

Input Parameters:

- **rxEvent**
  Reference to event buffer (event type defined in namespace `XLClass`)

Return value: TRUE if accessible, otherwise FALSE.

.xlPrintConfig

```
bool xlPrintConfig()
```

Description: Prints the current configuration of the port to the console.

Return value: TRUE if accessible, otherwise FALSE.

.xlPrintRx

```
bool xlPrintRx(XLClass.xl_event receivedEvent)
```

Description: Prints the received message to the console.

Input Parameters:

- **receivedEvent**
  Event to display (event type defined in namespace `XLClass`).

Return value: TRUE if accessible, otherwise FALSE.

2.3 General Layer

General layer of the .NET wrapper

The complete XL API can be accessed by the general layer, which is included in the class **XLDriver**. For further details on this methods, please look up the general XL API Description.

Info: The method names of the wrapper differs only in the prefix, e.g.

Wrapper	: `XL_OpenDriver`
XL API	: `xlOpenDriver`

Methods of the general layer

The following methods are available after the instantiation of **XLDriver**:

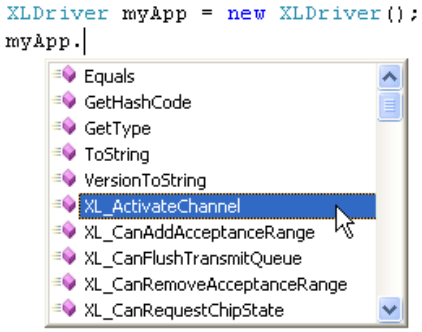
Class XLDriver

```
.XL_ActivateChannel
.XL_CanAddAcceptanceRange
.XL_CanFlushTransmitQueue
.XL_CanRemoveAcceptanceRange
.XL_CanRequestChipState
.XL_CanResetAcceptance
.XL_CanSetChannelAcceptance
.XL_CanSetChannelBitrate
.XL_CanSetChannelMode
.XL_CanSetChannelOutput
.XL_CanSetChannelParams
.XL_CanSetChannelParamsC200
.XL_CanSetChannelTransceiver
.XL_CanTransmit
.XL_CloseDriver
.XL_ClosePort
.XL_DAIORequestMeasurement
.XL_DAIOSetAnalogOutput
.XL_DAIOSetAnalogParameters
.XL_DAIOSetAnalogTrigger
.XL_DAIOSetDigitalOutput
.XL_DAIOSetDigitalParameters
.XL_DAIOSetMeasurementFrequency
.XL_DAIOSetPWMOutput
.XL_DeactivateChannel
.XL_FlushReceiveQueue
.XL_GenerateSynPulse
.XL_GetApplConfig
```

 © Vector Informatik GmbH

```
.XL_GetChannelIndex
.XL_GetDriverConfig
.XL_GetErrorString
.XL_GetEventString
.XL_GetReceiveQueueLevel
.XL_GetSyncTime
.XL_LinSendRequest
.XL_LinSetChannelParams
.XL_LinSetChecksum
.XL_LinSetDLC
.XL_LinSetSlave
.XL_LinSetSleepMode
.XL_LinWakeUp
.XL_OpenDriver
.XL_OpenPort
.XL_PopupHwConfig
.XL_Receive
.XL_ResetClock
.XL_SetApplConfig
.XL_SetNotification
.XL_SetTimerRate
...
```

Info: The complete list can be looked up with help of the intellisense function in your IDE:



Info: Some of these methods have parameters expecting objects. All needed objects are already defined in the Class `XLClass` and ready to use. Most of these classes can be found again in the XL API in the form of structures (instead of a class). Please refer to the general XL API Description for further details on these structures.

The `XLClass` also offers a wide range of enumerations for an easy access to values and definitions. See the following tables for the detailed class information.

2.3.1 XLClass - Defined Objects

xl_driver_config	Needed by method `XL_GetDriverConfig`. Contains driver configuration.
xl_event_collection	Needed by method `XL_CanTransmit.` Contains one or more xl_events (see below) for transmission.
xl_event	Contains data to be transmitted.
xl_bus_params	Used by subclass `xl_channel_config`.
xl_channel_config	Used by subclass `xl_driver_config`.
xl_can_message	Used by subclass `xl_event`.
xl_chip_params	Used by method `XL_CanSetChannelParams`.
xl_linStatPar	Used by method `XL_LinSetChannelParams`.

2.3.2 XLClass - Defined Enumerations

Info: Please refer to `vxlapi.h` for all available definitions, or use the integrated intellisense function of your IDE to look up the available enumerations.

Each enumeration can be directly used without instantiation of `XLClass` by using `XLClass.<subclass>.<enumeration_value>`,

e.g. `XLClass.XLstatus.XL_SUCCESS`.

XLstatus	→ .XL_SUCCESS → .XL_PENDING → .XL_ERR_QUEUE_IS_EMPTY → .XL_ERR_QUEUE_IS_FULL → .XL_ERROR → …
XLbusTypes	→ .XL_BUS_TYPE_NONE → .XL_BUS_TYPE_CAN → .XL_BUS_TYPE_HWSYNC → …
XLevenType	→ .XL_NO_COMMAND → .XL_RECEIVE_MSG → .XL_CHIP_STATE → .XL_RECEIVE_DAIO_DATA → …
XLmessageFlag	→ .XL_CAN_MSG_FLAG_ERROR_FRAME

 © Vector Informatik GmbH

- .XL_CAN_MSG_FLAG_OVERRUN
- .XL_LIN_MSGFLAG_CRCERROR
- …

XLtransceiverTypes

- .XL_TRANSCEIVER_TYPE_NONE
- .XL_TRANSCEIVER_TYPE_CAN_251
- .XL_TRANSCEIVER_TYPE_DAIO_8444_OPTO
- …

XLhwTypes

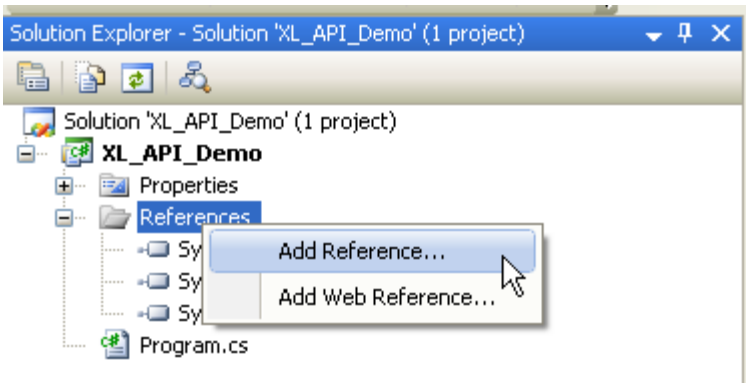
- .XL_HWTYPE_NONE
- .XL_HWTYPE_VIRTUAL
- .XL_HWTYPE_CANCARDXL
- .XL_HWTYPE_CANBOARDXL_PXI
- …

2.4 Including the wrapper in a new .NET Project

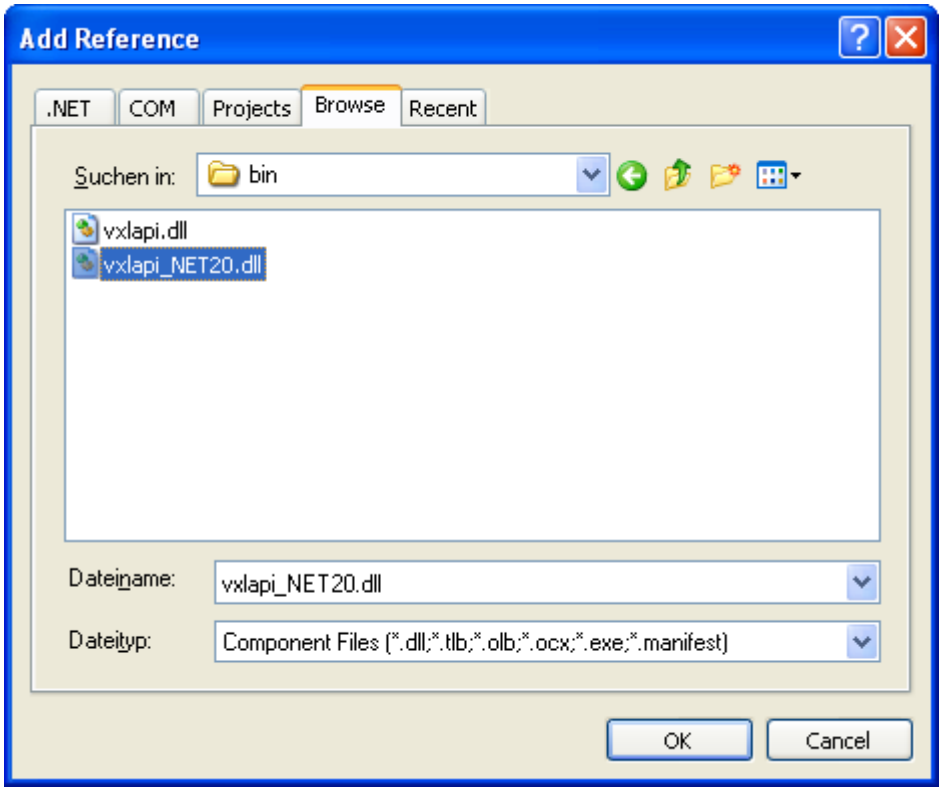
General notes

If the starting point of your project is an empty .NET application, do the following in order to use the XL Driver .NET Wrapper. These steps are related to Visual Studio 2008 and C# but similar in any other IDE.

1. Copy the general XL Driver Library `vxlapi.dll` to your execution folder of your project (this might be `\Debug` or `\Release`).
2. In VS2008 click with the right mouse button on **References** (Solution Explorer) of your project and select **Add Reference...**

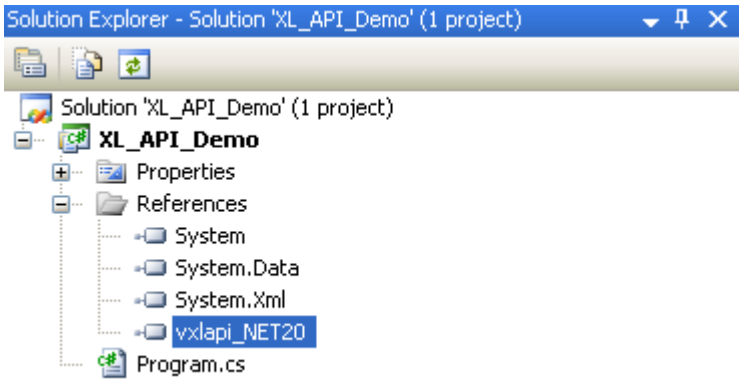


3. Browse for the .NET wrapper `vxlapi_NET20.dll`.



4. Close the dialog with **[OK]**. The DLL appears in the Solution Explorer.

 © Vector Informatik GmbH



5. Enter the following line in the top of your source code to access the wrapper:

```
using vxlapi_NET20;
```

6. Now, you are able to create an object from class `XLDriver`:

```
private static XLDriver MyApp = new XLDriver();
```

7. Try to open the port by entering the line:

```
MyApp.XL_OpenDriver();
```

Info: Take a look at our examples (source code) on the Driver Disk for further information.

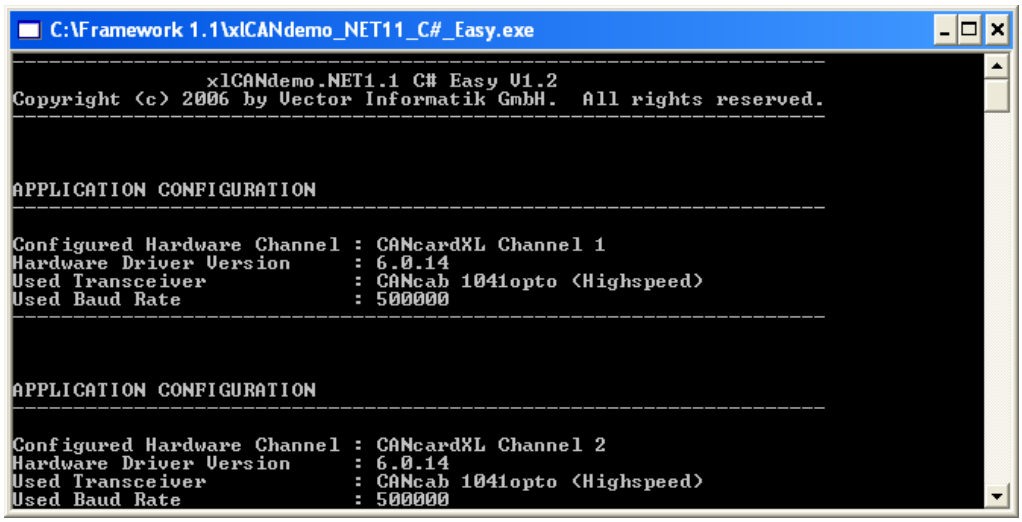
3 Examples

In this chapter you find the following information:

Info: All examples are available for C#, Visual Basic .NET, and Delphi .NET (Delphi 8).

3.1 xlCANdemo .NET Easy

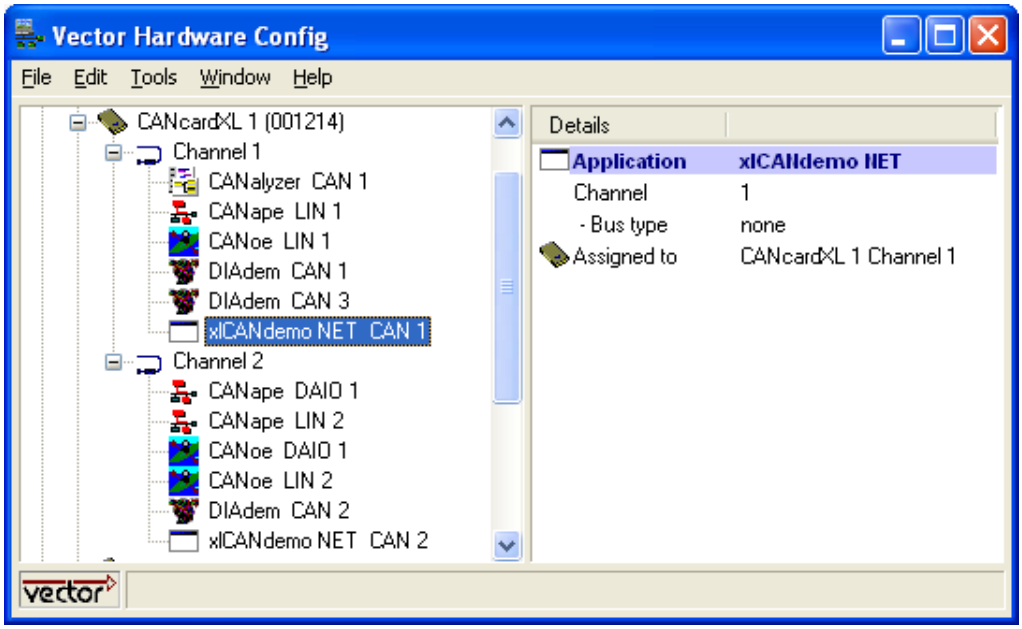
Screenshot



CAN loop application

This example is a simple CAN loop application which demonstrates the usage of the comfort layer. The access to the XL Interfaces is done by using the class `xlSingleChannelCAN_Port` while the whole configuration of the application is done through the Vector Hardware Config.

Vector Hardware Config



Starting the example

When the application starts, it looks for the application **xlCANdemo NET** in the **Vector Hardware Config** and reads its configuration there, e.g. the assigned XL Interfaces. Since such application is not registered at the very first time, it is automatically created with two CAN channels. Afterwards the channel assignment can be

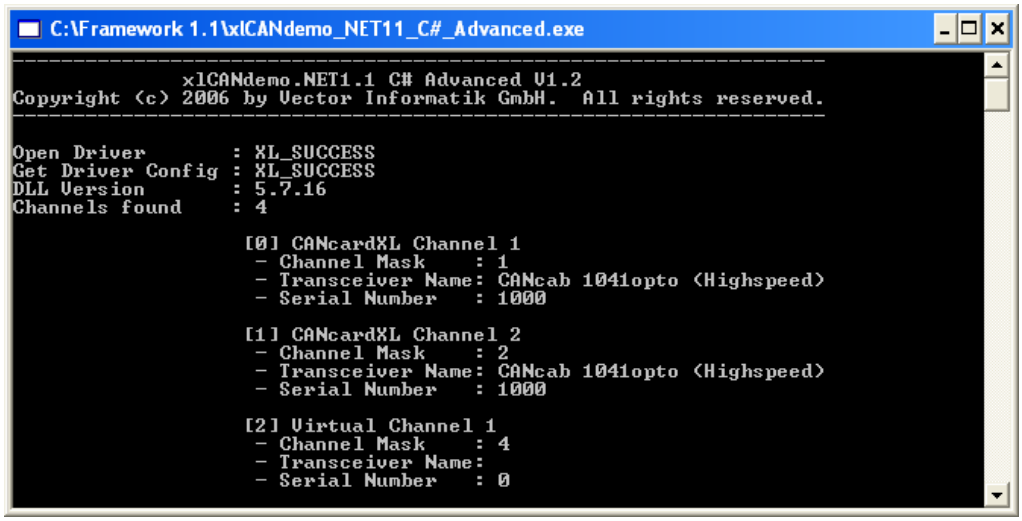
quickly done in **Vector Hardware Config**.

Send and receive messages

When the XL Interface(s) are assigned in the **Vector Hardware Config** and the physical channels are connected, CAN messages can be transmitted by pressing the **[ENTER]** key. The message is sent over the first configured channel and is received by the second one.

3.2 xlCANdemo .NET Advanced

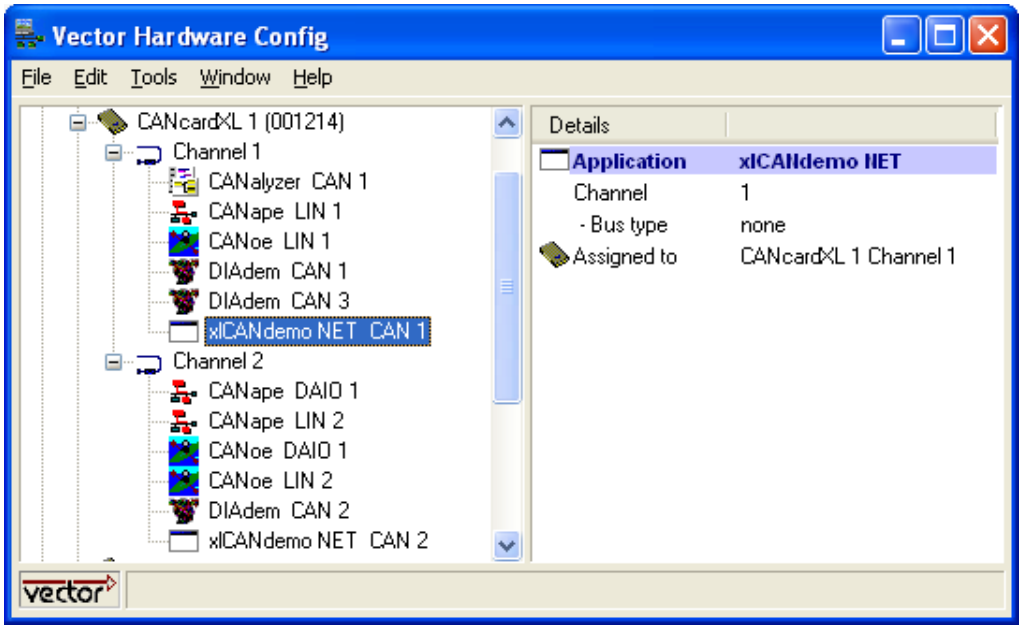
Screenshot



CAN loop application

This example is a simple CAN loop application which demonstrates the usage of the general wrapper in the common way. The access to the XL Interfaces is done by using the class `XLDriver` while the whole configuration of the application is done over the Vector Hardware Config.

Vector Hardware Config



Starting the example

When the application starts, it looks for the application **xlCANdemo NET** in the **Vector Hardware Config** and reads its configuration there, e.g. the assigned XL

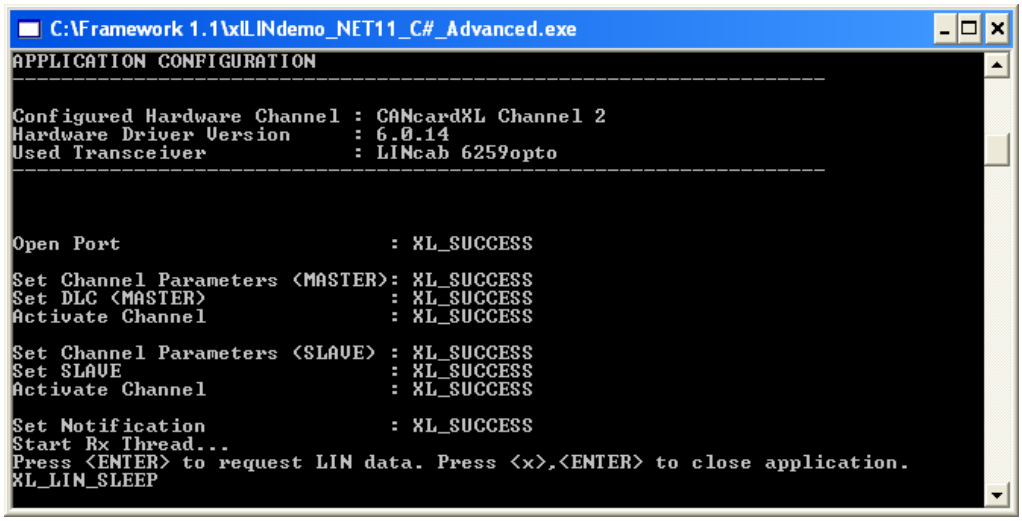
Interfaces. Since such application is not registered at the very first time, it is automatically created with two CAN channels. Afterwards the channel assignment can be quickly done in **Vector Hardware Config**.

Send and receive messages

When the XL Interface(s) are assigned in the **Vector Hardware Config**, and the physical channels are connected, CAN messages can be transmitted by pressing the **[ENTER]** key. The message is sent over the first configured channel and is received by the second one.

3.3 xlLINdemo .NET Advanced

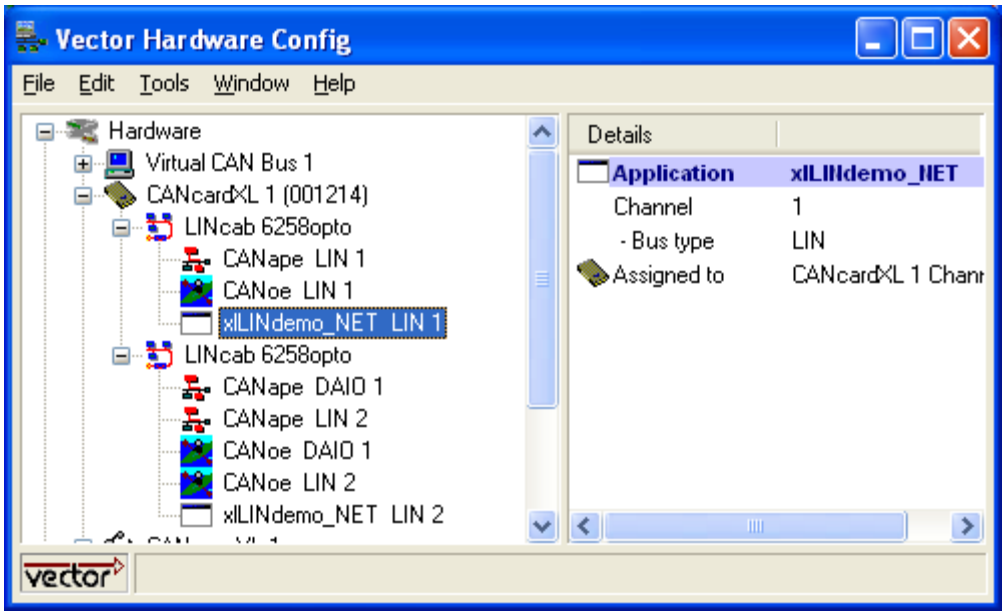
Screenshot



LIN loop application

This example is a simple LIN loop application which demonstrates the usage of the general wrapper in the common way. The access to the XL Interfaces is done by using the class `XLDriver` while the whole configuration of the application is done over the Vector Hardware Config.

Vector Hardware Config



Starting the example

When the application starts, it looks for the application **xlLINdemo NET** in the **Vector**

Hardware Config and reads its configuration there, e.g. the assigned XL Interfaces. Since such application is not registered at the very first time, it is automatically created with two LIN channels. Afterwards the channel assignment can be quickly done in **Vector Hardware Config**.

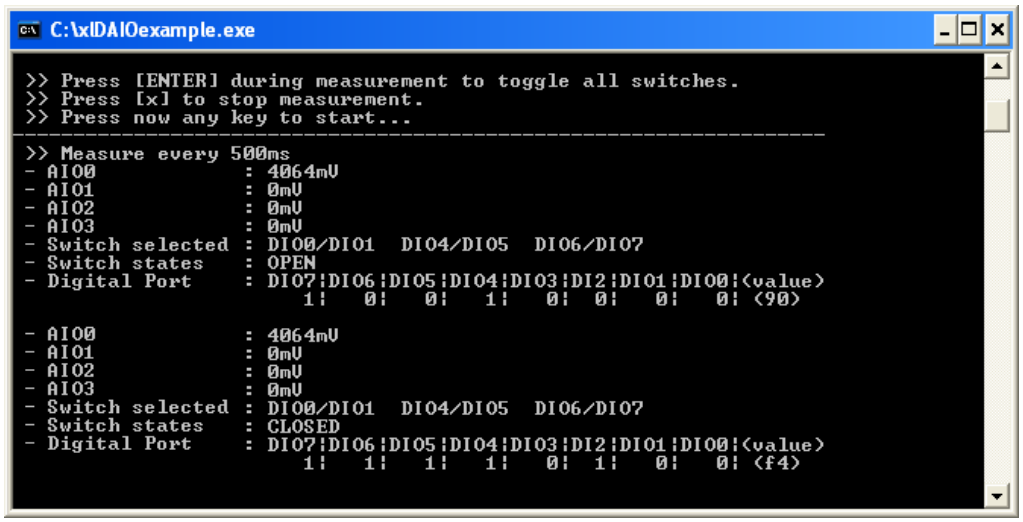
Send and receive messages

When the XL Interface(s) are assigned in the **Vector Hardware Config**, and the physical channels are connected, LIN messages can be transmitted by pressing the **[ENTER**] key. The message is sent over the first configured channel and is received by the second one.

3.4 xlDAIOexample .NET Advanced

Description

This example demonstrates the setup of a single IOcab 8444opto for a test, and how to access the inputs and outputs for cyclically measurement.

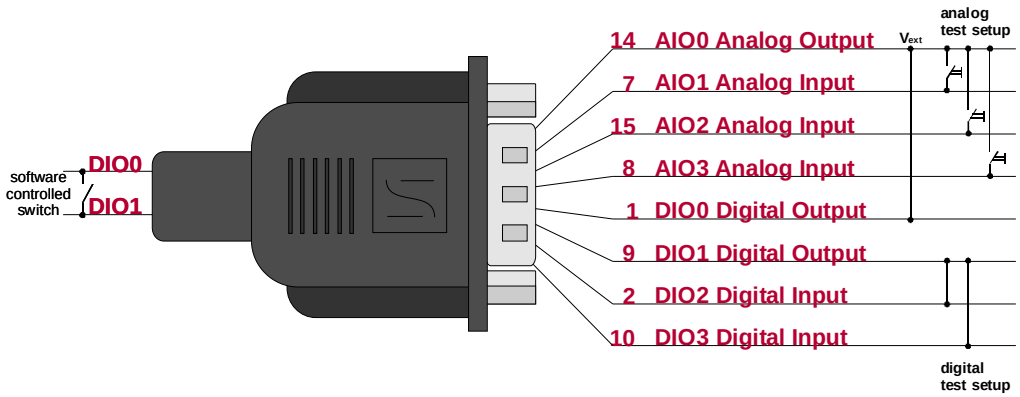


Pin definitions

The following pins of the IOcab 8444opto are used in this example:

- AIO0 (pin 14): Analog output.
- AIO1 (pin 7): Analog input.
- AIO2 (pin 15): Analog input.
- AIO3 (pin 8): Analog input.
- DIO0 (pin 1): Digital output (shared electronic switch with DIO1).
- DIO1 (pin 9): Digital output (supplied by DIO0, when switch is closed).
- DIO2 (pin 2): Digital input.
- DIO3 (pin 10): Digital input.

Setup



Info: The internal switch between DIO0 (supplied by AIO0) and DIO1 is closed/opened with `xlDAIOSetDigitalOutput`. If the switch is closed, the applied voltage at DIO0 can be measured at DIO1.

Keyboard commands

When the application is running, there is a couple of keyboard commands:

Key	Command
ENTER	Toggle digital output.
x	Closes application.

Example: Display output of xlDAIOexample.

```
AIO0                : 4032mV
AIO1                : 0mV
AIO2                : 0mV
AIO3                : 0mV
Switch selected     : DIO0/DI01
Switch states       : OPEN
Digital Port        : DIO7 DIO6 DIO5 DIO4 DIO3 DIO2 DIO1 DIO0 val
                        0    0    0    0    0    0    0    1 (1)
```

Explanation

- ➔ “AIO0” displays 4032mV, since it is set to output with maximum output level.
- ➔ “AIO1” displays 0mV, since there is no applied voltage at this input.
- ➔ “AIO2” displays 0mV, since there is no applied voltage at this input.
- ➔ “AIO3” displays 0mV, since there is no applied voltage at this input.
- ➔ ”Switch selected” displays DIO0/DIO1 (first switch)
- ➔ ”Switch states” displays the state of switch between DIO0/DIO1
- ➔ “Digital Port” shows the single states of DIO7…DIO0:
  - DIO0: displays ‘1’ (always ‘1’, due the voltage supply)
  - DIO1: displays ‘0’ (switch is open, so voltage at DIO0 is not passed through)
  - DIO2: displays ‘0’ (output of DIO1)
  - DIO3: displays ‘0’ (output of DIO1)
  - DIO4: displays ‘0’ (n.c.)
  - DIO5: displays ‘0’ (n.c.)
  - DIO6: displays ‘0’ (n.c.)
  - DIO7: displays ‘0’ (n.c.)

Example: Display output of xlDAIOexample.

```
AIO0                : 4032mV
AIO1                : 0mV
AIO2                : 4032mV
AIO3                : 0mV
Switch selected     : DIO0/DI01
Switch state        : CLOSED
Digital Port        : DIO7 DIO6 DIO5 DIO4 DIO3 DIO2 DIO1 DIO0 val
                        0    0    0    0    1    1    1    1 (f)
```

Explanation

- ➔ “AIO0” displays 4032mV, since it is set to output with maximum output level.
- ➔ “AIO1” displays 0mV, since there is no applied voltage at this input.
- ➔ “AIO0” displays 4032mV, since it is connected to AIO0.
- ➔ “AIO3” displays 0mV, since there is no applied voltage at this input.
- ➔ ”Switch selected” displays DIO0/DIO1 (first switch)

”Switch state” displays the state of switch between DIO0/DIO1
“Digital Port” shows the single states of DIO7…DIO0:
- DIO0: displays ‘1’ (always ‘1’, due the voltage supply)
- DIO1: displays ‘1’ (switch is open, so voltage at DIO0 is not passed through)
- DIO2: displays ‘1’ (output of DIO1)
- DIO3: displays ‘1’ (output of DIO1)
- DIO4: displays ‘0’ (n.c.)
- DIO5: displays ‘0’ (n.c.)
- DIO6: displays ‘0’ (n.c.)
- DIO7: displays ‘0’ (n.c.)

Info: If you try to connect DIO1 (when output is ‘1’) to one of the inputs DIO4…DIO7, you will notice no changes on the screen. The digital output is supplied by the IOcab 8444opto itself, where the maximum output is 4.096V. Due to different thresholds, the inputs DIO4…DIO7 needs higher voltages (>=4.7V) to toggle from ‘0’ to ‘1’.

Source code

The source file `xlDAIOexample.c` contains all needed functions:

Function

```
InitIOcab
```

Function Description

This function opens the driver and reads the current hardware configuration. (`xlGetHardwareConfig`). Afterwards a valid `channelMask` is calculated and **one** port is opened.

Function

```
ToggleSwitch
```

Function Description

This function toggles all switches and passes through the applied voltage at DIO0 to DIO1.

Function

```
CloseExample
```

Function Description

Closes the driver and the application.

 © Vector Informatik GmbH

4 Appendix A: Address Table

Vector Informatik GmbH

Vector Informatik GmbH
Ingersheimer Str. 24
70499 Stuttgart
Germany
Phone : +49 711 80670-0
Fax : +49 711 80670-111
info@de.vector.com
http://www.vector-informatik.com

Vector CANtech, Inc.

Vector CANtech, Inc.
Suite 550
39500 Orchard Hill Place
Novi, Mi 48375
USA
Phone : +1 248 449 9290
Fax : +1 248 449 9704
info@us.vector.com
http://www.vector-cantech.com

Vector Japan Co., Ltd.

Vector Japan Co., Ltd.
Seafort Square Center Bld. 18F
2-3-12, Higashi-shinagawa, Shinagawa-ku
140-0002 Tokyo
Japan
Phone : +81 3 5769 7800
Fax : +81 3 5769 6975
info@jp.vector.com
http://www.vector-japan.co.jp

Vector France SAS

Vector France SAS
168, Boulevard Camélinat
92240 Malakoff
France
Phone : +33 1 4231 4000
Fax : +33 1 4231 4009
info@fr.vector.com
http://www.vector-france.com

VecScan AB

VecScan AB
Theres Svenssons Gata 9
41755 Göteborg
Sweden
Phone : +46 31 764 7600
Fax : +46 31 764 7619
info@se.vector.com
http://www.vecscan.com

Vector Korea IT Inc.

Vector Korea IT Inc.
Daerung Post Tower III, 508
182-4 Guro-dong, Guro-gu
Seoul 152-790
Republic of Korea
Phone : +82 2 2028 0600
Fax : +82 2 2028 0604
info@kr.vector.com
http://www.vector-korea.com/

Vector GB Limited

Vector GB Limited
Rhodium, Central Boulevard
Blythe Valley Park
Solihull, Birmingham
West Midlands, B90 8AS
United Kingdom
Phone : +44 121 50681-50
Fax : +44 121 50681-66
info@uk.vector.com
http://www.vector-gb.co.uk

Get more Information!

Visit our Website for:

> News

> Products

> Demo Software

> Support

> Training Classes

> Addresses

www.vector.com