Software Specification

# G-API

# Quickstart Guide Version 2.1 (r8844)



GÖPEL electronic GmbH Göschwitzer Str. 58/60 D-07745 Jena Tel.: +49-3641-6896-597 Fax: +49-3641-6896-944 E-Mail: ats\_support@goepel.com http://www.goepel.com

#### Copyright © 2009 GOEPEL electronic GmbH. All rights reserved.

The software described in this manual as well as the manual itself are supplied under license and may be used or copied only in accordance with the terms of the license. The customer may make one copy of the software for safety purposes.

The contents of the manual is subject to change without prior notice and is supplied for information only. Hardware and software might be modified also without prior notice due to technical progress.

In case of inaccuracies or errors appearing in this manual, **GOEPEL electronic** GmbH assumes no liability or responsibility.

Without the prior written permission of **GOEPEL electronic** GmbH, no part of this documentation may be transmitted, reproduced or stored in a retrieval system in any form or by any means as well as translated into other languages (except as permitted by the license).

**GOEPEL electronic** GmbH is neither liable for direct damages nor consequential damages from the company's product applications.

#### **G-API:** Quickstart Guide

Version 2.1 (r8844) Published 04/28/2021

## **Table of Contents**

1 Introduction	. L
1.1 Introduction to the G-API	T
1.1.1 Features of the G-API	. L
1.1.2 System requirements	. 1
1.1.3 System structure	. 1
1.1.4 Directory structure	. 3
2 Installation	. 5
3 HardwareExplorer	. 9
3.1 Introduction	. 9
3.2 System Configuration	10
3.3 Device Configuration	11
3.4 Interface Configuration	12
3.4.1 Common Properties	12
3.4.2 CAN Interface Configuration	13
3.4.3 LIN Interface Configuration	14
3.4.4 K-Line Interface Configuration	15
3.4.5 IO Interface Configuration	16
3.4.6 Sequence Interface Configuration	17
3.4.7 MOST Interface Configuration	18
3.4.8 FlexRay Interface Configuration	20
3.4.9 Net2Run Interface Configuration	21
3.4.10 UserCode Interface Configuration	22
3.4.11 LVDS FrameGrabber Interface Configuration	23
3.5 Menu Options	25
3.5.1 Menu Bar	25
3.5.2 Tool Bar	28
3.5.3 Device Context Menu	29
3.5.4 Interface Context Menu	29
4 Building an Application	31
4.1 Introduction	31
4.2 Project Properties	31
4.3 Source Code	34
4.3.1 Include Headers	34
4.3.2 Local Variables	34
4.3.3 Open Interface	35
4.3.4 Read Firmware Version	35
4.3.5 Print Firmware Version	36
4.3.6 Close Interface	36
4.3.7 Error Handling	36
4.3.8 The Whole Example	37



# **List of Figures**

1.1	G-API architecture	. 2
2.1	Installer Welcome	. 5
2.2	License terms	. 5
2.3	Component selection	6
2.4	Installation directory	6
2.5	Finish	7
3.1	HardwareExplorer - Main Screen	9
3.2	System Configuration	10
3.3	System Configuration	11
3.4	CAN Interface Properties	13
3.5	LIN Interface Properties	14
3.6	K-Line Interface Properties	15
3.7	IO Interface Properties	16
3.8	Sequence Interface Properties	17
3.9	MOST Interface Properties	18
3.10	FlexRay Interface Properties	20
3.11	Net2Run Interface Properties	21
3.12	UserCode Interface Properties	22
3.13	LVDS FrameGrabber Interface Properties	23
3.14	Menu Bar	25
3.15	General Preferences	26
3.16	Ethernet Preferences	27
3.17	Tool Bar	28
3.18	Device Context Menu	29
3.19	Interface Context Menu	29
4.1	Project Template	31
4.2	Application Settings	32
4.3	Solution Explorer	32
4.4	C/C++ Properties	33
4.5	Add Library Files	33
4.6	Solution Explorer After Configuration	34
4.7	Assign Interface Name	35



# **Chapter 1 Introduction**

### **1.1 Introduction to the G-API**

**G–API** stands for **G**oepel electronic Application Programming Interface. It is the basic interface for accessing and configuring **GOEPEL electronic** devices.

### 1.1.1 Features of the G-API

Hardware abstraction

Uniform addressing of different hardware (e.g. USB 3060 and PXI 3060) via the same API functions and libraries

Logical Interface Names

Access to hardware interfaces via logical names

Parallel access

- · of one application to several hardware interfaces
- of several applications to different hardware interfaces
- or of several applications to the same hardware interface

Asynchronous access

- Non-blocking execution of commands
- Evaluation of responses via callback functions

### 1.1.2 System requirements

System requirements:

- Microsoft Windows  $^{\mathbb{R}}$  7
- Microsoft Windows<sup>®</sup> 10

### 1.1.3 System structure

The **G-API** consists of a multi-layer DLL hierarchy and a service that manages device access. The structure is shown in Figure 1.1, "G-API architecture". The user can access **G-API** commands over the DLLs by using the provided library and header files.





Figure 1.1 G-API architecture



### **1.1.4 Directory structure**

During installation, the following directories are created as subdirectories in the installation path:

\bin ①
\doc ②
\explorer ③
\samples ④

- *bin:* This directory contains all header and library files (\*.*h*, \*.*lib*) and the **G**-API service *G*\_Api\_Server.exe.
- *o doc:* Contains this quickstart guide and the reference manual.
- Explorer: Contains all files belonging to the G-API explorer.
- *samples:* This is the directory containing example applications as C source code.



# **Chapter 2** Installation

This chapter describes the installation process of the G-API under Microsoft Windows<sup>®</sup>.



For installing **G-API**, administrator rights are needed.

After starting the installer *g*-*api*-*Setup*-\*.*exe* (\* stands for the version number), follow the instructions given by the installer.



#### Figure 2.1 Installer Welcome

🔁 GÖPEL electronic API 1.3.2882.0	
License Agreement Please review the license terms before installing G-API.	PEL ctronic
Press Page Down to see the rest of the agreement.	
boftware License Agreement	<b>_</b>
§1 General & Definitions	
§1.1 GÖPEL electronic GmbH ("GÖPEL electronic") is willing to license the accompany Software ("Software") to You (the licensee) only if You accept all of the terms in this license agreement. If You do not agree to these terms, GÖPEL electronic will not lice the Software to You, and You should immediately delete all copies of the Software Y have in any form.	ing nse ou <b>v</b>
If you accept the terms of the agreement, click I Agree to continue. You must accept agreement to install G-API.	: the
GÖPEL electronic GmbH	
< <u>B</u> ack I <u>A</u> gree	Cancel

Figure 2.2 License terms



Please read the license terms and click on *IAGREE* for confirming that you have read and do agree to the licence terms.

🔁 GÖPEL electronic API 1.3.28	82.0	
Choose Components Choose which features of G-AP	I you want to install.	
Check the components you wan install. Click Next to continue.	t to install and uncheck the compor	nents you don't want to
Select components to install:	<ul> <li>G-API Components</li> <li>G-API Service</li> <li>Explorer</li> <li>Documentation</li> <li>Samples</li> </ul>	
Space required: 9.4MB	Description Position your mouse over a comp description.	onent to see its
GÖPEL electronic GmbH	< <u>B</u> ack	Next > Cancel

#### Figure 2.3 Component selection

Select the components to be installed. It is recommended to keep all components selected.

🔁 GÖPEL electronic API 1.3.2882.0	
<b>Choose Install Location</b> Choose the folder in which to install G-API.	GOPEL electronic
Setup will install G-API in the following folder. To install in a differe select another folder. Click Install to start the installation.	nt folder, click Browse and
Destination Folder C:\Programme\GOEPEL electronic\G-API	Browse
Space required: 9.4MB Space available: 24.2GB	
GÖPEL electronic GmbH	Install Cancel

#### Figure 2.4 Installation directory

Select a target path or confirm the default installation directory.



🔁 GÖPEL electronic API 1.3.20	882.0
	Completing the G-API Setup Wizard
	G-API has been installed on your computer. Click Finish to close this wizard.
	Execute HardwareExplorer
	Show version history
	www.goepel.com
	< Back <b>Einish</b> Cancel

Figure 2.5 Finish

In the last dialog of the installer, you have the option to start the HardwareExplorer and show the latest changes and additions in the Version History .

Make your choice and complete the installation by clicking *FINISH*.

The installer can also be run from the command line. It provides several switches for customizing the installation:

- /? : shows the options dialog
- /S : silent install
- /L : create an installation log file (install.log) in the installation directory
- /INSTDIR="C:\MyInstallDir" : installation directory
- /NODESKTOPICONS : do not create desktop icons

Usage: G-API-Setup-x.y.z.exe [/?] [/S] [/L] [/INST DIR="C:\MyInstallDir"] [/NODESKTOPICONS]



# Chapter 3 HardwareExplorer

### **3.1 Introduction**

The **GOEPEL electronic HardwareExplorer** is a tool for configuring **GOEPEL electronic** devices. It provides a graphical representation of available devices and offers possibilities for testing and configuring the device interfaces.

At each start, the HardwareExplorer scans the system for connected **GOEPEL electronic** devices. All devices that could be found are displayed in the tree on the left hand side of the screen, while further information about the selected device or interface is diplayed on the right hand side.

🔁 Goepel electronic HardwareExplorer 1.3.288	32.0		
<u>File T</u> ools <u>H</u> elp			
a 🗉 🔊 🔊			
🖃 🖳 My system	1. Interface		
Devices and Interfaces	2. Interface Type	CAN	
E CI 3051 (6189)	3. Controller Number	1	
SEQUENCE1	4. Interface Number	1	
SEQUENCE2	5. Available	True	
CAN CAN1	2. Configuration		
CAN CAN2	1. Interface Name	CAN1	
	2. Enable Automatic Configuration	False	
	3. Enable Blinking		
	4. Baud Hate [Daud]	Standard	
	5. 10 Mode		
	0. Acknowledge	The	
	1. Interface Name		
	Logical name assigned to the interface		
J			
😑 Ready			

#### Figure 3.1 HardwareExplorer - Main Screen

Every device is listed with its type, followed by its serial number in braces. Interfaces belonging to this device are listed right below with a symbol signalizing the interface type followed by the logical name of the interface.

When a device is disconnected from the system, it will appear greyed out. Nevertheless, its interface configuration can be edited and is stored.

For maintaining clarity, all disconnected devices can be hidden from the device list by pressing in the toolbar above the device list.

If you wish to delete all disconnected devices from the device list, go to **Tools**  $\rightarrow$  **Remove Disconnected Devices**. Attention! The configuration data for all interfaces of disconnected devices will be erased!



By default the HardwareExplorer scans for PCI, PXI, USB and Ethernet devices. The scan for Ethernet devices will include every IP address of every network adapter that was detected. If this behaviour is not desired, you can select which network adapter IP addresses should be scanned under **Tools**  $\rightarrow$  **Preferences**.

### **3.2 System Configuration**

The topmost configuration page shows the system configuration. It consists of the system name (the name of the computer per default) a system description and information about the **GOEPEL electronic** firmware package .

Soepel electronic HardwareExplorer 1.3.42			×
<u>File T</u> ools <u>H</u> elp			
💋 🛃 😂 🖗 象			
MySystem Devices and Interfaces Devices and Interfac	Configuration Info  Name Description Tested Firmware Package Version Installed Firmwa	MySystem 1.0.113 1.0.113	
😑 Ready			

#### Figure 3.2 System Configuration

A **GOEPEL electronic firmware package** contains a set of firmware binary files for several devices and is available as a separate installer.



### **3.3 Device Configuration**

When a device is selected in the left part of the window, the device configuration window is shown. It contains hardware and firmware information and furthermore you can select whether this device shall use a dedicated firmware version or not. To use this feature, a **GOEPEL electronic** firmware package has to be installed that matches the firmware package the **G**-API was tested with.

If enabled, the **G-API** will check for the dedicated firmware version everytime a port is opened to an interface of the device. If the firmware does not match the tested firmware, it will automatically be flashed to the dedicated version.

Information about the installed and needed firmware package can be found in the Section 3.2, "System Configuration" section.

Figure 3.3 System Configuration



### **3.4 Interface Configuration**

By selecting an interface in the device list, the interface properties are shown on the right hand side of the HardwareExplorer. Depending on the interface type, a part of the available properties may vary.

Property values displayed in **bold** letters can be edited.

### **3.4.1 Common Properties**

These properties are available for every interface type.

#### Interface Type

Type of the seleceted interface

#### **Controller Number**

Number of the controller the interface is implemented on

#### Interface Number

Number of the selected interface

Since every controller may have multiple interfaces, this value represents the interface index for the controller the interface is implemented on.

#### Available

Interface availability

True - Interface is available

False - Interface is currently not available (disconnected)

#### Interface Name

Logical name assigned to the interface

This name is used to uniquely identify the interface within your applications. Use this name when connecting to the interface with **G-API** command G\_Common\_OpenInterface.

#### **Enable Automatic Configuration**

Enable automatic configuration

If set to True, the interface configuration will be applied to the interface when calling G-API command  $G_Common_OpenInterface$ . The interface is configured with the defined property values.

For example, if you want to use a CAN interface with a baud rate other than default (= 500000 baud), you have to enter the desired baud rate value under **Baud Rate** and set **Enable Automatic Configuration** to **True**. The next time you call G\_Common\_OpenInterface with the interface name of the selected interface, the changed baud rate value will be applied.



For compatibility reasons, this feature is set to **False** by default, which means that the values you entered are not applied to the interface automatically. When using the **Automatic Configuration Feature** of the HardwareExplorer, make sure that this value is set to **True**.



Soepel electronic HardwareExplorer 1.3.2	882.0	
<u>File T</u> ools <u>H</u> elp		
📄 🖂 My system	1. Interface	
Devices and Interfaces	2. Interface Type	CAN
E PCI3051 (6189)	3. Controller Number	1
SEQUENCE1	4. Interface Number	1
SEQUENCE2	5. Available	True
CAN1	2. Configuration	
CAN2	1. Interface Name	CAN1
	2. Enable Automatic Configuration	False
	3. Enable Blinking	False
	4. Baud Rate [baud]	500000
	5. ID Mode	Standard
	6. Acknowledge	True
	1. Interface Name Logical name assigned to the interface	
\varTheta Ready		

### 3.4.2 CAN Interface Configuration

#### Figure 3.4 CAN Interface Properties

#### **Enable Blinking**

If set to **True**, an LED of the device will blink periodically. This feature is mainly used for debug purposes or for identifying a device in a multi-device-setup.

#### Baud Rate [baud]

Baud Rate value in baud

#### IDMode

CAN Identifier mode

The following values are possible:

#### Standard

Only Standard identifiers (11 Bit) are used

Extended

Only extended identifiers (29 Bit) are used

Mixed

Use standard (11 Bit) and extended (29 Bit) identifiers

#### Acknowledge

If set to **False**, no acknowledge will be sent if a CAN frame is received. This feature is used for invisibly tracing bus communication.





If set to False , the interface cannot transmit any data.

### 3.4.3 LIN Interface Configuration

Soepel electronic HardwareExplorer 1.3.288	2.0	<b>_</b>	
<u>File T</u> ools <u>H</u> elp			
🛋 🗐 🔊 🔊			
🖃 🖳 My system	1. Interface		
Devices and Interfaces	2. Interface Type	LIN	
□ • • • • • • • • • • • • • • • • • • •	3. Controller Number	1	
SEQUENCE1	4. Interface Number	4	
SEQUENCE2	5. Available	True	
CAN CAN1	2. Configuration		
CAN2	1. Interface Name	LIN1	
□ USB3080 (20070120)	2. Enable Automatic Configuration	False	
SEQUENCE3	3. Enable Blinking	False	
	4. Baud Rate [baud]	19200	
CAN3	3. Master Configuration		
CAN4	1. Enable Master Task	True	
LIN1	2. Break Time	13	
LIN2	3. Break Delimiter Time	1	
KLINE1	4. Slave Configuration		
KLINE2	1. Enable Slave Task	True	
	2. Enable Baud Rate Detection	False	
	1. Interface Name		
	Logical name assigned to the interface		
eady			

#### Figure 3.5 LIN Interface Properties

#### Enable Blinking

If set to **True**, an LED of the device will blink periodically. This feature is mainly used for debug purposes or for identifying a device in a multi-device-setup.

#### **Baud Rate [baud]**

Baud Rate value in baud

#### **Enable Master Task**

If set to True, the interface acts as the LIN Master and is able to send frame headers.

#### **Break Time**

Duration of the 'Break' that signals the start of a new LIN frame (in bit times).

#### **Break Delimiter Time**

Time between the end of the 'Break' and the beginning of the start bit (in bit times)

#### **Enable Slave Task**

If set to True, the interface is able to receive and to respond to frame headers.

#### **Enable Baud Rate Detection**

If set to True, the slave task will detect the baud rate automatically by analyzing the 'SyncByte'.



Goepel electronic HardwareExplorer 1.3.288	2.0		
File Tools Help			
My system Devices and Interfaces Devices and Interfaces SEQUENCE1 SEQUENCE2 CAN CAN1 CAN CAN2 SEQUENCE3 ID1 CAN CAN3 CAN CAN4 CAN LIN1 CAN CAN4 CAN LIN1 CAN CAN4 CAN LIN1 CAN CAN4 CAN4 CAN4 CAN4 CAN4 CAN4 CAN4 CAN4	<ul> <li>I. Interface</li> <li>2. Interface Type</li> <li>3. Controller Number</li> <li>4. Interface Number</li> <li>5. Available</li> <li>2. Configuration</li> <li>1. Interface Name</li> </ul>	KLINE 1 5 True KLINE1	
	1 Interface Name		
	Logical name assigned to the interface		
😑 Ready			

### 3.4.4 K-Line Interface Configuration

Figure 3.6 K-Line Interface Properties

Since the properties of the K-Line interface depend strongly on the type of diagnostic protocol, there are no common K-Line interface properties.

Use the **G-API** K-Line commands to configure the interface in your application. A description of these commands can be found in the **Reference Manual**.



	2.0		
Goepel electronic HardwareExplorer 1.3.288	2.0		
<u>Hie Loois H</u> elp			
🧔 💾 🕾 🔊			
My system Devices and Interfaces Devices and Interfaces SEQUENCE1 SEQUENCE2 CAN1 CAN2 SEQUENCE3 SEQUENCE3 SEQUENCE3 CAN3 CAN4		IO 1 9 True 101	
e Ready			

### 3.4.5 IO Interface Configuration

#### Figure 3.7 IO Interface Properties

The IO interface provides access to the inputs and outputs, relays and internally generated signals.

Use the  $\,{\rm G-API}\,$  IO commands to gather information about the available resources and for controlling these.



Soepel electronic HardwareExplorer 1.3.288	2.0		
💋 🗐 🔐 🔊			
🖂 🛶 Mulsustem	□ 1 Interface		
Devices and Interfaces     PCI3051 (6189)	2. Interface Type 3. Controller Number	SEQUENCE 2	
SEQUENCE2	5. Available	True	
CAN CAN1	2. Configuration		
CAN CAN2	1. Interface Name	SEQUENCE1	
	1. Interface Name		
	Logical name assigned to the interfa	ce	
😑 Ready			

### **3.4.6 Sequence Interface Configuration**

#### Figure 3.8 Sequence Interface Properties

The Sequence interface offers the possibility to record and play back command sequences.

This can be useful for example if a large number of commands have to be executed for more than one time.

You first need to allocate a sequence handle with  $\ensuremath{\mathsf{G}}\xspace\ensuremath{\mathsf{API}}\xspace$  command <code>G\_Sequence\_Replay\_Allocate</code>.

After calling command G\_Sequence\_Replay\_Recording\_Start, all commands to interfaces other than the sequence interface are recorded.

To stop recording, simply call <code>G\_Sequence\_Replay\_Recording\_Stop</code>. Now, you can control the recorded sequence by calling <code>G\_Sequence\_Replay\_Playback\_Start</code> and <code>G\_Sequence\_Replay\_Playback\_Stop</code>.

On devices providing a file system the recorded sequence can be stored permanently. The **G-API** command G\_Sequence\_Replay\_Autoplay\_Enable can then be used to enable the automatic start of the sequence at power on of the device.



Soepel electronic HardwareExplorer	1.3.2882.0		_ 🗆 X
File Tools Help			
🖂 🖂 My system	1. Interface		
Devices and Interfaces	2. Interface Type	MOST	
🖻 🗝 PCI3051 (6189)	3. Controller Number	1	
SEQUENCE1	4. Interface Number	1	
SEQUENCE2	5. Available	True	
CAN1	2. Configuration		
CAN2	1. Interface Name	MOST1	
⊡ +< USB3060 (5082009)	2. Enable Automatic Configuration	on False	
MOST1	4. Device Mode	Timing Master	
	5. Clock Mode	44100 Hz	
	<ol> <li>Logical Node Address Mode</li> </ol>	Derive From Position	
	7. Logical Node Address	0x0100	
	8. Group Address	0x03FF	
	9. Network Master Mode	Master	
	1. Interface Name Logical name assigned to the interf	ace	
😑 Ready			

### 3.4.7 MOST Interface Configuration

#### Figure 3.9 MOST Interface Properties

#### **Device Mode**

The following values are possible:

#### Timing Master

The **Timing Master** generates the system clock for the frames and blocks. The timing slaves synchronize onto the system clock.

Timing Slave

The Timing Slave synchronizes onto a given system clock.

Spy

The interface is not visible in the net but can listen to the entire bus communication.

#### **Clock Mode**

The following values are possible:

44100 Hz

The system clock is set to 44100 Hz.

48000 Hz

The system clock is set to 48000 Hz.

#### Logical Node Address Mode

The following values are possible:



Derive From Position

The logical node address is derived from the position in the MOST ring (logical node address = 0x100 + NodePosition).

Static Address

The address given in Logical Node Address is used.

#### Logical Node Address

Logical node address (only relevant if Logical Node Address Mode is set to Static Address (0x0010..0x02FF, 0x0500..0x0FEF)

#### **Group Address**

Address of the group the interface belongs to (0x0300..0x03FF)

#### **Network Master Mode**

The following values are possible:

#### Master

The interface controls the system state and manages the central registry.

#### Slave

The interface is controlled by a network master.



Goepel electronic HardwareExplorer 1.3.2882.0         File       Tools         Help         Image: State of the state of	Goepel electronic HardwareExplorer 1.3.2	882.0	
File       Iools       Help         Image: Second state of the second s	File Tools Help		
Image: Second system       Image: Second system         Image: Devices and Interfaces       Image: Image: Second system         Image: Devices and Interfaces       Image: Image: Second system			
My system     I. Interface       Devices and Interfaces     2. Interface Type	I 🗐 I 🔊 💫		
My system         I. Interface           Devices and Interfaces         2. Interface Type			
Devices and Interfaces 2. Interface Type FLEXRAY	] — My system	1. Interface	
	Devices and Interfaces	2. Interface Type	FLEXRAY
Controller Number	🖻 🕶 PCI3051 (6189)	3. Controller Number	1
SEQUENCE1 4. Interface Number 13	SEQUENCE1	4. Interface Number	13
SEQUENCE2 5. Available True		5. Available	True
CAN CAN1 2. Configuration	CAN CAN1	2. Configuration	
Lancan CAN2 1. Interface Name FLEXRAY1	CAN CAN2	1. Interface Name	FLEXRAY1
Base CLAN6153 (20100265) 2 Enable Automatic Configuration False		2. Enable Automatic Configuration	False
SEQUENCE3 3. Single Channel False	SEQUENCE3	3. Single Channel	False
UI 4. Lontroller Type MH14300		4. Controller Type	MFR4300
CAN LAN3 5. Transceiver Type IJA TUBU	CAN LANS	5. Transceiver Type	1JA1080
	ELEXBAY2		
	e osenooden		
1 Interface Name		1 Interface Name	
Logical name assigned to the interface		Logical name assigned to the interface	
😝 Ready	Ready		

### 3.4.8 FlexRay Interface Configuration

#### Figure 3.10 FlexRay Interface Properties

The FlexRay interface parameters are used to identify the communication module that is used on the device. This is necessary, because not all devices provide an automatic detection of the communication module.

#### Single Channel

If set to **True**, the **single channel mode** of the FlexRay controller is enabled. Otherwise it will be set to **dual channel mode**.

#### **Controller Type**

The following values are possible:

#### MFR4300

The FlexRay Communication Controller Module MFR4300 is used on the device.

#### MFR4310

The FlexRay Communication Controller Module MFR4310 is used on the device.

#### Transceiver Type

The following values are possible:

TJA1080

The transceiver TJA1080 is used.



Goepel electronic HardwareExplorer 1.3.288	2.0		
<u>File T</u> ools <u>H</u> elp			
🧔 📙 😂 🔊			
My system Devices and Interfaces Devices and Interfaces Devices and Interfaces SEQUENCE1 SEQUENCE2 CAN1 CAN2 Devices CAN6153 (20100265) SEQUENCE3 ID1 CAN3 CAN4 CAN5 CAN6 EUSERCODE1 SERCODE1 SERCEDE1	<ul> <li>I. Interface</li> <li>2. Interface Type</li> <li>3. Controller Number</li> <li>4. Interface Number</li> <li>5. Available</li> <li>2. Configuration</li> <li>1. Interface Name</li> </ul>	NET2RUN 1 23 True NET2RUN1	
	<ol> <li>Interface Name Logical name assigned to the interface</li> </ol>		
● Ready			

### 3.4.9 Net2Run Interface Configuration

Figure 3.11 Net2Run Interface Properties

The Net2Run interface allows signal based communication. It represents an abstraction layer, that is used to address signals on different busses by name and to translate physical values to internal values and vice versa.

There are no specific Net2Run properties to be configured within the HardwareExplorer, because Net2Run is configured with a separate application, called **Net2Run Network Runtime Configurator**.



Soepel electronic HardwareExplorer 1.3.288	2.0		
File Tools Help			لل اعبد
Image: Second	<ul> <li>I. Interface</li> <li>2. Interface Type</li> <li>3. Controller Number</li> <li>4. Interface Number</li> <li>5. Available</li> <li>2. Configuration</li> <li>1. Interface Name</li> </ul>	USERCODE 1 24 True USERCODE1	
	<b>1. Interface Name</b> Logical name assigned to the interface	ce	
\varTheta Ready			

### 3.4.10 UserCode Interface Configuration

#### Figure 3.12 UserCode Interface Properties

The UserCode interface allows access to the part of the device, where applications written by the user can be stored and controlled.

Compared to the **Sequence Interface**, the **UserCode Interface** allows more sophisticated activities. Here, an entire application can be programmed by using the command set of the **Onboard API**.

It is possible to react to specific events or states within the application. After the application is compiled, it can be downloaded to the UserCode area, where it can be started, stopped and even be debugged.

Just like a sequence, a UserCode application can be configured to start automatically after a power on reset.



Goepel electronic HardwareExplorer 1.3.2	2892.0	
File Tools Help		
😑 — My system	□ 1. Interface	
Devices and Interfaces	2. Interface Type	LVDS
□ PCI3051 (6189)	3. Controller Number	1
CAN1	4. Interface Number	1
CAN2	5. Available	True
SEQUENCE1	2. Configuration	
SEQUENCE2	1. Interface Name	LVDS1
□ USB4120 (7)	2. Enable Automatic Configuration	False
LVDS1	□ 3. Resolution	800; 480
	1. Width	800
	2. Height	480
	4. Sync Width	0; 0
	1. Horizontal	0
	2. Vertical	0
	5. Signal Levels	Signal level configuration
	1. Data Enable	High-Active
	2. HSync	Low-Active
	3. VSync	High-Active
	6. Clock Edge	Falling Edge
	7. Routing	Bit to Pin routing
	1. Colors	Color signal routing
		255; 0; 1; 2; 3; 4; 5; 6
	🗄 🗄 2. Green	255; 7; 8; 9; 10; 11; 12; 13
		255; 14; 15; 16; 17; 18; 19; 20
	1 2. Control	Control signal routing
	8. Deserializer Configuration	0xC5; 0x7F; 0x28; 0x0C
	1. Interface Name	
	Logical name assigned to the interface	
J		
😝 Ready		

### 3.4.11 LVDS FrameGrabber Interface Configuration

#### Figure 3.13 LVDS FrameGrabber Interface Properties

#### Resolution

Resolution of the grabbed image in pixels

#### Sync Width

Horizontal and vertical synchronization width

#### **Signal Levels**

Signal levels for signals Data Enable , HSync and VSync

The following values are possible:

```
Low-Active
```

The selected signal is low active

```
High-Active
```

The selected signal is high active

#### **Clock Edge**

Edge selection for the clock signal

The following values are possible:

Falling Edge

The falling edge of the clock signal will be used for synchronization

Rising Edge

The rising edge of the clock signal will be used for synchronization



#### Routing

Routing of hardware pins of the LVDS deserializer to bits of image signals, including color and control pins

For Example, if Bit 0 of the color Red is connected to the 3rd pin of the deserializer, the value for Routing  $\rightarrow$  Colors  $\rightarrow$  Red  $\rightarrow$  Bit0 has to be 3.



Unused signal bits need to be set to 255 !



### 3.5 Menu Options

### 3.5.1 Menu Bar

K	Goepel electronic HardwareExplorer 1.3.288	2.0		
	<u>File T</u> ools <u>H</u> elp			
¢	2 🛃   🖑 🖗			_
Г	∃ — My system	🗆 1. Device		
L	Devices and Interfaces	Available	True	

#### Figure 3.14 Menu Bar

The menu bar located in the upper half of the main window offers access to common features of the HardwareExplorer.

#### File Menu

File  $\rightarrow$  Open Configuration

Load configuration from file

A snapshot of a device configuration can be loaded from a file. This is useful for restoring a configuration backup or when working with different configurations for one device.



If the file contains configuration data for currently connected devices, the current configuration of these devices will be overwritten.

#### File $\rightarrow$ Save Configuration

Save configuration to file

A snapshot of the current device configuration is saved to a file.

#### $\textit{File} \ \rightarrow \ \textit{Exit}$

Closes the program

#### **Tools Menu**

#### Tools $\rightarrow$ Refresh Configuration

Scans for newly connected or disconnected devices and updates the device list accordingly.

#### Tools $\rightarrow$ Remove Disconnected Devices

Removes all devices that are currently not connected from the device list.

#### Tools $\rightarrow$ Reset G-API Service

Resets (actually restarts) the G-API Service that is used for communication with **GOEPEL elec**tronic devices.

#### $\textbf{Tools} \ \rightarrow \ \textbf{Preferences}$

Opens the preferences dialog

The preferences dialog gives access to configuration parameters of the HardwareExplorer. It is divided into categories that are shown in the list on the left side of the dialog.



#### **General Preferences**

💈 Preferences		×
General Ethernet	Device List  Hide disconnected devices  Do not scan for new devices	
	Ok Cancel	

#### Figure 3.15 General Preferences

Device List

Hide disconnected devices

If checked, all disconnected devices are hidden (not deleted!) from the device list.

Do not scan for new devices

If checked, only devices that are already in the device list are updated. When new devices are connected, they will not show up in the device list until this option is deactivated.



#### **Ethernet Preferences**

Vreferences		×
Ethemet	Scan Mode Scan IP Lis	t
	<ul> <li>☐ 172.16.4.142 255.255.0.0</li> <li>☑ 192.168.1.1 255.255.255.0</li> </ul>	IP addresses and sub net masks of network adapters that will be scanned for Goepel electronic ethemet devices using broadcast scans.
	IP List ✓ 192.168.1.61 ✓ 192.168.1.62 Remove	List with IP addresses that will be scanned for Goepel electronic ethemet devices.
		Ok Cancel

#### Figure 3.16 Ethernet Preferences

#### Scan Mode

Defines the mode for scanning for GOEPEL electronic ethernet devices.

#### Scan Network Adapters

A broadcast scan is performed for all selected ethernet adapters in the 'Network Adapters' list.

You can specify which adapters will be scanned by checking / unchecking the adapters in the 'Network Adapters' list.

#### Scan IP List

A list of user defined IP addresses that will be scanned.

IP addresses can be added to or removed from the list by using the dedicated buttons next to the 'IP List'.

You can specify which ip addresses will be scanned by checking / unchecking them in the 'IP List' list.



After the first install, all Scan Mode Options are unchecked. This means that no scan for **GOEPEL electronic** ethernet devices will be preformed.

#### Help Menu

#### $\textit{Help} \rightarrow \textit{About}$

Display the About Screen containing version and contact information.



### 3.5.2 Tool Bar

🔁 Goepel electronic Hardw	vareExplorer 1.3.3157	
<u>File T</u> ools <u>H</u> elp		
💋 🛃  🖗 🔒		

#### Figure 3.17 Tool Bar

The tool bar located below the menu bar offers one-click-access to the most common features of the HardwareExplorer.

#### ø

Load configuration from file

A snapshot of a device configuration can be loaded from a file. This is useful for restoring a configuration backup, or when working with different configurations for one device.



If the file contains configuration data for currently connected devices, the current configuration of these devices will be overwritten.

Save configuration to file

A snapshot of the current device configuration is saved to a file.



Ē

Scans for newly connected or disconnected devices and updates the device list accordingly.



Hide disconnected devices from device list.

27			١.	
8.0				
- 62			8	÷
- 54		е		
20				
80 B	-	с.		

Do not scan for new devices

When activated, only the devices that are already in the device list are updated. If new devices are connected, they will not show up in the device list until this option is deactivated.



### 3.5.3 Device Context Menu

Open the Device Context Menu by right-clicking on a device in the device list.



#### Figure 3.18 Device Context Menu

**Flash Firmware** 

Update the firmware of the device.

### 3.5.4 Interface Context Menu

Open the Interface Context Menu by right-clicking on an interface in the device list.



#### Figure 3.19 Interface Context Menu

#### Self Test

Perform a quick self test of the interface.

#### Reset Interface

Reset the interface to its initial state.

#### **Read Firmware Version**

Read the firmware version of the interface.



# **Chapter 4 Building an Application**

### 4.1 Introduction

This chapter is a step by step tutorial of building a simple application with Microsoft Visual C++ 2008 Express Edition .

When working with other development environments, these steps may differ slightly.

### **4.2 Project Properties**

After starting Visual C++, go to  $\textit{File} \rightarrow \textit{New} \rightarrow \textit{Project}$  .

This will open a project wizard, where you first have to choose a project template.

New Project			?	×
Project types:		Templates:		6-6- 6-6- 6-6-
E Visual C++		Visual Studio installed templates		
Win32		Win32 Console Application 🔄 Win32 Project		
in General		My Templates		
		Search Online Templates		
A project for crea	ting a Win32 console a	pplication		
<u>N</u> ame:	<enter_name></enter_name>			
Location:	C:\Dokumente und Einstellungen\fischer\Eigene Dateien\Visual Studio 2008\Projects			
Solution Name:	<enter_name></enter_name>	Create directory for solution		
		ок	Cancel	

Figure 4.1 Project Template

Since we want to build a console application, we choose Win32 Console Application .

Enter a project name and click OK to proceed. In this example, our project is called G-API Tutorial.

In the following dialog, click **NEXT** for changing the standard application settings.



/in32 Application Wizard - G-API Tutorial				
C:>_	ntion Settings			
Overview Application Settings	Application type:         ○       Windows application         ○       Cgnsole application         ○       DLL         ○       Static library         Additional options:       ✓         ✓       Empty project         □       Export symbols         □       Precomplied header	Add common header files for: ☐ ATL ☐ MFC		
	< Previous	Next > Finish Car	ncel	

Figure 4.2 Application Settings

Under Additional Options uncheck Precompiled Header and check Empty Project for creating an empty project.

Close the project wizard by clicking FINISH.

In the **Solution Explorer** window on the left, right click on **Source Files** and choose  $Add \rightarrow New Item...$  and enter *tutorial.c* as file name.



Figure 4.3 Solution Explorer

This will create the empty file *tutorial.c*, add it to the project and open it in the edit window.

Since our program needs to access functions contained in *g\_api\_common.dll* and *g\_api.dll*, we need to include the library information for these files.

To do this, right click on **G-API Tutorial** in the **Solution Explorer** and choose **Properties**. This will open a dialog box with the project properties.



nfiguration:  Active(Debug)	Platform: Active(Win32)	•	Configuration Manager
P Common Properties Configuration Properties General General General General General General General Generator Gene	Additional Include Directories Resolve #using References Debug Information Format Suppress Startup Banner Warning Level Detect 64-bit Portability Issues Treat Warnings As Errors Use UNICODE Response Files	*C:\Programme\GOEPEL e Program Database for Edi Yes (/nologo) Level 3 (/W3) No No Yes	lectronic\G-API\bin"

Go to C/C++ and enter the path to the G-API bin directory in the G-API directory.

Figure 4.4 C/C++ Properties

Close the properties dialog, right click on Resource Files in the Solution Explorer and choose Add  $\rightarrow$  Existing Item... .

Go to the *bin* directory of your G-API installation, select the files *g\_api\_common.lib* and *g\_api.lib* and click *ADD* to add the files to the project.

Add Existing Iten	n - G-API Tutorial		×
<u>S</u> uchen in:	🗀 bin	💌 🔾 🦻	⊳ ⊞•
Desktop Projects Arbeitsplatz	FpgaFiles  Gapi.h Gapi.lib Gapi.can.h Gapi.can.h Gapi.can.lib Gapi.common.h Gapi.can.lib Gapi.common.lib Gapi.common.lib Gapi.clig.h Gapi.flexray.h Gapi.flexray.h Gapi.flexray.rml.loader.lib Gapi.flexray.xml.loader.lib Gapi.flexray.h Gapi.flexray.xml.loader.lib Gapi.flexray.h Gapi.flexray.xml.loader.lib Gapi.flexray.h Capapi.flexray.xml.loader.lib Capapi.flexray.h Capapi.flexray.xml.loader.lib Capapi.flexray.h Capapi.flexray.h Capapi.flexray.xml.loader.lib Capapi.flexray.h Capapi.flexray.h Capapi.flexray.xml.loader.lib Capapi.flexray.h	<pre>g_api_io.lib n g_api_kline.h g_api_kline.lib n g_api_lin.h g_api_lin.lib n g_api_vds.h g_api_vds.lib n g_api_most.lib n g_api_net2run.lib n g_api_sequence.h g_api_sequence.lib m g_api_server.exe n g_api_user_code.h</pre>	g_api_user_code.lib g_devices.ini g_devices.xml g_error.h host_types.h HWESettings.dat
	•		Þ
	Objekt <u>n</u> ame: ["g_api_comm	non.lib'' ''g_api.lib''	✓ <u>A</u> dd
	Objekttyp: All Files (*.*)		Abbrechen

#### Figure 4.5 Add Library Files

If a dialog appears asking for a custom build rule simply click NO.

If your project requires access to other G-API DLL files you need to include these as well.



At this point all project properties are set and the Solution Explorer should look like this:



Figure 4.6 Solution Explorer After Configuration

### 4.3 Source Code

In this tutorial, we will build a simple application that reads the firmware version of an interface and displays it in a console window.

### 4.3.1 Include Headers

Our example application needs two header files to be included.

```
#include "g_api_common.h"
#include <stdio.h>
```

*g\_api\_common.h* is required for accessing functions of *g\_api\_common.dll*, *stdio.h* is a windows library header needed for the printf function.

### 4.3.2 Local Variables

The main function of our application has a set of local variables:

```
G_Error_t rc;
G_PortHandle_t portHandle;
char buffer[1024];
u32_t length = 1024;
```

rc

Provides the return code of a G-API function call. If the function returns without error, the value of rc is  $G_{NO}_{ERROR}$ .

portHandle

The **port handle** is used to specify the connection of the application to the interface. It is returned by function G\_Common\_OpenInterface and used for subsequent G-API function calls.

buffer[1024]

The data buffer for storing the firmware information string. In our example it is a **char** array that can hold 1024 characters.

length

The variable length is used to pass the size of our buffer to the G-API. On function return, it contains the size of the returned string in characters.



### 4.3.3 Open Interface

At first, we need to open a port to an interface. The interface is specified by its name assigned in the HardwareExplorer.

In this example we will choose a CAN interface and assign the name TUTORIAL.

Goepel electronic HardwareExplorer 1.3.2882.0				
<u>Eile T</u> ools <u>H</u> elp				
🕫 🛃   🍣 🖗				
🖃 🛶 My system	🗉 1. Interface			
Devices and Interfaces	2. Interface Type	CAN		
E PCI 3051 (6189)	3. Controller Number	1		
SEQUENCE1	4. Interface Number	1		
SEQUENCE2	5. Available	True		
TUTORIAL	2. Configuration			
CAN CAN2	1. Interface Name	TUTURIAL		
	2. Enable Automatic Configuration	False		
	3. Enable Blinking	False		
	4. Baud Hate (Daud)	Standard		
	5. 10 Mode 6. Acknowledge			
	0. Acknowledge	1146		
	1. Interface Name			
	Logical name assigned to the interface			
• Ready				

#### Figure 4.7 Assign Interface Name

The port is opened by calling G\_Common\_OpenInterface, passing the interface name and the pointer to our portHandle variable. On function return portHandle contains the port handle.

```
rc =
  G_Common_OpenInterface(
    "TUTORIAL",
    &portHandle
);
```

### 4.3.4 Read Firmware Version

To read the firmware version of the interface we need to call G\_Common\_GetFirmwareVersion, passing the **port handle**, a pointer to our buffer and a pointer to our length variable.

Note that length is initialized with the size of our buffer on definition! On function return, buffer will contain the firmware version string and length will contain the length of the string in characters.



```
rc =
  G_Common_GetFirmwareVersion(
   portHandle,
   buffer,
   &length
);
```

### 4.3.5 Print Firmware Version

A simple printf command is used to print the firmware version string to the console window.

```
printf("%s\n", buffer);
```

### 4.3.6 Close Interface

After the firmware version has been read we need to close the port to the interface.

```
rc = G_Common_CloseInterface(portHandle);
```

All resources of the connection are freed and the port handle becomes invalid.

Subsequent calls of G-API functions will need a new port handle that can be retrieved with  $\tt G\_Common\_OpenInterface$ .

### 4.3.7 Error Handling

It is important to check if a G-API function call is successful by validating the return code rc.

```
if (rc != G_NO_ERROR) {
  printf("%s\n", G_GetLastErrorDescription());
  return;
}
```

If the return code is not <code>G\_NO\_ERROR</code>, a description of the last <code>G-API</code> error is queried with <code>G\_Get-LastErrorDescription</code> and printed to the console window. After that the program is closed.



### 4.3.8 The Whole Example

```
#include "g api common.h"
#include <stdio.h>
void main(void) {
G_Error_t rc;
G_PortHandle_t portHandle;
char buffer[1024];
u32_t length = 1024;
 // open port to interface
rc =
 G_Common_OpenInterface(
  "TUTORIAL",
  &portHandle
 );
 if (rc != G_NO_ERROR) {
 printf("%s\n", G_GetLastErrorDescription());
 return;
 }
 // read firmware version
rc =
 G_Common_GetFirmwareVersion(
  portHandle,
  buffer,
  &length
 );
 if (rc != G_NO_ERROR) {
 printf("%s\n", G_GetLastErrorDescription());
 return;
 }
 // print firmware version to console window
printf("%s\n", buffer);
 // close port
rc = G_Common_CloseInterface(portHandle);
if (rc != G_NO_ERROR) {
 printf("%s\n", G_GetLastErrorDescription());
 return;
 }
}
```

