Berghof IMX and Pi control platform in CODESYS V3

BERGHOF

Disclaimer

The content of this publication was checked for compliance with the hardware and software described. However, discrepancies may arise, therefore no liability is assumed regarding complete compliance. The information in this document will be checked regularly and all necessary corrections will be included in subsequent editions. Suggestions for improvements are always welcome.
Subject to technical changes.

Trademarks

→   CANtrol® and CANtrol®- dialog are registered trademarks of Berghof Automation GmbH

→   Microsoft®, Windows® and the Windows® Logo are registered trademarks of Microsoft Corporation in the USA and other countries.

→   EtherCAT® is a registered trademark and patented technology, licensed from Beckhoff Automation GmbH, Germany.

→   CiA® and CANopen® are registered community trademarks of CAN in Automation e.V.

All rights reserved by the individual copyright holders.

General Information on this Manual

This equipment manual contains product-specific information valid at the time of publication.
This equipment manual is only complete in conjunction with the product-related hardware and software user manuals required for the individual application.

You can reach us at:
Berghof Automation GmbH
Arbachtalstrasse 26
72800 Eningen
Germany
T +49.7121.894-183
F +49.7121.894-100
e-mail: support-controls@berghof.com
www.berghof-automation.com

Berghof Automation GmbH works in accordance with DIN EN ISO 9001:2000.

Change log

Version	Date	Subject
1.0	26.06.2018	Non-existent
1.1	25.09.2018	Linguistic revision
1.2	30.09.2022	Update nomenclatur / redesigned
1.3	16.06.2023	Control portfolio updated (Pi added) Version overview updated Web interface updated Target package installation updated Software options updated Cyber Security added Multi-Core quick start added Update of the controller added USB Update updated Type plate updated
1.4	15.07.2024	Web interface updated
1.5	14.04.2025	Version overview updated Web interface for ET updated

Contents

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

5

1.   General information

This system manual is intended for qualified personnel and contains information on the CODESYS V3 development environment when used with Berghof controllers of the MX6 platform. The information in this document is subject to change without notice.

1.1.   Qualified personnel

Installation, commissioning, and maintenance of programmable logic controllers of the Berghof MX6 plat-form requires qualified personnel.
Qualified personnel within the meaning of this documentation and the safety instructions contained therein are trained specialists who have the authority to assemble, install and commission devices, sys-tems and circuits in accordance with the standards of safety technology and who are familiar with the safety concepts of automation technology.

1.2.   Hazard categories and signal terms

The signal terms described below are used for safety instructions that you must observe for your personal safety and to prevent property damage.

The signal terms have the following meaning:

Symbol	Meaning
→ …	List entry
▶ …	Individual instructions or subitem of the list entry
1. … 2. …	List entries that must be executed in the specified order.
i	Additional product information

1.3. Hazard categories and indications

The following indications are used in the case of warning messages so as to ensure your personal safety and avoid any damage to property.

The indications have the following meanings:

⚠ **DANGER**

Immediately imminent danger

Failure to observe the information indicated by this warning may result in death, serious injury or extensive property damage

⚠ **WARNING**

Imminent danger

Failure to observe the information indicated by this warning may result in death, serious injury or extensive property damage.

⚠ **CAUTION**

Danger

If you do not follow these instructions, there may be personal injury or property damage.

ℹ **NOTE**

No danger

Additional information on the product is available here.

1.4.   Qualified personell

Qualified personnel within the meaning of this documentation and the safety instructions contained therein are trained personnel who are familiar with the safety concepts of automation technology and are authorized to process and commission software packages such as the product specified here.

1.5.   Intended use

This system manual refers to the products of the Berghof MX6 control platform, an Ethernet-based, modular automation system for industrial control applications.
For the proper commissioning and use of the individual devices of the MX6 control platform, please refer to the user manuals of the individual devices.

The automation system is intended for use within overvoltage category I (IEC 364-4-443) for the control and regulation of machinery and industrial processes in low voltage installations where the rated supply voltage does not exceed is 1,000 VAC (50/60 Hz) or 1,500 VDC.

Qualified project planning and design, proper transport, storage, installation, use and careful maintenance are essential to the flawless and safe operation of the automation system.
The automation system may only be used within the scope of the data and applications specified in this documentation and the associated user manuals.

Use the automation system only as follows:
→   as intended
→   in perfect technical condition
→   without arbitrary changes
→   exclusively by qualified users

Observe the regulations of the professional associations, the technical inspection association, the VDE regulations or corresponding national regulations.

Safety-related systems
The use of PLC controls in safety-related systems requires special measures. If a PLC control is to be used in a safety-oriented system, the user should obtain extensive advice from the PLC manufacturer, in addition to any standards or guidelines for safety-related installations that may be available.

⚠ WARNING

As with any electronic control system, failure of certain components can result in an unregulated and / or unpredictable operation.

All types of system level failures and associated backups should be considered. If necessary, the manufacturer of the automation system should be consulted.

2. Introduction

2.1. Purpose of this document

This system manual helps in using a Berghof controller EC Slim, EC Compact and Dialog Controller device family together with the **CODESYS V3** development environment from 3S-Smart Software Solutions GmbH. This includes:

→ The connection and configuration of the controller.
→ The installation of the required software.
→ Creating and compiling a project.
→ The integration of additional libraries.
→ Monitoring and debugging running programs.

This manual serves as a reference work and guide to frequently used workflows or applications, but due to the large subject area makes no claim to completeness. Rather, this manual is intended to help you help yourself and therefore refers to other documents or the online help of CODESYS V3 for some topics. Further information can be found in the chapter 6.3.

The examples and assistance shown in this manual may contain errors, may be out-dated or incomplete. For this reason, in particular, program code cannot be adopted unchecked, but must be carefully tested after integration. The user is responsible for the security of his program, the controller and the connected devices.

Suggestions for improvement and changes as well as suggestions for mistakes shall of course be gladly welcome.

> ⚠ **CAUTION**
>
> This manual requires knowledge in the area of programming programmable logic controllers (PLC) according to EN 61131-3 in Structured Text.

2.2. Terminology

To ensure consistent use of terms, they are defined in the following table.

Term	Synonyms	Definition
Control	PLC, Programmable Logic Controller, PLC(s), Programmable logic controller, CODESYS V3 Runtime	A programmable logic controller. In this documentation, the CODESYS V3 Runtime is sometimes referred to as a controller. This software PLC runs on the operating system and can be started or stopped independent of it.
Application	App program, PLC program, control program	A CODESYS V3 object of type "application" including all subordinate objects and resources.
CODESYS V3	Development environment, Programming environment, IDE(s)	The development environment CODESYS from version 3.x.
Target	Device description	Archive with device descriptions and component libraries. Is needed by CODESYS V3 to be able to connect to a Berghof controller.
System manual	-	This document.
User manual	-	Hardware-specific manuals for the respective devices of the Berghof IMX and Pi platform

Term	Synonyms	Definition
I/O	I/Os, E/A, E/As	Abbreviation for input / output in English or input / output in German.
B-Fortis Compact Control	CC, CC-Lite, CC-Slim, CC-Prime, CC-DIO, CC-AIO, ECC, ECC22xx, ECC2200, ECC2220, ECC2250, ECC2100 Slim, ECC2110 Slim, ECC21xx, ECC2100, ECC2110, Slim, Box Controller	General term for the device family with integrated industrial interfaces, digital and analog I/Os in a compact all in one form factor. Also known by its old name Ethernet Compact Control (ECC2xxx).
B-Nimis Modular Control	MC-Pi Pro, MC-Pi Prime, MC-Pi Plus, MC-Pro, MC-I/O, EC2000, EC1000, E-I/O	General term for the device family in a highly flexibale modular design which includes PLCs, digital and analoge I/O. Also known by its old name Ethernet Control (EC2000/1000) or E-I/O.
B-Primis Dialog Control	DC-Pro, DC-Prime, DC, Display Controller, DC2000, DC2xxx, DC2004, DC2007, DC2110, DC2115	General term for the PLC device family with integrated screen, industrial interfaces, digital I/Os and analog inputs. Also known by its old name Dialog Controller (DC2xxx).
B-Primis Ethernet-Terminal	ET-Pro, ET-Prime, ET, Web-Terminal, Display, ET2000, ET2xxx, ET2004, ET2007, ET2115, ET1000, ET1003, ET1005, ET1010	General term for the device family of ethernet based control panel. Pure display devices with VNC client and web browser, no control functionality. Also known by its old name Ethernet Terminals (ET2xxx/1xxx).
Powertrack	Vehicle control, segment control, switch controller, FZ, SEG, WST, Retrofit	General term for the device family of controllers for intra logistics applications with SC-CAN. There is a separate system manual for the Powertrack device family.

2.3. Supplementary documentation

This manual refers to existing documentation in many sections. Carefully read the **User's manual** of your Berghof equipment before connecting or commissioning the controller. The user manual contains the technical specification of the control, installation, and assembly instructions, as well as information about the voltage supply and existing data connections.

→ **No user manual at hand?**
Download the manual for your controller from www.berghof-automation.com. Select the menu item "Downloads" and register in the category "Automation Technology". Now, categorized per product type, there is a list of all Berghof device. Click on the desired product and a dropdown menu will open and provide matching documents for download.

The second source for further assistance is the **Online Help** of CODESYS V3 (chapter 6.3).

Finally, the **questions and answers** web page should be referred to as frequently asked questions are answered there in a short form.

→ The questions and answers (FAQ) can be found at:
https://www.berghof-automation.com/en/automation-technology/faq

2.4. Schematic overview of the hardware components
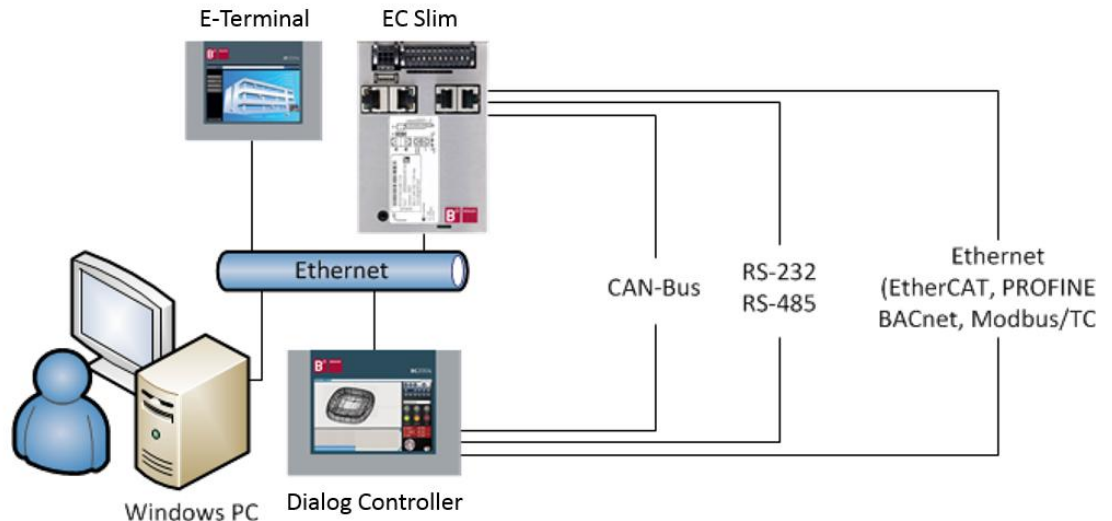
Picture 1: Schematic 1.1

2VF100684DG00.cdr

This picture shows how the hardware components are connected. All controllers and Ethernet terminals are connected via the first **Ethernet interface (eth0)** to the Windows PC on which CODESYS V3 is running. Controllers can be connected to an existing 10 or 100Mbit/s standard Ethernet network or connected directly to the network card of the PC.

All devices of EC Slim, EC Compact and Dialog Controller families also have additional data connections: A second **Ethernet interface (eth1)** that is configured as an **EtherCAT™ master** interface, but also supports the **EtherNetIP**, **Profinet™ Device**, **OPC UA**, and **Modbus/TCP** protocols. Protocols that do not require exclusive access to an Ethernet interface, such as EtherCAT™ or Profinet™ Device, can also use the **Ethernet interface (eth0)**. It is also possible to use different protocols over an interface if the protocol does not require exclusive access to the interface:
→   **RS-232** and **RS-485** as serial interfaces.
→   **CAN interface** with up to 1Mbit/s.
→   **USB** for the integration of USB storage media with FAT/FAT32 formatting.
→   **MicroSD card interface** for connecting microSD memory cards (up to 32GB).

Further information on the connections including assignment can be found in the chapter "Data connections" in the user manual of the respective controller.

2.5. Schematic overview of the software components
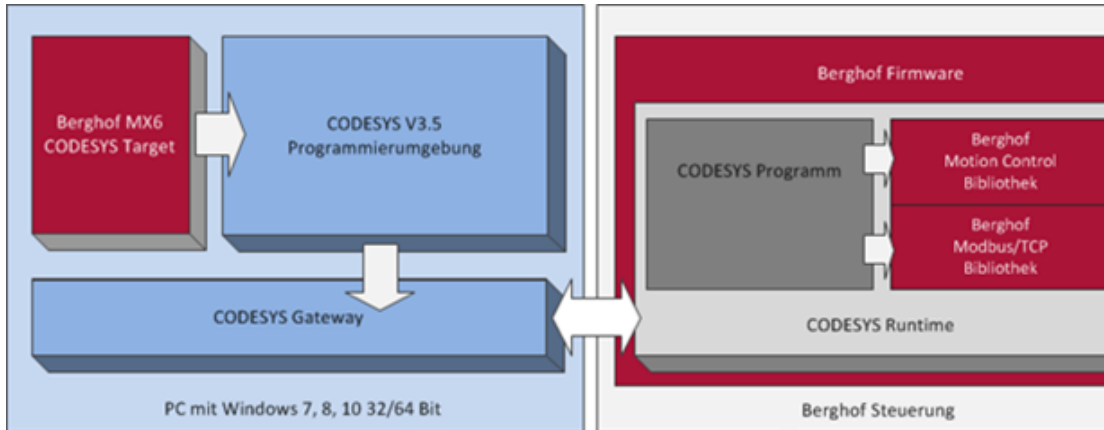
Picture 2: Schematic 1.2

2VF100685DG00.cdr

This figure shows a schematic overview of the software components involved. The representation is greatly simplified and not complete but illustrates the dependence of the terms used in this manual. The glossary and further definitions can be found in the chapter 2.2.

The left figure is a Windows PC on which the **CODESYS V3 development environment** is running. With the help of the **CODESYS V3 Gateway**, the development environment can detect controllers connected via Ethernet and connect to these controllers. The so-called **target** is necessary so that a controller can be used in this way. Components of the target are device descriptions for the PLC, device descriptions for I/Os and device-specific libraries. Without this target, CODESYS V3 will not be able to recognize or use the connected controller.

The figure on the right shows the controller on which a custom **firmware**, consisting of a real-time capable Linux and various hardware drivers, is running. In Linux, the **CODESYS V3 Runtime** is executed, the actual PLC. This software can execute and monitor individual control programs. The executed CODESYS V3 program has access to various standard libraries or additional libraries offered by Berghof that extend the functionality of the controller.

Further information on the required software components and the compatibility of the versions can be found in the chapters 5.2 and 5.3 Installation.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

17

3.   Commissioning

3.1.   Minimal required hardware

To be able to use a Berghof controller, at least the following hardware is required:
→   A Berghof controller of the MX6 or Pi control platform.
→   A PC or notebook with Windows 7, 8, 10 or 11 with 10/100 Mbit/s compatible Ethernet network card.
→   A power supply with 24V/2A DC.
→   A network cable with 10BaseT/100BaseTX RJ-45 Plug.
→   Optional: A FAT/FAT32 formatted USB memory with at least 512MB of memory. The USB memory is only needed if you want to perform a firmware update without Ethernet access or change the configuration of the controller without Ethernet access.

3.2.   Connection of the controller

Before you start the connection and configuration of the controller, read the user manual for the controller. It contains information about connecting the controller to the voltage supply as well as the assignment of the inputs and general information on operating the controller. The green power LED indicates that the controller is properly connected to the voltage supply. Further information on the status displays of the controller can be found in the **user manual** of the respective controller.

3.3.   Configuration and diagnostics options

The Berghof controllers offer extensive configuration and diagnostics options with which they can be adapted to different requirements. Basically, the controllers are connected via **Ethernet** to a PC or note-book. Since the CODESYS V3 development environment supports only Windows, this manual assumes that the user is using a PC running Windows. After the user has connected the controller and configured the network, he can either use a browser to call up the configuration interface of the controller or log on to the controller via the **ftp** protocol.
More information about the web interface can be found in chapter 4, Information on login via ftp access in chapter 8.1.

3.3.1. Diagnostics via status LEDs of the controller

If the rare case occurs that a controller does not react after switching on and you cannot connect to it, there is the possibility to read the system status of the LEDs. The LED signalling is based on the system and CODESYS states. If the CODESYS V3 Runtime is not active, the firmware controls the LEDs. As soon as the CODESYS V3 Runtime is active, the LEDs are operated exclusively by the CODESYS V3 Runtime.

System states

RUN/STOP LED (green/red/yellow/off)	Definition
off	Bootloader active
off	Linux Boot process active
yellow	System is supplied with undervoltage (from FW 1.6.0 or later)
yellow 1x flash, 2s break	Maintenance work mode active
yellow steady flashing slowly (1s)	Package update via USB active
yellow steady flashing fast (400ms)	Device is running out of RAM (firmware update active)
yellow 2x flash, 2s break	Device requires reboot (for example as ready message after USB update)
red or green	For PLC devices: CODESYS states see CODESYS operating states
green	For Visu terminals: Visu (VNC/WEB) is active

CODESYS V3 Runtime operating states

RUN/STOP LED (green/red)	ERROR LED (red)	Definition
red	off	At least one PLC application is stopped
red	off	All PLC applications are stopped
green	off	All PLC applications are running
red	red	At least one PLC application stopped because of an error
red flashing	-	Run preparation for RESET COLD (via switch/PG)
?	off	Wait for peripherals (e.g. retain / undervoltage, Ethernet initialization, etc.)

3.4. Access via Ethernet

Connect the controller to your PC or notebook via an Ethernet network. You can connect both devices to a network switch or connect the two devices directly with a network cable. Make sure the network cable is plugged in properly. For controllers of the Dialog Controller family, and EC Slim, the first Ethernet port is marked as **X4**. Controllers of the EC Compact family can use one of the **X10**, **X11** or **X12** marked Ethernet ports as these are internally coupled to a switch.

> **i   NOTE**
>
> It is recommended to use (non-crossed) network cables as they will work in any case. The controller's Ethernet interfaces are either connected to a switch or additionally support autocross, thus eliminating the need for crossed cables.

To access a controller in the default state or with an unknown network configuration, start the controller in configuration mode. See the next section for more information.

3.4.1. Access to a controller with unknown configuration

If you need access to a controller that has an unknown network configuration, start the controller in configuration mode and use the factory IP address (169.254.255.xx) of the controller to connect. The exact procedure will be described in detail in the next chapters.

3.4.2. Starting the controller in configuration mode (Maintenance Mode)

To start the controller in configuration mode, perform the following steps:
1.   Disconnect the controller from the power supply.
2.   Now press the function button S1 of the controller and keep it pressed.
3.   Turn on the controller by connecting the power supply.
4.   Hold down the function button until the status LED (Start / Stop) flashes every two seconds.
5.   The controller is now in configuration mode.

The position of the function button and the status LED can be found in the user manual for your controller in the section "Operation". Alternatively, the position of the function switch and the status LED are shown on the printed label on the respective controllers.
The configuration mode differs from the normal operation mode in the following points:
→   The CODESYS V3 Runtime is not executed, so no control programs are running. This also means that the display screen remains dark in case of display controls because no visualization is running.
→   In the network configuration, the default settings are additionally loaded, so that each controller can be addressed via a clearly defined IP address. Information about this network configuration can be found in chapter 3.4.3.
→   The text "(maintenance)" appears instead of "(plcactive)" under the name of the controller in the web interface.

20

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
01-0099-82-00-00-000 | Berghof_Control_Systemmanual_1.5_EN.docx

3.4.3. Network settings in the default state or configuration mode

Each controller in the default state or configuration mode has a static IP address according to a defined scheme. In configuration mode, the controller can be reached over its default IP **additionally** to it's already configured Ethernet configuration. In default, the controller can only be accessed via this IP address.

→ IP address: 169.254.255.xx
→ Subnet mask: 255.255.255.0

The last two digits of the controller serial number are determined by the digits in the last block of the IP address.

Serial number of the controller ends with	IP address of the controller
0x	169.254.255.x
xx	169.254.255.xx
00	169.254.255.100

The serial number of the controller can be found on the label glued on the side or on the back of the controller. The last two digits of the serial number are underlined by way of example in red in the figure. For example, the IP address of this device is 169.254.255.45.
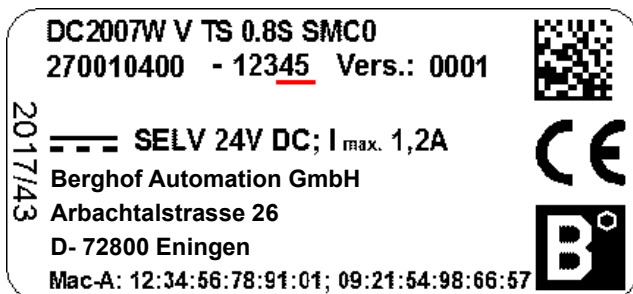
2VF100686DG00.cdr

3.4.4. Customize network settings of Windows

To connect to a controller in the default state or in configuration mode, you must adapt the network settings of the PC connected to the controller. The following example refers to Windows 7. For other Windows versions, the steps need to be adjusted accordingly.

Please note that you generally need administrator rights to configure the network under Windows. If you cannot customize your network configuration yourself, please contact your system administrator.

Please proceed as follows:

1. Open the overview of the network connections.
    (Control Panel \ Network and Internet \ Network Connections)
2. Open the properties of the network card connected to the controller.
3. Select the item Internet Protocol Version 4 (TCP / IPv4) and press "Properties".
4. Set an IP address and subnet mask compatible with the controller.

The recommended subnet mask is 255.255.255.0 and an IP address in the range 169.254.255.101 to 169.254.255.254, then the PC is in the same subnet as the controller, but a collision of the IP addresses is excluded.

Allgemein

IP-Einstellungen können automatisch zugewiesen werden, wenn das Netzwerk diese Funktion unterstützt. Wenden Sie sich andernfalls an den Netzwerkadministrator, um die geeigneten IP-Einstellungen zu beziehen.

○ IP-Adresse automatisch beziehen
◉ Folgende IP-Adresse verwenden:

IP-Adresse:	169 . 254 . 255 . 253
Subnetzmaske:	255 . 255 . 255 . 0
Standardgateway:	. . .

○ DNS-Serveradresse automatisch beziehen
◉ Folgende DNS-Serveradressen verwenden:

Bevorzugter DNS-Server:	. . .
Alternativer DNS-Server:	. . .

☐ Einstellungen beim Beenden überprüfen

Erweitert...

OK   Abbrechen

Picture 3: subnet mask

# 4.	Configuration via web interface

Configuration via the web interface is the easiest way to check or change many settings of the device.

> **i NOTE**
>
> The display devices of the E-Terminal device family also have the web interface, but to a lesser extent due to the lack of control functionality.

To call the web interface, open a browser and enter the https://"IP address" of the controller in the address bar of the browser.
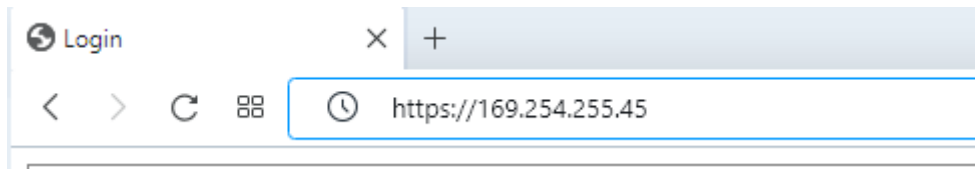
Depending on the browser you will receive a warning because of an unknown connection, which you must trust manually inside the browser.

> **i NOTE**
>
> Depending on the installed firmware version, your controller might not support https connections yet. If this is the case, use http://"IP address" as the URL for the webinterface.

If you now see a login window with the Berghof company logo, then the network is configured correctly, and the controller can be used. If you cannot call the web interface, some network settings may be wrong. Information about the configuration of the network can be found in the chapter 3.4.

There are several user accounts on the controller but only one has access to the web interface. The default password is printed on the device label of the controller. The following user have access to the web interface:

User name	Password	Rights
Admin	printed on label	Full access

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

23

> **ℹ NOTE**
>
> If you have the case that your controller does not have a default password printed on the device label it uses our old deafault password system where the passwords match the username.
> The default login in this case would be user: admin / password: admin.

> **⚠ CAUTION**
>
> For controllers that are used productively or in a network, change all passwords for existing users. Otherwise, easy access to the controller can be achieved by using the default password!

4.1. Configuration interface: Configuration

4.1.1. Menu item "Network"

On this page different network settings of the controller can be adjusted.

> **ℹ NOTE**
>
> The network settings are always only activated after a restart of the device.

Host name

The hostname is a unique name of the controller. The host name is used as the **device name** in CODESYS V3 and as the **computer name** in Windows. Host names consist of one or more labels that are separated by a point. A label consists of one or more characters.

Labels can be up to 24 characters long and can only consist of the ASCII characters:

→ a–z or A–Z (uppercase and lowercase letters are not distinguished),

→ Digits 0–9,

→ Hyphen / minus sign –

Other characters are not allowed according to RFC952 and can cause problems. The host name may also be a Fully Qualified Domain Name (FQDN) such as *plc24.mycompany.de*.

> **ℹ NOTE**
>
> Please note that there are no DNS or WINS services running on the controller, so for this reason the controller cannot be addressed by the host name without additional configuration (for example, an entry in the DNS server of the network).

Default Gateway

If the controller is to be connected to the Internet, the IP address of the router or gateway must be entered here. If the controller is only used in the local network, the entry can be set to 0.0.0.0 (default setting). The gateway settings apply only to the first network interface (eth0).

DNS Server

If the controller is to have access to the Domain Name System (DNS), i.e. if other hosts are to be addressed by their name instead of by IP address, then at least one valid DNS server must be entered here. The second DNS server serves as a fall-back if the first server is unreachable. If no DNS is to be used, both entries can be set to 0.0.0.0 (default setting).

ETH0 and ETH1

On Linux, eth0 and eth1 are the device names of the two network interfaces of the controller. Both interfaces can be largely configured and used independently of each other. In the default state, the first network interface eth0 is configured with a static IP address (Mode: static) and the second network interface eth1 is configured as EtherCAT™ device (Mode: ethercat).

ℹ NOTE

When using a multicore PLC setting eth0 or eth1 to "ethercat" will lock the interface to the last CPU core of the PLC for the best performance (recommended for SoftMotion). If this core isolation for EtherCAT is not required set the interface settings to an unconfigured mode like "dhcp" or profinet device" as it is still possible to run EtherCAT with these settings.

ETH0:1 and ETH1:1

Virtual extensions of the eth0 and eth1 network interfaces, allowing to set a second static IP address for the respective network interface, e.g. for a separate service access. Can only be operated in the mode: static and the IP must have a different IP range than the basic interface. Cannot be activated if ethercat mode is active on the basic interface.

Network Mode: inactive

In this mode, the network interface is completely deactivated.

Network Mode: static

In this mode, a static IP address can be configured. The static IP address and the netmask (also called NetMask or Subnetmask) are needed. Use this mode even if you want to use the interface as a BACnet, Ethernet/IP or Modbus/TCP interface.

Network Mode: dhcp

In this mode, the network interface can be automatically assigned an IP address from a DHCP server when the controller is started. The IP Address and NetMask fields can be set to 0.0.0.0 (default setting).

Network Mode: ethercat

In this mode the network interface is configured as EtherCAT™ device. It is recommended to use the eth1 interface for EtherCAT™. In CODESYS V3, the interface selected as EtherCAT (for example, "eth1") must then also be set as the bus interface in the EtherCAT™ master configuration.

Network Mode: profinet device

In this mode, the network interface is configured as a Profinet™ device. It is recommended to use the eth1 interface for Profinet™ device. In addition, in CODESYS V3, in the Ethernet configuration, the interface selected as the Profinet™ device (for example, "eth1") must be set as the bus interface. If Profinet™ device is to be operated together with EtherCAT™, it is recommended to use the eth0 interface for Profinet™ device together with a statically set second IP on the eth0:1 interface.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

25

> ## ⚠ CAUTION
>
> If network interfaces are to be used simultaneously in network mode static or dhcp, then the interfaces must be located at least in different subnets; otherwise, the second network interface cannot be reached. It is recommended to use two network interfaces in static or dhcp mode only if they are in different (physically separate) networks.

Examples of incorrect configurations:

Network interface	IP address	NetMask
Eth0	169.254.255.11	255.255.255.0
Eth1	169.254.255.12	255.255.255.0

Network interface	IP address	NetMask
Eth0	10.0.1.11	255.255.0.0
Eth1	10.0.2.12	255.255.0.0

Examples of working configurations:

Network interface	IP address	NetMask
Eth0	169.254.255.11	255.255.255.0
Eth1	169.254.254.12	255.255.255.0

Network interface	IP address	NetMask
Eth0	10.1.255.11	255.255.0.0
Eth1	10.2.254.12	255.255.0.0

4.1.2. Menu item "CAN"

The settings on this page make it possible to activate the CAN interfaces outside of CODESYS V3 with a certain baud rate. In the default setting "set by codesys" the interface remains inactive until the initialization of the interface has taken place from the CODESYS V3 application.
If one of the selectable baud rates is set, the CAN interface is set active during system start-up independently of a CODESYS V3 application. Access to the interface from the application is still possible. Primarily, this setting is required for the service channel functionality of the SC-CAN interface on Powertrack controllers.

4.1.3. Menu item "Time and Date"

On this page you can set the time for the Real-Timeclock (RTC) of the controller and set time zones. By default, the time zone on Berghof controller is set to Coordinated Universal Time (UTC). The feature of this setting is that the UTC time zone is the RTC of the controller. Changing the UTC time value will also change the RTC. On delivery, the RTC of the controller is set to the current German time.
This information is important if you want to set a time zone in the controller to use functions such as the automatic change between winter and summer time. Before the new time zone is selected, the RTC/UTC must be set from the pre-set German time value to the actual UTC time value. After the correct setting of the RTC/UTC, the time including changed time zone is output correctly.
Please note that some functions of the CODESYS V3 system libraries read the time as RTC (UTC).
If the time including time zone is to be given, the read time must be converted into local time.

Additionally, it is possible to set the system time via the Network-Time-Proticol (NTP).
When activated the PLC will check the configured IP for an NTP-Server during bootup and sync the time.
After the first sync the PLC will update the time once every 24 hours.

For Pi devices it is also possible to sync the RTC and not only the system time as well as to configure up to 3 NTP-Servers.

4.1.4. Menu item "Display" (only devices with display)

On this page various settings concerning the display can be changed.

Setting	Description
Brightness	The brightness of the display can be adjusted to eight different levels.
Touch Calibration	The previous calibration data of the touchscreen can be deleted.
Splash-Screen	An image can be uploaded in PNG format, which will be shown on the display when the controller is started.
Rotation	Rotate the displayed image counterclockwise in set degrees.
Touchscreen Mode	Only for devices with capacitive display Changes the touchscreen mode between single and multi touch. When set to multi the the multitouch option in the Visualization Manager inside the CODESYS project must be activated.
Visualization type (Pi)	None/vnc: Display interface is deactivated, target visu can be shown over VNC-Client Target: Shows target visu over the display interface Web: Shows web visu over the display interface via an integrated browser
Show cursor timeout (Pi)	Timeout in seconds until mouse cursor becomes invisible
Screensaver timeout (Pi)	Timeout until the screensaver is activated on the display interface

⚠ CAUTION

Enabling the web visualization type causes additional CPU-load.

The integrated browser used to display the web visu locally needs a lot of system ressources and can cause jitter in the CODESYS application. It is not recommended to use this option in highly cycle time sensitive applications like Softmotion.

4.1.5. Menu item "VNC server" (only devices without display)

This page allows you to activate or deactivate as well as to set the resolution and color settings for the controller's internal VNC server. Depending on the connected E-terminal, the values should be changed accordingly. If images with an alpha channel (such as transparent PNGs) are used for the visualization, 32Bit is recommended as the color depth. If best best performance is a priority, 16Bit is the recommended color depth.

Target system	Resolution in pixels (native)
ET-Pro 4	480x272
ET-Pro/Prime 7	800x480
ET-Pro/Prime 10	1280x800
ET-Pro 15	1366x768
ET-Prime 15	1920x1080

It is also possible to activate a password protection for the VNC server.

4.1.6. Menu item "VNC client" (ET only)

This page allows you to configure the connection settings to a VNC-Server when an E-Terminal device is in VNC-Client mode.

Setting	Description
VNC-Server	IP-adress of the VNC-Server which the ET should connect to.
Port	Network port of the VNC-Server which the ET should connect to.
Quality	Image quality settings for the displayed VNC connection. Lower quality settings reduce bandwith and load on the ET.
Screen scaling	Option to scale the resolution of the VNC-Server down to fit the ET. No scaling: displays the VNC-Server in native resolution Keep aspect ratio scales the VNC-Server image down to fit the ET resolution and keeps the native aspect ration Scale to screen scales the VNC-Server image down to fit the ET resolution in full screen disregarding the original aspect ratio
Lifegurading	Lifeguarding mode to check VNC-Server connection Ping LG: default lifeguarding via a network ping Berghof VNC lifeguarding: legacy lifeguarding mode that can only be used with older Berghof CODESYS V2.3 controllers

Additionally, is it possible to set a password for a VNC-Server that has an active password protection.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

29

4.1.7. Menu item "FTP server"

On this page, the controller's internal FTP server can be enabled or disabled. The FTP-Server uses TCP-Port 21. The following users can log on to the FTP server:

Username	Start directory	Start directory change	Rights
root	/root	Yes	Read/Write
ftpuser	/flash/ftpupload	No	Read/Write (only ftpupload)
ftpadm	/flash/ftpupload	Yes	Read/Write (only ftpupload)
ftpreader	/flash/ftpupload	No	Read
ftp custom users	Configurable	Configurable	Configurable (ftpreader/ftpuser/ftpadm)

Additionally, it is possible to activate a secure FTP server (FTPS) as well as a brute force detection.

4.1.8. Menu item "SSH server"

On this page, the controller's internal SSH server can be enabled or disabled.
Additionally, it is possible to activate a brute force detection and to configute the conncetions attemps and lockout time in seconds after the configured attempts are reached.
Only the root user can access the controller via SSH.

4.1.9. Menu item "Modbus server" (ET only)

Activates an optional Modbus TCP server on ET devices, which allows remote control of the screen brightness and beeper via controllers that support Modbus TCP client.

The control values can be set via a Modbus TCP Client – Function Code 16 (write multipe registers) with start address 0x0000 and length 4

0x0000 – Brightness of screen while in active mode – Valid values: 0 (off) - 9
0x0001 – Brightness of the screen while in passive (screensaver) mode – Valid values: 0 (off) - 9
0x0002 – Timeout until screen goes into passive mode in sec. – Valid values: 0 (off) - 7200
0x0003 – Volume of the integrated beeper - Valid values 0 (off) – 4

It is also possible to read the set values via a Modbus TCP Client – Function Code 04 (read input registers) with start address 0x0000 and length 4, the settings are in the same order as when you write the values.

4.1.10. Menu item "Web Server"

On the settings sub menu the controller's internal web interface can be enabled or disabled.

> ⚠ **CAUTION**
>
> **By disabling the web interface, you will lose access to the configuration menu!**
> To get access to the web interface again after disabling, the maintenance mode can be used.

Additional settings are to change the default ports for http and https and depending on the device.
the automatic redirection from http to https.

> ℹ **NOTE**
>
> Non-encrypted access to the web interface is no longer possible with firmware 1.27.x (IMX), requests
> to the web interface via http:// are automatically redirected to https://
> All Pi devices can only be accessed over https:// and there is no redirect option.
> All controllers always have a default (self-signed) certificate that must be trusted manually the first time
> the web interface is loaded up by a browser.

It is also possible to upload a new logo to change web-logo shown on the login screen and on the upper left corner with a customer logo.

> **NOTE**
>
> The logo/picture format must be in *.gif, neither the web interface nor the web browser will do any scaling on the logo, so an appropriate resolution for the logo must be chosen.

On the Certificate (self-signed) sub menu the controller can create a self-signed certificate with different credentials. The self-created certificate can be downloaded and imported into the browser which is to display the web interface, or you can trust the certificate again the next time the web interface is called.

On the Certificate (ext. CA) the controller can create a certificate request with different credentials. You can then download this request and have it signed by an external certificate authority (CA). Once you have received the externally signed certificate, you can upload it to the controller using the upload button. After rebooting the controller and calling up the web interface for the first time, the browser should then recognize the externally signed certificate itself and trust it automatically.

Every certificate process for self or external signed must be done with every device.

4.1.11. Menu item "Users"

On this page, the passwords of the users present on the controller can be changed.
In addition, it is possible to create up to five own FTP-Users, which can be assigned to any username and home directory, set the user active or inactive as well as set the access rights based on the three standard FTP-Users.
Change the passwords of all users before using the controller in a productive environment, or make sure that no physical access to the controller or the network connected to the controller is possible.

User name	FTP / Web	Rights FTP	Rights Web
root	Yes / Yes	Read/Write	Read/Write
admin	No / Yes	None	Read/Write
ftpuser	Yes / No	Read/Write (only ftpupload)	None
ftpadm	Yes / No	Read/Write (only ftpupload)	None
ftpreader	Yes / No	Read	None
ftpuser1-5	Yes / No	Configurable (ftpreader/ftpuser/ftpadm)	None

4.1.12. Menu item "Operating mode" (ET only)

On this page, the operating mode of ET devices can be configured to either E-Terminal (VNC Client) or Web-Terminal (Web client).

4.1.13. Menu item "Web browser "(Pi and ET only)

The Web browser menu consists of two sub menus for "Settings" and "Certificates".

i NOTE

Early firmware version on Pi controllers, before the target visu was supported and ET devices have the options for "Input Config". and "Font Files" as a submenu in the Web browser menu

The "Settings" submenu allows to set one "Destination URL" which will be loaded during the bootup of the controller.

Setting	Description
Browser decorated mode	By default, the integrated web browser starts in kiosk mode without any browser GUI. This option loads the browser with its full GUI.
URL connectibity check	Activates a loading page that shows information while trying to load the "Destination URL".
Browser remote debugging	Expert setting, enables remote debugging with the browser development tools
Activate onscreen keyboard (ET)	Enables the use of an automatic pop-up keyboard when using the ET in web mode

The "Certificates" submenu shows an overview of all installed certificates which are used by the internal web browser, this certificate submenu is not related to the certificates for the web server.
There are options to delete selected certificates or the whole list as well as an upload option to load external certificates into the internal web browser.

4.1.14. Menu item "Input Config"

This menu allows you to set the keymapping for connected USB keyboard to either US (QWERTY) or DE (QWERTZ).

4.1.15. Menu item "Config protection"

This menu allows you to set a password for the on-screen configuration button on the service menu of devices with display and VNC server.

4.1.16. Menu item "Reset Config"

On this page, the settings of the controller can be reset to the default state or factory settings. These include the network settings, the date and time settings, the display settings, the settings of the FTP server and the passwords of all users.

> ⚠ **CAUTION**
>
> **All user data, CODESYS V3 applications and settings are also deleted!**
>
> **Excluded from the deletion are only licenses installed on the controller.**
>
> After resetting the settings, the controller must be restarted.

4.2. Configuration interface: System

4.2.1. Menu item "System Info"

On this page you will find all the important information about the controller.

Option	Example	Explanation
Part-Name	B-Nimis MC-PI Prime S02	Product name of the controller.
Device ID	S-01030302-0200-00025	Combined product and serial number
Firmware-Version	1.3.2	Version of the firmware currently installed on the controller.
Codesys RTS Version	3.5.18.40	Version of the CODESYS V3 Runtime currently running on the controller. The first two digits stand for the CODESYS main version, the third for the service pack, the fourth for the patch level, and the fifth digit for the hotfix level (if any).
Installed options / Licenses	(S104) ETHERCAT-MASTER (S114) WEBVISU	All licenses installed on the controller. For some libraries, e.g. Modbus TCP requires additional licenses that may need to be installed.
System operation Time	1612 hours 0 min	Total running time of the controller since the first commissioning.
System Uptime	0 day 0 hour 19 min	Run time of the controller since the last start of the operating system.
CPU Temperature	Cur: 49.7°C, Timestamp: 05/25/23 : 10:09:37	Shows the current, max. and min. temperatures measured directly on the CPU.
Sensor Tempearture	Cur: 50.1°C, Timestamp: 05/25/23 : 11:00:51	Shows the current, max. and min. temperatures measured inside the PLC case.
Memory	total: 1860 MB available: 1536 MB	Total and available amount of RAM on the PLC
Flashmemory State	MLC: 20 - 30% device lifetime used preEOL: normal (1, consumed less than 80% of reserved blocks)	Total number of full write cycles on available flash memory as well as memory health information

4.2.2. Menu item "System Update"

On this page, you can upload various files to the controller to install firmware updates or to install additional licenses. To perform such an update, all CODESYS V3 applications must first be stopped on the controller.

First, select the desired file (for example, firmware_mx6-plc_x.x.x.tgz) with the "Browse..." button and upload it by pressing the "Send data" button. Uploading the file may take several minutes, depending on the size and quality of the connection. After the upload, the web interface displays a description and version of the uploaded file, and you can check it. With the "Start" button, you can now initiate the update process; the update can take up to two minutes depending on the size of the .tgz file.

⚠ CAUTION

A started update can not be interrupted anymore. The power supply must not be removed from the device during an update process. Premature termination of an Update makes the device a repair case!

ⓘ NOTE

While the firmware update is in process it might happen that the connection to the web interface is lost. In this case try reloading the web browser, if a connection cannot be established again pay close attention to the blinking sequence of the Run/Stop LEDs, until the indicate a reboot (blink 2x, 2s break).

⚠ CAUTION

After the update, the controller must be restarted. No applications or user data on the controller will be deleted during the update. The controller tries to load existing boot applications in the status "AS_RUN" after the required restart.

4.2.3. Menu item "System Reboot"

The controller can be restarted on this page. Some changes to the controller settings require a subsequent restart. All running applications on the controller are interrupted in this.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

35

4.3. Configuration interface: PLC-Manager

4.3.1. Menu item "PLC Control"

On this page, you can influence CODESYS V3 applications located on the controller.
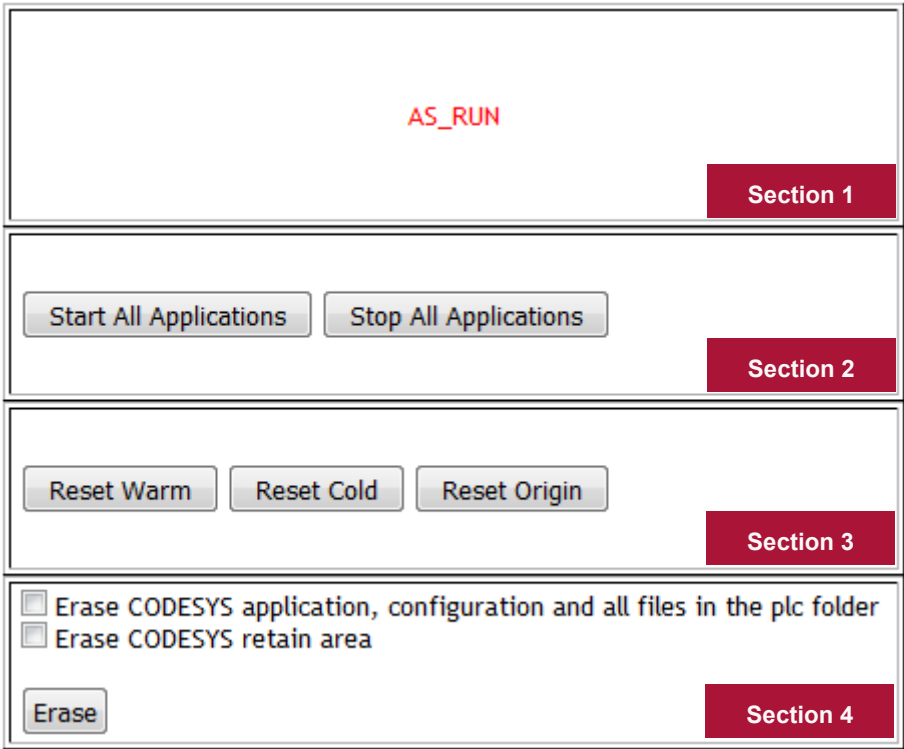
Picture 4: PLC-Control

In section 1, we summarize the status of all applications on the controller.

Status	Explanation
AS_PARTIALLY_STOPPED	There is at least one application in the status "AS_STOP".
AS_RUN	All applications on the controller are in status "AS_RUN".
AS_STOP	All applications on the controller are in status "AS_STOP".
AS_NO_APPLICATION	There are no applications on the controller.

In section 2, all applications can be started or stopped at once.

Option	Explanation
Start All Applications	An attempt is made to shift all Applications into the status "AS_RUN".
Stop All Applications	An attempt is made to shift all Applications in the "AS_STOP" status.

In section 3, the control can be reset.

Option	Explanation
Reset Warm	All Applications on the controller are shifted to the status "AS_STOP" and all normal variables are reset.
Reset Cold	Like "Reset Warm", only all RETAIN variables are additionally reset.
Reset Origin	All applications on the controller are deleted and all variables on the controller are reset. Also the RETAIN and RETAIN PERSISTENT variables of all applications are reset. After this action, no login is possible without re-downloading the application to the controller.

In section 4, the control can be reset.

Option	Explanation
Erase CODESYS application, configuration, and all files in the plc folder	All applications on the controller are deleted and all variables on the controller are reset. Also, the RETAIN and RETAIN PERSISTENT variables of all applications are reset. Additionally, all files and folders in the /home/plc/applications directory are deleted. A restart of the controller is necessary after this action.
Erase CODESYS retain area	The RETAIN and RETAIN PERSISTENT variables of all applications are reset. A restart of the controller is necessary after this action.

4.3.2. Menu item "Config"

Some special settings of the controller can be made on this page.

Option	Explanation
PLC application on SD-Card (IMX)	This option causes the SD memory card to be connected in such a way that the controller can execute applications directly from the memory card. Warning. If you want to use this feature, you must use a special Berghof provided SD card. Commercially available SD cards are only recognized as mass storage but not as additional system memory. If the option is activated, the system will not start without Berghof SD card anymore and it will not be possible to load an application on it. In this case, however, the web interface remains accessible so that the option can be switched off again.
CODESYS Gateway enabled	Select if the CODESYS Software Gateway functionality is activated on the controller.
CODESYS Edge Gateway in configuration mode	Restart the CODESYS Software Edge Gateway in configuration mode to setup the connection to the CODESYS Automation Server via CODESYS V3.
CODESYS OPC UA-Server enabled	Select if the CODESYS OPC UA Server functionality should be activated inside the CODESYS Runtime.
CODESYS PLC Enforced User Management	Enables the forcing of the CODESYS User management. This means when connecting to the controller for the first time via CODESYS V3, the user must set a username and a password. This user management is independent from the used CODESYS project.
CODESYS Force IEC File Path	Option to restrict all file access from the application (IEC code) to the "PlcLogic" working directory

4.3.3. Menu item "Application Info"

On this page you will find information about applications that are located on the controller.

Characteristic	Explanation
Applicationname	Name of the application must be unique. Can be changed in the CODESYS V3 development environment by changing the name of the "Application" object.
Status	The status of the application. AS_RUN: Application is running. AS_STOP: Application is not running. Manual stop or error during execution.
Projectname	The project information specified in the CODESYS V3 development platform is displayed. To change this information, open the menu "Project" in CODESYS V3 in the menu bar and select the menu item "Project information".
Projectauthor	
Projectversion	
Projectprofile	

38

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
01-0099-82-00-00-000 | Berghof_Control_Systemmanual_1.5_EN.docx

Projectdescription	
Exception-ID	Indicates if the application is in an error state. Exception ID 0x00000000 means that there is no error.
Exception	Name of the error state.

4.3.4. Menu item "Application Files"

This page shows an overview of all files on the controller. All files can be downloaded individually. In addition, the following actions are available:

Action	Effect
Download folder from PLC	All files in the /home/plc directory are packed with additional recovery information into a compressed archive, which can then be downloaded. If the archives are to be used for an upload to another controller, they must not be changed or unpacked.
Upload folder to PLC	A previously downloaded archive can be uploaded to the controller. It can be used to quickly make backups and restore, or to quickly distribute Applications to many controllers that cannot be reached over a network. Attention! Existing files and Applications are overwritten without further request. A restart must be carried out after uploading an image.
Clean folder	All applications are removed from the controller. Configuration files of CODESYS V3 are retained. If you want to remove all files including CODESYS V3 configuration files from the controller, then use the in chapter 4.3.1 presented function "Erase CODESYS application, configuration and all files in the plc folder" in the section PLC Control. After a Clean, the controller must be restarted.

4.3.5. Menu item "Font Files"

This page shows an overview of the installed fonts. The fonts are divided into "system fonts" and "PLC fonts". Chapter 8.3 explains, how you can install new fonts on the controller.
Please note that **either** the system fonts **or** the custom fonts can be used. If you would like to use both your own fonts and the fonts that already exist on the controller, you must first download the system fonts and then upload them again with your own fonts to the "PLC Fonts" on the controller.

On Pi devices this optins are a submenu in the Web browser menu

Overview point	Description
System Fonts	The default fonts preinstalled on the controller are listed. It is not possible to change or delete the default fonts.
PLC Fonts	The fonts loaded by the user to the controller are listed.

4.4. Configuration interface: Diagnostics

4.4.1. Menu item "PLC Log"

You can see the log of the CODESYS V3 Runtime on this page. The data recorded in the log includes:
→   Information about the used CODESYS V3 version used and activated licenses.
→   The used system libraries including version.
→   Network information.
→   CODESYS V3 Events such as logging in or logging out of users or downloading applications.
→   Error cases or exceptions that occur in the CODESYS V3 Runtime.

4.4.2. Menu item "System Log"

This page is divided into two sections:
→   In the "System Log" area, the system log is displayed, which can be found in the file system under /var/log/messages. It contains general information about the operating system and running services and programs. For example, accesses to the web interface are recorded by the lighttpd web server.
→   The "System Diag" area records interactions between the system and the CODESYS V3 Runtime. Entries contain, for example, information about changes to the retain memory, states of the CODESYS V3 Runtime, boot and power fail times.

4.4.3. Menu item "Ethernet"

This page provides information about the network interfaces of the controller. In contrast to the menu item "Network" (see chapter 4.1.1), no settings can be made. However, there is detailed information such as MAC address, set IP, and received and sent packets and data quantities

4.4.4. Menu item "CAN"

Information on the CAN interfaces can be viewed on this page. Information about the BUS status can be read out via an internal Error Frame counter:

can state : ERROR_ACTIVE   →   CAN active   (<96 Error Frames)
can state : ERROR_WARNING →   CAN active   (<128 Error Frames)
can state : ERROR_PASSIVE   →   CAN inactive   (<256 Error Frames)
can state : ERROR_BUS_OFF →   CAN off   (>=256 Error Frames)
can state : ERROR_SLEEPING→   CAN in Standby
can state : STOPPED   →   CAN stopped

In addition, the set baud rate, as well as received and transmitted packets, data volumes and the total number of Error Frames received can be displayed.

4.4.5. Menu item "Storage"

Information about the memory status of the controller can be viewed on this page. The most important information for the user is the memory status of the flash memory (marked in green here) and the status of the external SD card (marked in blue here) if one is integrated.
If one memory stick or several USB memory sticks via Hub should be connected, these are recognizable in the "Mounted on" column with the entry "/media/usbx" (marked here in orange). The x stands for mount order (with a stick normally number 1).

```
Filesystem            Size   Used Available Use% Mounted on
ubi0_0               47.5M  18.9M    28.6M  40% /
devtmpfs            114.3M      0   114.3M   0% /dev
/dev/ubi0_1          47.5M  41.3M     6.2M  87% /usr
none                196.0M  88.0K   195.9M   0% /tmp
none                122.5M      0   122.5M   0% /media
none                122.5M 164.0K   122.3M   0% /run
none                122.5M  68.0K   122.4M   0% /var/log
none                122.5M 164.0K   122.3M   0% /var/run
none                122.5M      0   122.5M   0% /var/lock
none                122.5M      0   122.5M   0% /var/tmp
/dev/ubi1_0         107.4M   5.8M   101.6M   5% /flash
/dev/ubi1_0         107.4M   5.8M   101.6M   5% /home/plc
/dev/ubi1_0         107.4M   5.8M   101.6M   5% /usr/local
/dev/ubi1_0         107.4M   5.8M   101.6M   5% /var/cache
/dev/ubi1_0         107.4M   5.8M   101.6M   5% /var/spool
/dev/ubi0_1          47.5M  41.3M     6.2M  87% /etc
/dev/mmcblk0p1      977.1M   4.0K   977.1M   0% /media/sd
none                196.0M  88.0K   195.9M   0% /var/www/tmp
/dev/sda1             7.7G   1.4G     6.3G  18% /media/usb1
```

Picture 5: File system overview

4.4.6. Menu item "System Dump"

On this page, an image file of the entire diagnostics area of the controller can be created. This function is used for analysis of an application or the controller in case of an error. It is recommended to create the image file immediately after the error occurs, without prior restart. Creating the image file can take several minutes. Once the file has been created, it is offered for download by the browser. Save the file and send it to Berghof Support for analysis.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

41

5.   CODESYS V3 development environment

5.1.   General information

The following part of the system manual deals with the installation and configuration of the CODESYS V3 development environment. It is assumed that the controller is already configured and connected to the developer's PC. If you have not yet accessed or configured the controller, before installing CODESYS V3, please follow the steps described in chapter 3 to commission the controller.

5.2.   Important notes on different CODESYS V3 versions and supply sources of installation files

As described in chapter 2.5, the software components consisting of the CODESYS V3 development environment, the firmware of the controller and the additional system libraries and device description of the controller (called "target") form a unit and are coordinated with each other. This means that the Berghof controllers **cannot** be used with any arbitrary CODESYS V3 version. For this reason, it is important that you install the CODESYS V3 version intended for your controller **rather than** just using the latest available version of the CODESYS V3 development environment.

If you do not have installation files, you can download all the necessary components from the Berghof Closed Download Area. Additionally, the matching CODESYS Version can be downloaded from the CODESYS Store with the following address: https://store.codesys.com/en/. If you have any questions, please contact the technical support (support-controls@berghof.com).

The required version of CODESYS V3 and the corresponding target depend on the firmware version of the controller. To find out the firmware version, log in to the web interface of the controller and open the "System Info" page.

The following table shows the required CODESYS V3 and target version for your firmware.

Necessary CODESYS Version	Platform	Corresponding Firmware / Target
CODESYS V3.5 SP 4 Patch 3	IMX	1.0.0.
CODESYS V3.5 SP 4 Patch 3	IMX	1.1.x.
CODESYS V3.5 SP 5 Patch 3	IMX	1.2.x.
CODESYS V3.5 SP 5 Patch 4	IMX	1.3.x.
CODESYS V3.5 SP 7 Patch 4	IMX	1.7.1
CODESYS V3.5 SP 8 Patch 4	IMX	1.10.x
CODESYS V3.5 SP 9 Patch 4	IMX	1.13.x

CODESYS V3.5 SP 10 Patch 5/7	IMX	1.17.x.x
CODESYS V3.5 SP 11 Patch 4	IMX	1.18.x
CODESYS V3.5 SP 13 Patch 4	IMX	1.21.x
CODESYS V3.5 SP 14 Patch 4	IMX	1.22.x
CODESYS V3.5 SP 15 Patch 4	IMX	1.23.x
CODESYS V3.5 SP 16 Patch 4/7	IMX	1.24.x / 1.25.x
CODESYS V3.5 SP 16 Patch 5	PI	1.1.x
CODESYS V3.5 SP 17 Patch 3	IMX	1.26.x
CODESYS V3.5 SP 17 Patch 3	PI	1.2.x
CODESYS V3.5 SP 18 Patch 4	IMX	1.27.x
CODESYS V3.5 SP 18 Patch 4	PI	1.3.x
CODESYS V3.5 SP 19 Patch 7	IMX	1.28.x
CODESYS V3.5 SP 19 Patch 7	PI	1.4.x
CODESYS V3.5 SP 20 Patch 4	IMX	1.29.x
CODESYS V3.5 SP 20 Patch 4	PI	1.5.x

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

43

5.3. Installation

The installation of CODESYS V3 may require an internet connection under certain circumstances if some required components such as the Microsoft .NET Framework have to be additionally installed.
Run the file **CODESYS.3.5.xx.xx** as administrator to start the CODESYS V3 installation. If you do not know what features you need, you should do a full installation with all the features, otherwise you can de-select some features that are not absolutely necessary.
Only the features **CODESYS V3** and **CODESYS V3 Gateway** are required in any case.
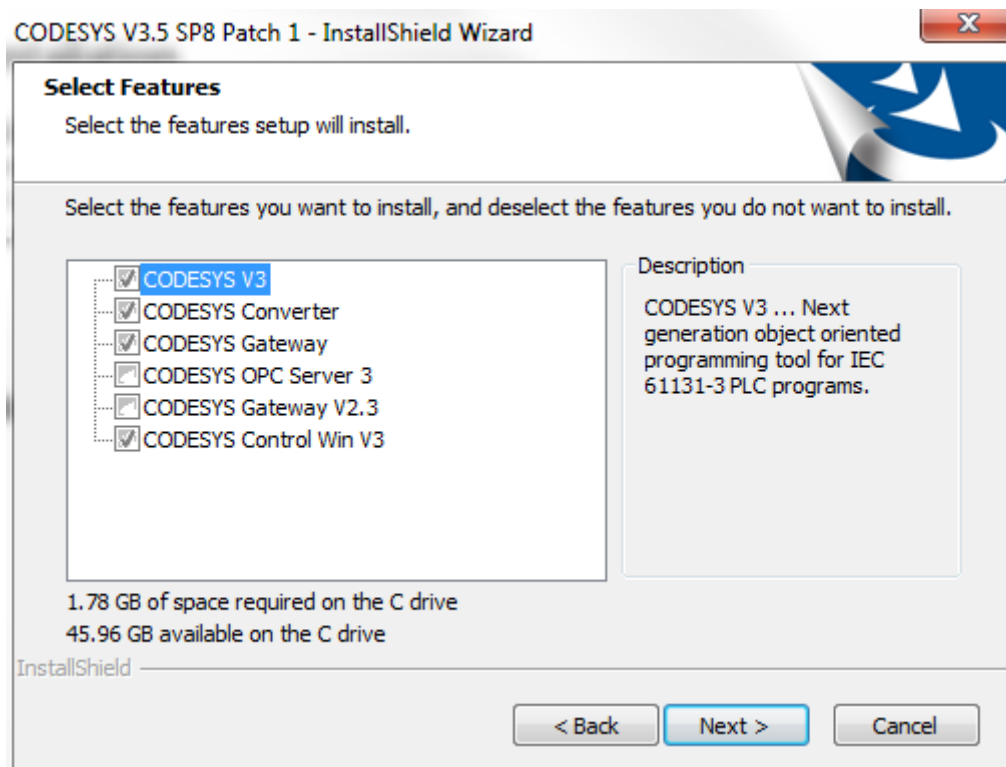
Picture 6: Codesys Installation

Decide for yourself which features are needed:
→   If you do not want to convert CODESYS V2.3 projects to V3, you can refrain from installing the
        CODESYS Converter. (Obsolete with newer CODESYS Versions)
→   If you do not need access to CODESYS V2.3 controllers, you can refrain from installing the
        CODESYS Gateway V2.3. (Obsolete with newer CODESYS Versions)
→   If you do not use OPC functionality, you can refrain from installing the **CODESYS OPC Server 3**.
        Please note: You can also access the controller via OPC UA without an OPC server, but you will
        need an OPC UA license for your controller.
→   If you do not need a software PLC for Windows to run and test projects directly under Windows,
        then you can do without installing the **CODESYS Control Win V3**.

Now you must install the appropriate target for your controller. The target is available in the form of a CODESYS V3 package (package file). Depending on the CODESYS Version these can be installed via the CODESYS V3 Package Manager or the CODESYS Installer.
It is recommended to install CODESYS V3 Packages only with administrator rights.

Using the Package Manager (3.5.16.xx or older):

Open your CODESYS IDE and select Tools -> Package Manager in the Menu.
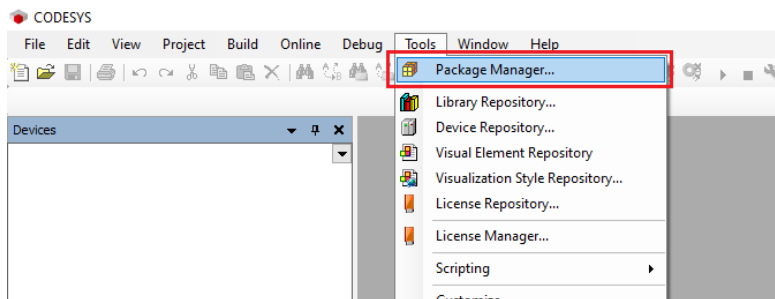
Picture 7: Package Manager

Open the Package Manager and click on Install to open file browser and select the Target Package file.
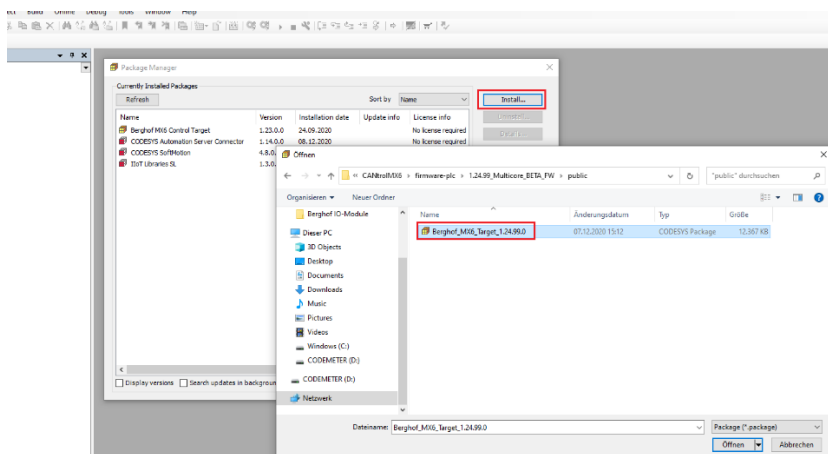
Picture 8: Package Installation

Continue through the setup and select full installation.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

45

Using the CODESYS Installer (3.5.17.xx or newer):

Search for the CODESYS Installer in your Windows Search Bar.
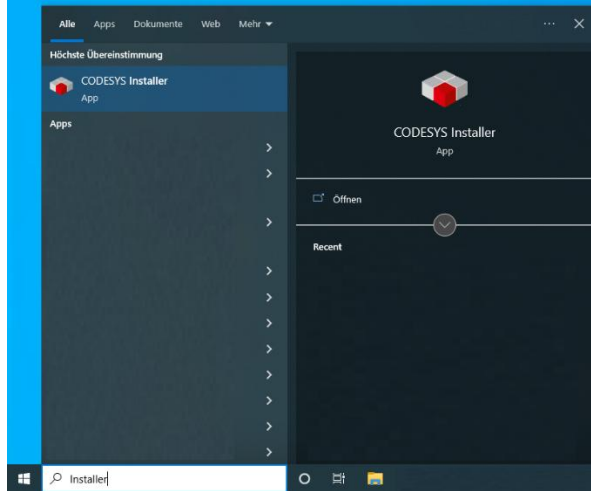
Picture 9: Open CODESYS Installer

Run CodeSys Installer as administrator and select the CODESYS installation you want to install the target package.
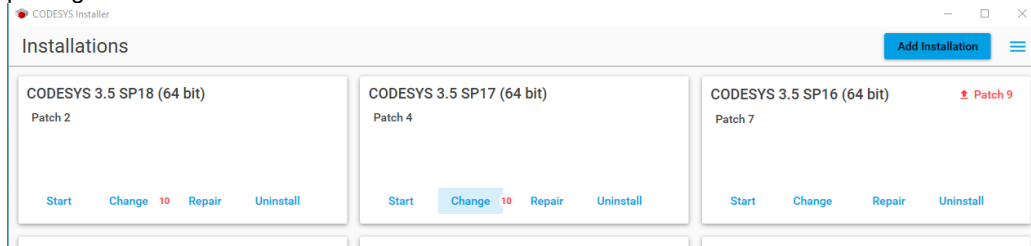
Picture 10: CODESYS Installer

Click Install File to open file browser and select the Target Package file.
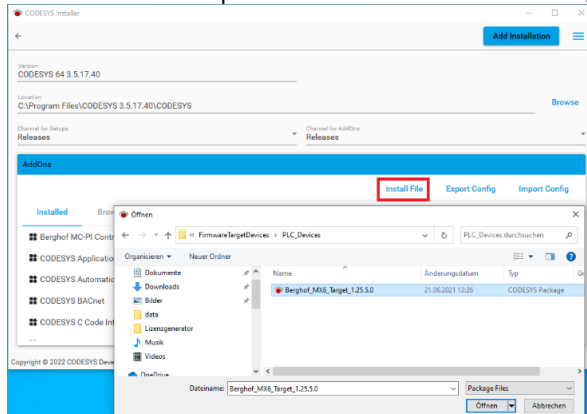
Picture 11: CODESYS Installer package installation

If asked trust the package and continue through the setup and select full installation.

Restart your computer after installing CODESYS V3 and the target.

5.4. Upgrade / downgrade of a CODESYS V3 version

There are several scenarios that require an update or upgrade of CODESYS V3. An update in this case is a minor change (e.g., bug fix) of the software. In the case of CODESYS V3 thus increasing the patch level. By contrast, an upgrade significantly adds new features to the software. In this case, a new service pack of CODESYS V3 will be released.
The following scenarios require an update or upgrade of CODESYS V3:

1. Berghof will release new firmware for your controller that fixes errors pertaining to one of your projects. In this case, you need to update the firmware on your controller, then install the appropriate CODESYS V3 version and also a new target matching the firmware. Furthermore, the entire system must be fully tested again to avoid unwanted side effects or new problems.
2. A new version of CODESYS V3 is released by 3S Smart Software Solutions GmbH, which fixes some errors related to one of your projects (new patch level). In this case, it is sufficient to install the new CODESYS V3 version and recompile and test the project.

To perform an update / upgrade, the setup file of the appropriate version is executed; close all CODESYS V3 instances first. The installation wizard automatically detects the installed features; confirm the setup with "Next" until the installation of the update begins. It is recommended to restart the PC after the update.
Basically, several CODESYS V3 versions can be installed and used at the same time. It is only important that the version of CODESYS V3 suitable for the firmware and a corresponding target are always installed. This is the only way to ensure that all required system libraries are available. Later patch levels of CODESYS V3 can then be additionally installed.

> **i   NOTE**
>
> Only install a newer version of CODESYS V3 if you explicitly need features of this new version or if this version fixes errors pertaining to one of your projects.

If you have a newer version of CODESYS V3 installed for which, however, there is still no suitable firmware and no suitable target for your controller, a downgrade of CODESYS V3 is recommended to avoid incompatibilities. A downgrade of CODESYS V3 consists of:

1. The complete uninstallation of all CODESYS V3 versions on the target system.
2. Manually delete the folder C:\ProgramData\CODESYS.
3. The subsequent reinstallation of CODESYS V3 in the released version.

ℹ NOTE

Please note:

The folder C:\ProgramData\ is normally hidden. However, you can view the contents of the folder by typing the path directly into the address bar of Windows Explorer and pressing "Enter."
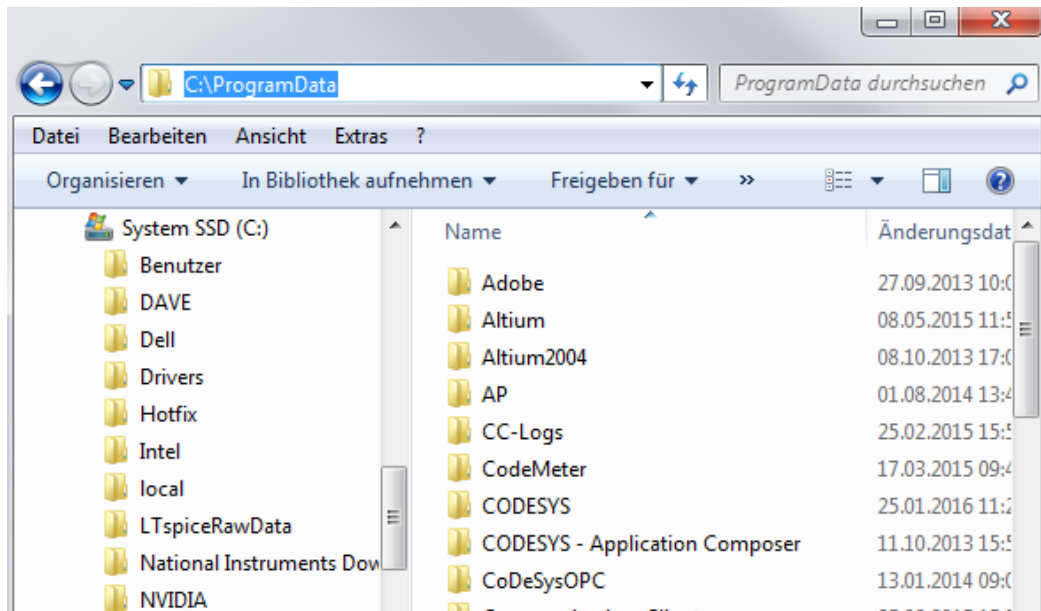
Picture 12: Folder

6.   CODESYS V3 Quick start

The following chapter explains how to create, compile, and then run a first CODESYS V3 project. To understand the steps presented in this chapter, the following requirements must be met:

→   The network settings of the controller must be configured correctly so that the controller can be accessed via Ethernet. (See chapter 3.4)

→   A functional combination of CODESYS V3 and Berghof Target Package must be installed on the computer of the developer. Of course, this combination must be compatible with the firmware installed on the controller. (See chapter 5.2)

If these requirements are not met, first read the relevant chapters in this manual and then prepare the controller and your workstation accordingly.

In addition, some experience in managing development environments and programming control programs in Structured Text is helpful. If you are unaware of CODESYS V3 or are not yet sure about using Structured Text, you may want to consider attending a **training** at Berghof. The topics of training include:

→   The entry into PLC programming.

→   Designing visualizations for local and web-based ads.

→   Regulating and controlling of challenging drive tasks with CODESYS V3 SoftMotion CNC.

On request, Berghof can also offer in-house training programs tailored to your individual needs.

For more information, see https://www.berghof-automation.com/en/automation-technology/training-courses

6.1. CODESYS V3 Editor overview
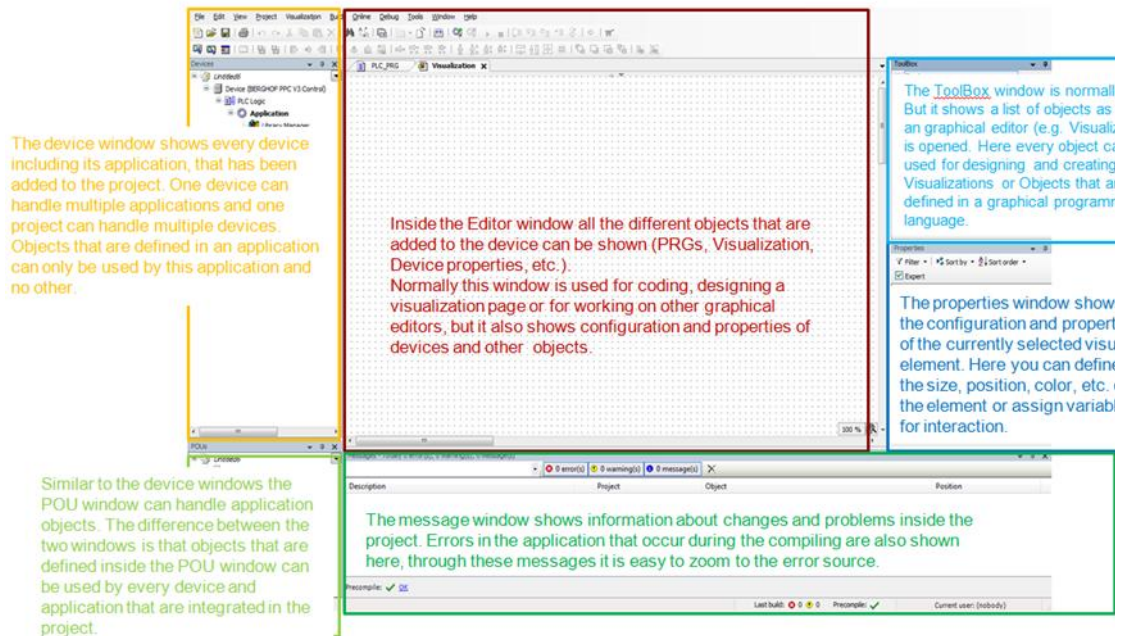
Picture 13: Codesys Editior

6.2. CODESYS V3 expand overview

6.2.1. Device Repository

Additional devices or libraries whether by Berghof or another manufacturer are always installed via CODESYS V3 in so-called "repositories". These "repositories" are a global filing location for CODESYS V3 on the hard disk. All standard devices and libraries as well as devices and libraries from other manufacturers which are subsequently installed are in this storage location and are accessible to all installed CODESYS V3 versions installed on the PC. The "repositories" are PC bound, that is, for each PC on which CODESYS V3 is installed, all retrofitted devices and libraries must be reinstalled. Every project and every user use it, there is no longer a direct link to a file, as known from older CODESYS versions.

The two most important "repositories" in connection with our controllers are the following:
→   Devices Repository.
→   Libraries Repository.

The device repository is the storage location for all control descriptions, I/O peripherals, drives and other devices installed in CODESYS V3. The repository is independent of manufacturer and bus, so that all description formats supported by CODESYS V3 can be installed in the repository. Installed devices can then be selected in the control configuration of CODESYS V3.
The repository can be opened in CODESYS V3 via the Context menu: Tools-> Devices Repository.

50

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
01-0099-82-00-00-000 | Berghof_Control_Systemmanual_1.5_EN.docx
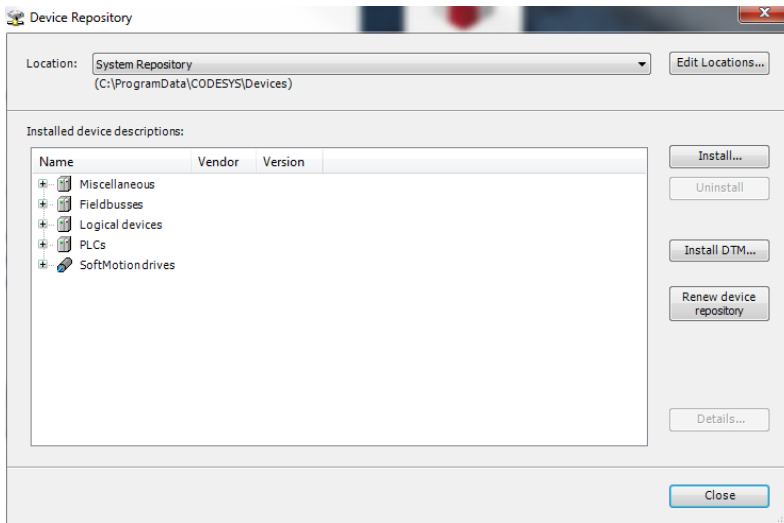
Picture 14: Context-menu

The installed devices are sorted into five subgroups for a better overview.

→   Miscellaneous: for peripherals not clearly assigned to a bus. Descriptions for the I/O's integrated into the Berghof PLC are for example listed here.

→   Field buses: Each field bus periphery, whether master or slaves, are sorted here. In the subgroups, the devices are then always arranged in the corresponding field buses.

→   Logical devices: Logical I/Os for software-only linking of variables between multi-control projects, e.g. for data exchange between a normal PLC and a safety PLC inserted in the project.

→   PLCs: All controllers installed in CODESYS V3 are displayed here

→   SoftMotion devices: Drives specially marked as Softmotion devices are listed here. These are also categorized as with the fieldbuses in subgroups per device type and field buses.
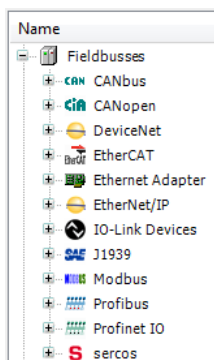
Picture 15: Fieldbusses

To install a device later, open the device repository in CODESYS V3 and click on the "Install" button. In the newly opened "File Open" dialog, navigate to the device description file and open it.
CODESYS V3 then installs the device in its repository; these devices can then be used in CODESYS V3.

If the file is not displayed in the destination folder, the file filter (in the dialog box at the bottom right, next to the file path) is probably set to a specific format; this filter should be set to "All supported files".

6.2.2. Libraries Repository

The library repository is the storage location for all libraries installed in CODESYS V3. The repository is manufacturer-independent; so, all CODESYS V3 libraries can be installed in the repository. Installed libraries can then be selected in the library management of CODESYS V3.
The repository can be opened in CODESYS V3 via the:
Context menu->Tools-> Libraries Repository.

A categorization is optionally switched on and off; in addition, either all installed libraries can be displayed or filtered by manufacturer.
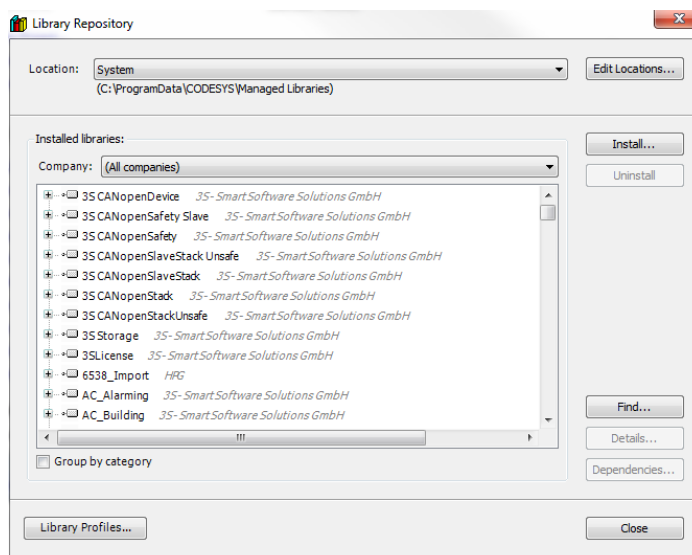
Picture 16: Libraries Repository

To install a library later, open the library repository in CODESYS V3 and press the "Install" button.
In the newly opened "File Open" dialog, navigate to the library file and open it.
CODESYS V3 then installs the library in its repository, then these libraries can be used in CODESYS V3.
If the file is not displayed in the destination folder, the file filter (in the dialog box at the bottom right, next to the file path) is probably set to a specific format.
The formats for libraries are as follows:

→ Libraries: Normal CODESYS V3 Library library file (*.library)
→ Compiled libraries: CODESYS V3 libraries which have been stored precompiled (*.compiled-library)
→ CODESYS libraries: CODESYS V2.x Libraries which can be opened under CODESYS V3 to be converted (if possible) (*.lib)
→ Libraries Repository.

6.2.3. CODESYS V3 Package

A package is a special CODESYS V3 archive format in which device descriptions, libraries, sample projects, visualization style, etc. are installed in one process. Packages are usually provided by control and device manufacturers, so that customers can conveniently install all the files they need. The Berghof Target is such a package and must be installed so that CODESYS V3 Berghof devices are correctly recognized.
The Package Manager is like a device or library repository. This shows which packages are installed on a PC in CODESYS V3. Unlike with devices or libraries, these packages are not used directly in a project, but rather their contents, which can be found in the other repositories.
The manager can be opened in CODESYS V3 via:
Context menu->Tools-> Package Manager
To install a package or the Berghof Target Package, first open a new CODESYS V3 instance as administrator. If the Package Manager is open in CODESYS V3, press the "Install" button. In the newly opened "File Open" dialog, navigate to the package file and select it. CODESYS V3 opens a small installation menu; if it is not known which contents are needed, the "complete installation" is normally selected here. After completing the package installation, an overview is displayed as to which contents of the package have been installed. These contents will then be available in the projects as of this time.

Since CODESYS 3.5.17.xx the package manager has been replaced with the CODESYS Installer.
The CODESYS Installer is a separate Tool which manages all local installations of different CODESYS versions and provides support for their deployment and update.
It can also be used to install 3rd party package files as well as download and install all packages that are provided in the CODESYS store.

6.3. CODESYS V3 Help

The first point of contact for questions about CODESYS V3 is on the PC on which CODESYS V3 is installed and each installation of CODESYS V3 contains the so-called "CODESYS Online Help"; unlike the name suggests, this help is also available offline as all needed files are on the hard disk.

The online help is a digital documentation platform for CODESYS V3, with category display and integrated index search. Each CODESYS V3 standard functionality is documented here, e.g. the user interface, settings, editors, data types, operands, programming features, visualization, visualization elements, standard libraries etc.

To open the online help, simply open an instance of CODESYS V3 and press the "F1" key on the keyboard while the window is open. After a short loading time, the online help is now available and can be searched.

> **i NOTE**
>
> **Tip:**
>
> If you highlight the element, you want information about, whether it's a window, a module in a graphical editor, ST code or a visu element, and then press F1, the online help will open immediately in the entries of the highlighted element, if any are existent. Otherwise, the index itself can be used to search for a matching entry.

Alternatively, you can find it in the CODESYS V3 installation directory in the subfolder Documentation (e.g. C:\Program Files\CODESYS 3.5.18.40\CODESYS\Documentation\en) several PDF files as an additional source of information.

CODESYS also offers a range of freely accessible online documentation, FAQ, webinars and example projects that explain the use of various CODESYS V3 features and demonstrate their use.

→ The webinars in German and English are available at: https://clips.codesys.com
→ The online documentation in German and Englisch are available at: https://www.helpme-codesys.com/
→ The FAQ is available at: https://content.helpme-codesys.com/en/CODESYS%20FAQ/_faq_start_page.html
→ Example projects for different CODESYS features are available at: https://forge.codesys.com/prj/codesys-example/home/Home/

54

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
01-0099-82-00-00-000 | Berghof_Control_Systemmanual_1.5_EN.docx

6.4. Sample project

In the next chapters, a sample project with CODESYS V3 will be created step by step, loaded on the controller and then executed. Several visualizations are created, and the digital inputs and outputs of the controller are used. The entire sample project can be downloaded as an archive through the "Mein Berghof" extranet section if you want to skip the step-by-step instructions. Users who already have experience with CODESYS V3 or have already attended training on this topic, can skip this chapter, as they already have all the necessary knowledge.

6.5. Create a new project

Start CODESYS V3 and create a new project by opening the "File" menu in the menu bar and then selecting the "New project" menu item. Give the project a name and select a folder in which you want to save the project. If the selected folder does not exist, you will be asked during initialization of the project if a new folder should be created. Select the symbol "Standard project" and push the "OK" button to create the project.
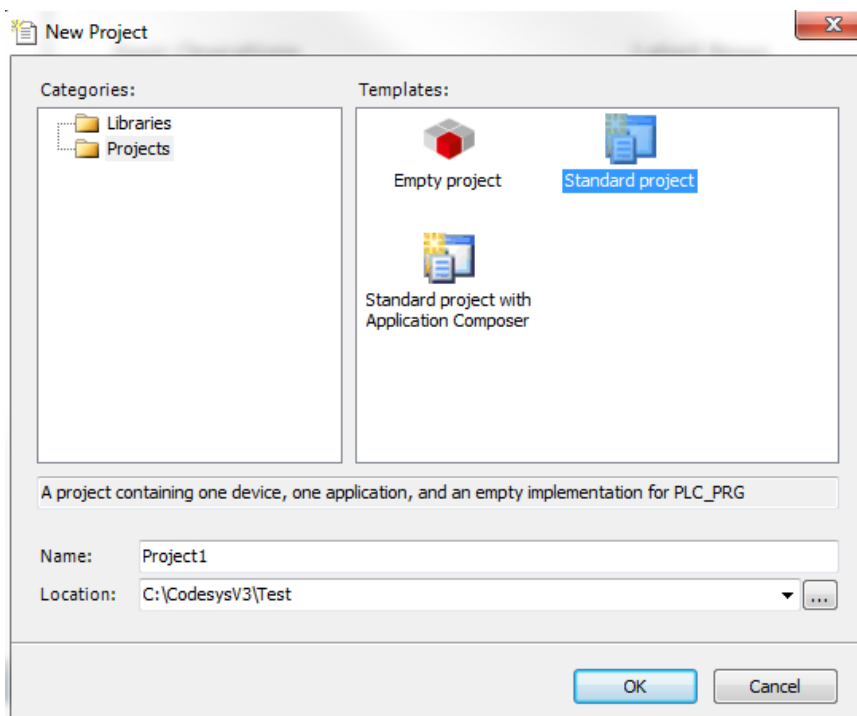
Picture 17: Create a new project

6.6.   Link the controller in the project

After you have created an "Standard project", you must first integrate a controller into the project. First select your newly created project and open the context menu with a right-click. Now select the menu item "Add device".
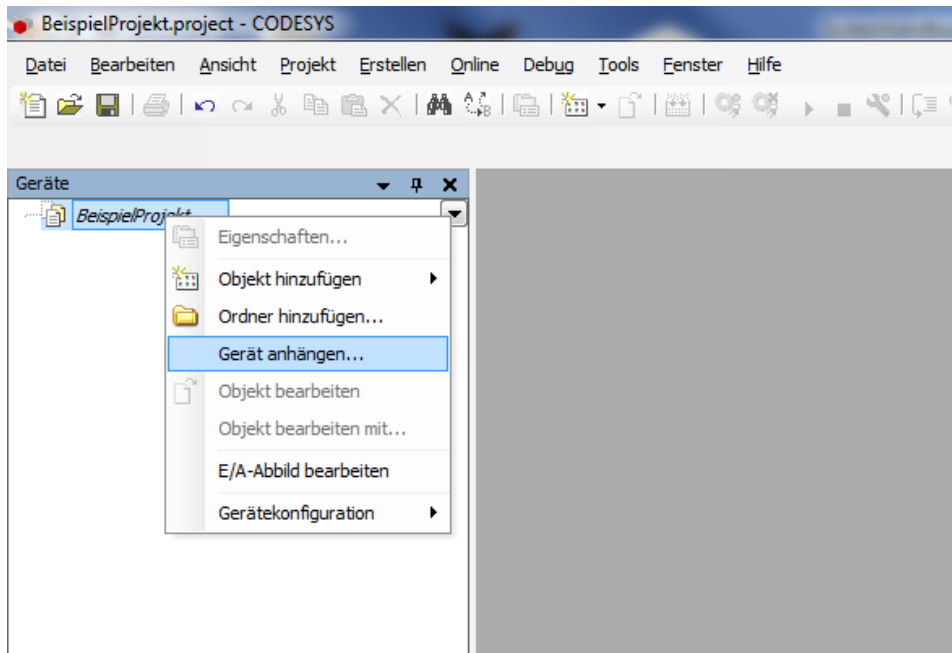
Picture 18: Add device 1.1

Make sure you link the right device in your project. For a controller of type EC2xxx or DC2xxx, select the entry "Berghof MX6 Control". Verify that the correct version of the device descriptions (Target) has been selected. In the example shown here, a device with target version 1.7.1.0 is selected. This target fits on a controller with the firmware version 1.7.1 and requires the CODESYS V3 Runtime in version 3.5.7.40. The version needed for this target is therefore CODESYS V3.5 SP 7 Patch 4. If you have multiple targets installed, then you can display all available targets by activating the "Show all versions" checkbox.
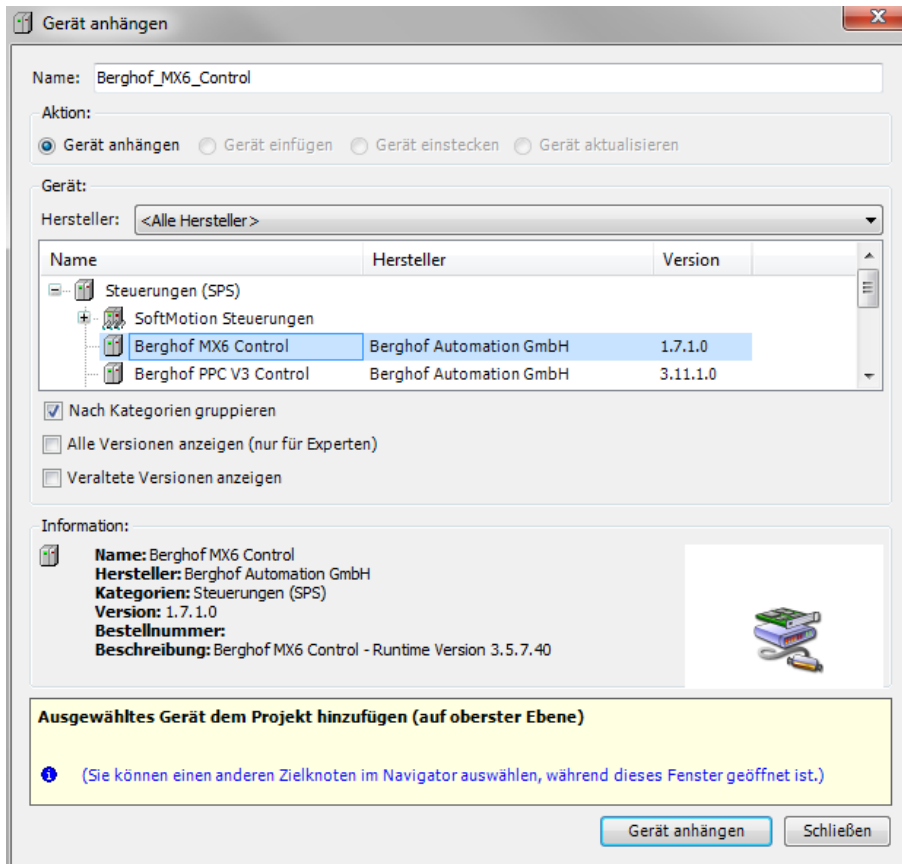
Picture 19: Add device 1.2

Now press the button "Add device" and then close the window.

If you have integrated the device, CODESYS V3 automatically appends additional objects to the project.

In the device window, an object of the type "PLC logic" is now displayed, below this an object of the type "Application" is appended and the library manager is appended under this.

6.7. Create a program and define a task

In order to get a complete and executable program, at least one object of the type "program" has to be inserted into the project and a suitable task has to be defined for the program to be executed. Select your application object, right-click to open the context menu, select "Add object" and then select "POU..." in the submenu.
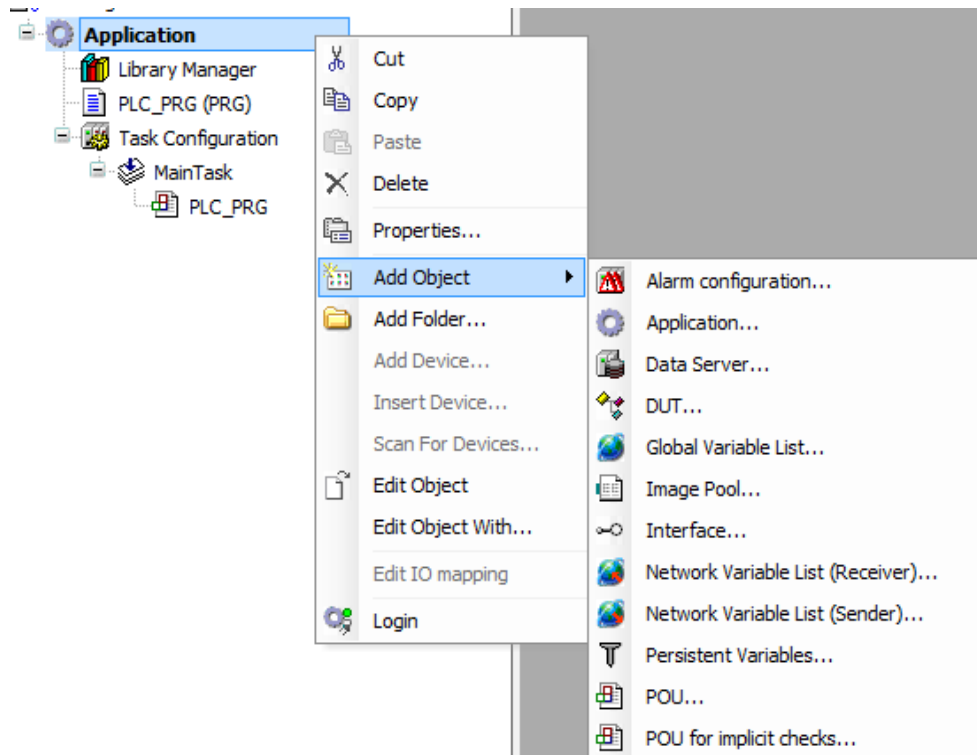
Picture 20: Create and define 1.1

Select the type of POU as "Program" and give the program a name. In the example, the name "Main" is used to indicate that this is the main program of the controller. The implementation language selected is ST (Structured Text). Then edit the module by double-clicking and define in the upper half of the editor window (also called interface editor) our first variable with the name "counter" and the data type INT. We initialize this variable with the value "0".
In the lower half, we now implement a simple program that runs when you call the "Main" object. The program adds the value "1" to the variable "counter" with each call.
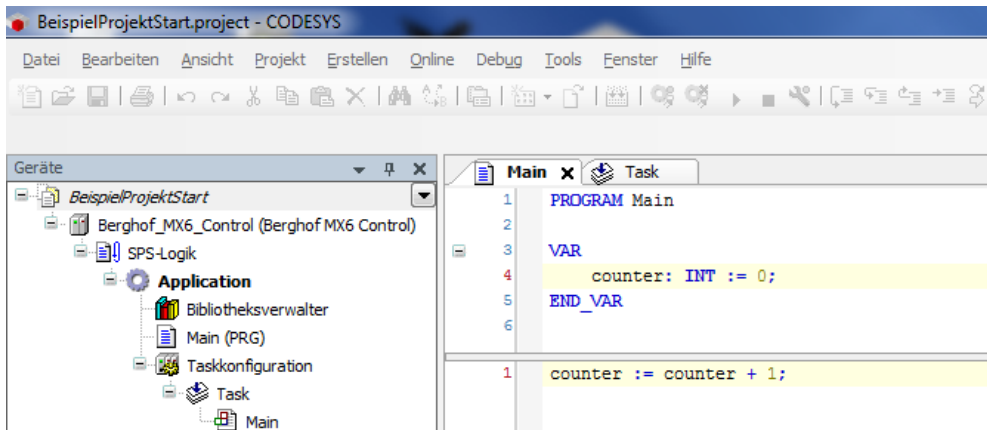
Picture 21: Create and define 1.2

For the program to be called, an object of the type "Task configuration" must be inserted. This object automatically creates a sub-object of type "Task". You can configure this by double-clicking on the "Task" object. Select "Add call" and then select your already created "Program" object.

By default, the task is set to a cycle time of 25ms. This means the controller will call and run your "program" object every 25ms. For several defined tasks, the priority indicates the probability with which the scheduler grants a computing time for a task.
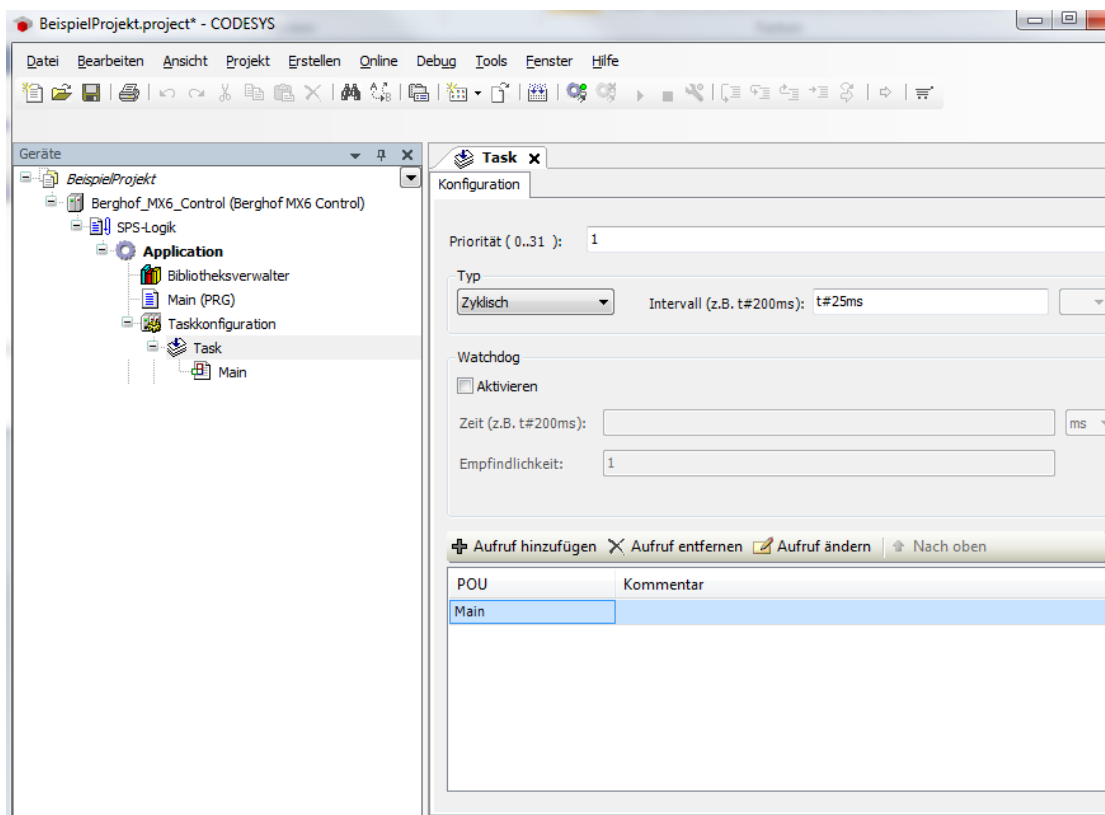
Picture 20: Create and define 1.3

> **ℹ NOTE**
>
> The fastest cycle time that can be used with a Berghof IMX controller in CODESYS V3 is 1 ms.
> PI based controllers are able to run cycle times down to 500us.

> **ℹ NOTE**
>
> Tasks with the priorities 0-15 are executed on the Berghof control system with real-time relevance and are recommended for important program tasks or bus cycles. The priorities 16-31 have no real-time relevance and are recommended for file operations on internal and external memory, logging or visualization.

6.8. Log in to the controller and download the project

If an application is to be loaded onto a controller, CODESYS V3 does not automatically know in which controller the project should be loaded. This requires the user to assign a controller to their CODESYS V3 project. In addition to the assignment of a controller, the application in your setup must be free from errors.

> **⚠ CAUTION**
>
> Machines must be brought to a safe state before a project download. A project download with active machines can lead to uncontrolled reactions.

Double-clicking on **"Device"** in the device window opens the communication settings, where no device has yet been assigned. By default, a Gateway added (symbol in the middle) must be released. The symbol on the right is the assigned controller; here still empty.

Now the network must be searched for the connected controllers. Simply click on the **"Search Network"** button.

After a short time, one or more devices should be displayed under the **"Gateway"**. The displayed device names correspond to the host name in the network settings on the web interface of the respective controller. You mark the desired device and double-click on it. The window then closes automatically.

If this is the case, the project has been assigned the desired controller and you can log in now. This is indicated by the green status point in the right symbol and by the information now displayed for the assigned controller. Before logging in to the controller and loading the application, the project should be checked for errors.

Alternatively, you can also scan directly to a controller, if the IP address is known, just type in the IP address of the controller in the text box below the controller symbol and start the scan with the Enter key.

In the menu bar under the item **"Create -> Build"** or by pressing the **"F11"** key, the program is built and checked for errors in the code, in the visualization and in the settings.
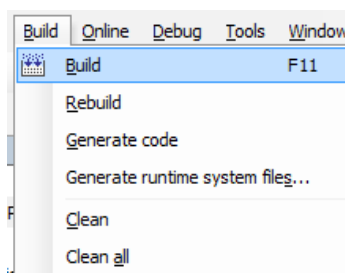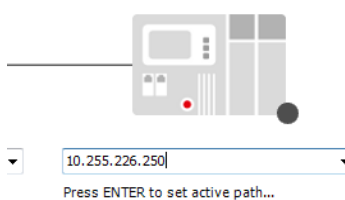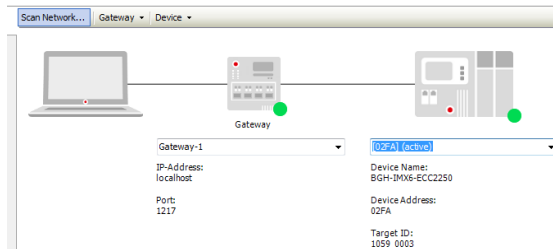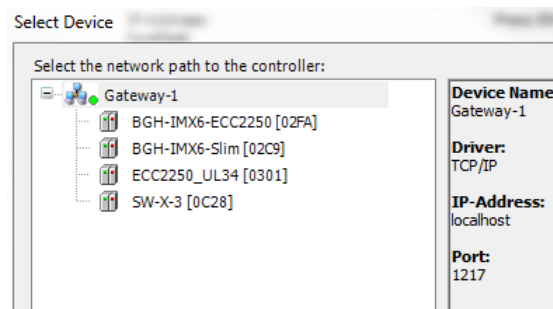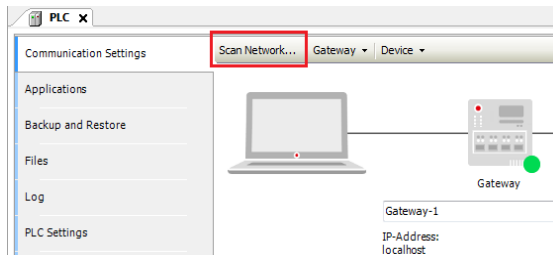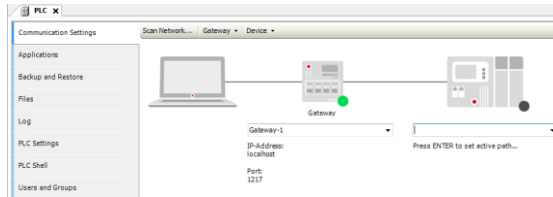
After a short wait, the result is displayed in the message window. If you are not mistaken in creating this example, **"0 errors"** and **"0 warnings"** should be displayed.
If an error does occur, it will be displayed in the messages. By double-clicking on the error message, CODESYS V3 will automatically jump to the location of the error. This makes it possible to fix errors effectively and easily.

If the project has been completely error-free and a controller has been assigned to the project, you can load the program onto the controller.

To log in, press in the menu bar **"Online -> Login"** or on the **"Login"** button, which is located below the items **"Window"** and **"Help"** in the menu bar.

If there is no application on the controller so far, a message appears that there is no application on the controller.

If an application is already loaded on the controller, a message appears stating that an unknown application is on the controller. This message may vary depending on whether the existing application is currently running or not.
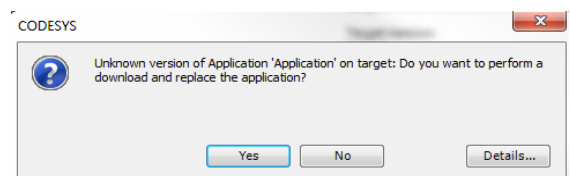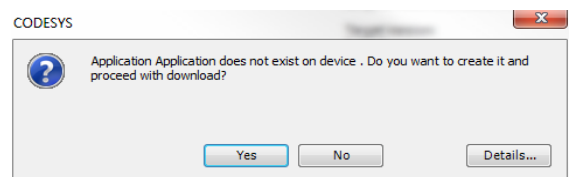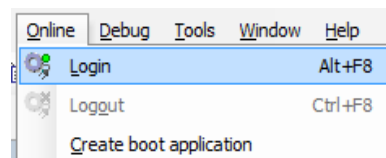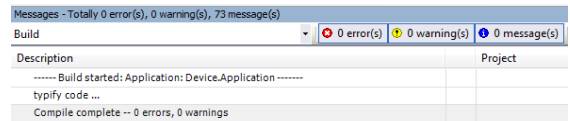
In all cases, confirm with **"yes"**. The only exception would be if the message comes that there are still errors in the program. In this case, cancel the log in and first find the errors in the program and rectify them. CODESYS V3 then starts to load the application onto the controller.

The loading process is finished when **"Device"** and **"Application"** have a green background and **"[Connected]"** or **"[Stop]"** behind them.
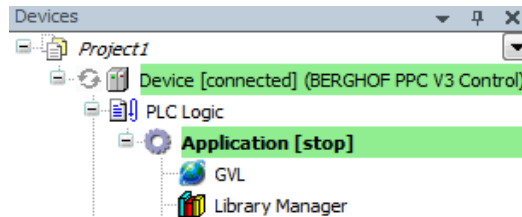
The application has been completely loaded onto the controller, but is still in idle state and is not yet running.

In addition, one can still create a boot application. This ensures that the application is retained even after a restart of the EC1000 and starts automatically. To do this, after logging in, click on **"Online -> Create Boot Application"** and wait until the loading process has been completed.
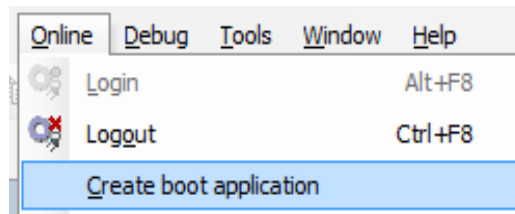
→ Optional

To start the program, click on **"Debug -> Start"** in the menu bar or press the **"F5"** key.
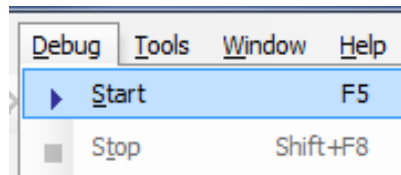
When the status of **"Application"** in the device window is changed from **"[stop]"** to **"[run]"**, the program is executed on the controller.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

63
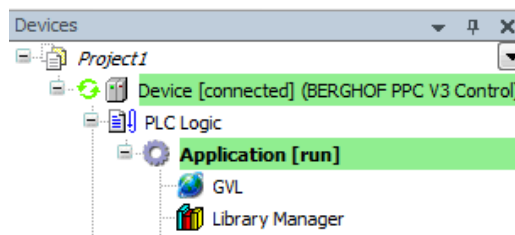
6.9. Common project modules and objects

With the steps mentioned in chapter 6.7, you have created an application executable on a controller.
A project can be extended with different objects. The insertion of an object always takes place according to the procedure described in chapter 6.7. Mark the object in which you want to insert an object and execute a right-click. In the menu that appears, navigate with the mouse to the item "Add object" and select the desired object in the submenu that now appears.
The most commonly used are POUs, DUTs and GVLs. These objects are briefly explained in this chapter.
An explanation of all other objects that can be used in CODESYS V3 and examples can be found in the "CODESYS V3 Help" described in chapter 6.3.

6.9.1. Program Organization Unit (POUs)

An object of type POU is a program organization unit in a CODESYS V3 project. In POUs, you write source code for your controller program. POUs can be programmed in all implementation languages supported by CODESYS V3.
There are the following types of POUs:
→ Program
→ Function
→ Functional module

As an alternative to the right-click method, you can add an object POU using the command Add ➜ project object in the device tree or in the POUs view. When adding a POU, specify the POU type and the implementation language. Other programming objects (method, action, etc.) can in turn be added as objects in POUs.

6.9.2. POU – Program

A program is a POU that returns one or more values when executed. All values are retained after execution of the program until the next execution. Define the call order of the programs within an application in task objects.
In the device tree and in the POUs view, the program POUs have the suffix (PRG).
The editor of a program consists of the declaration part and the implementation part.
The topmost line of the declaration part contains the following declaration:
PROGRAM <Name>

6.9.3. POU – Function

A function is a POU that delivers exactly one data element during execution and whose call in textual languages can occur as an operator in expressions. The data element can also be an array or a structure.
Device Tree or in the POUs view, Function POUs have the suffix (FUN).

> **ℹ NOTE**
>
> Functions have no internal status information, which means that functions do not save the values of their variables until the next call. Calls to a function with the same input variable values always return the same output value. Therefore, functions must not use global variables and addresses.

The editor of a function consists of the declaration part and the implementation part.
The topmost line of the declaration part contains the following declaration:
FUNCTION <Name> : <Data type>

Beneath that, you declare the input and function variables.
The output variable of a function is the function name.

6.9.4. POU – Function block

A function block is a POU that supplies one or more values during execution.
In the device tree or in the POUs view, function module POUs have the suffix (FB).
You always call a function block using an instance that is a copy of the function block.
The editor of a function block consists of the declaration part and the implementation part.
The values of the output variables and the internal variables are retained after execution until the next one. This means that the function block does not necessarily supply the same output values, when called several times, with the same input variables.
The topmost line of the declaration part contains the following declaration:
FUNCTION_BLOCK <Name>

6.9.5. Data Unit Type (DUTs)

A DUT (Data Unit Type) describes a user-specific data type.
There are the following types of DUTs:
→ Structure
→ Enumeration
→ Alias
→ Union

You can add an object *DUT* below the application or in the *POUs* view. When you add, you can make certain definitions, see below: Dialog *Add DUT*. Structures and enumerations are explained briefly in this chapter. An explanation of aliases and unions and examples can be found in the "CODESYS V3 Help" described in chapter 6.3.

6.9.6. DUT - Structure

This DUT consists of a structure of different data types. It is thus possible to combine several variables of different data types in one structure. Arrays and other structures can also be used. The only limitation is that you cannot assign addresses to variables (the AT declaration is not allowed).

Syntax for structure declaration:

```
TYPE <Name>:
STRUCT
        <Variable declaration 1>
        ...
        <Variable declaration n>
END_STRUCT
END_TYPE
```

<Name> is a type that CODESYS V3 recognizes throughout the project and that you can use as a standard data type.

6.9.7. DUT Enumeration

An enumeration (list) is a user-defined data type composed of a comma-separated set of components, also called enumeration values. The enumeration values are identifiers for global constants known in the project. Declare an enumeration in a DUT object that you have created in the project using the *Add object* command.
Syntax:

```
TYPE <Name>:
(
        <enum_0>:=<Value>,
        <enum_1>:=<Value>,
         ...,
        <enum_n>|:=<Value>
)
<Basic data type>:=<Standard value>;
END_TYPE
```

<Name> is a type that CODESYS V3 recognizes throughout the project and that you can use as a standard data type.

6.9.8. Global Variables and List of Global Variables (GVL)

Variables declared in POUs can only be used exclusively within this POU under CODESYS V3, with the exception of input and output variables. If you need variables that you want to access from the entire project, you should use so-called global variables. Global variables are "normal" variables, constants, external or remanent variables that are known throughout the project. The system recognizes a global variable by prefixing the variable name with a point, for example, iGlobVar1.
These are declared in global variable lists. A global variable list is used to declare, edit and display global variables. In global variable lists, variables of any type, whether standard data type or DUT, can be declared and function blocks can be instantiated.

6.10. Integrate library in the project

Libraries are an integral part of CODESYS V3; with libraries you can easily combine stand-alone modular components into a container and easily distribute them. Behind almost every CODESYS V3 feature is a library in which the respective functionality is programmed.

CODESYS V3 comes with an extensive portfolio of pre-installed libraries, with a large number of different functionalities. Further supply points for libraries are the "CODESYS Store" or special device libraries from device manufacturers such as drives, gateway modules, complex I/O modules or controllers. Libraries are always installed in the library repository (see chapter 6.2.2) and are thus available to be used; in order to use a library in a project, the library must be integrated from the repository inside the project. This chapter shows this process.

When creating a project and integrating an application in the device tree, the library manager is added automatically. In the library manager, you have an overview of which libraries are integrated in a project, how they are integrated and it is here where new libraries are integrated.
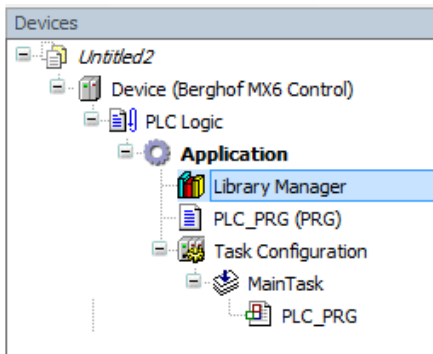
Picture 21: Library Manager 1.1

Double-click on the library manager to open it. For this purpose, it may so happen that libraries are integrated even in a new project, even though you start the library manager for the first time.
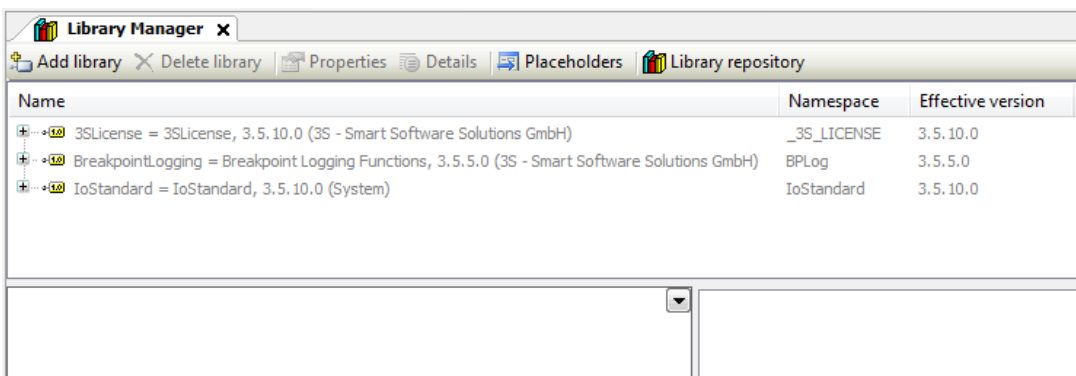
Picture 22: Library Manager 1.2

These already existing libraries are saved with a gray font and are so-called implicitly integrated libraries; these libraries are automatically integrated by the CODESYS V3 system, if one uses CODESYS V3 features of all kinds and should not be changed or removed by the user, as otherwise errors can enter the project. Libraries integrated by the user are always displayed in the default settings with a black font and are always included in the project in the same type and manner. To manually add a library to the project as a user, click on the "Add Library" button; then a new window opens.
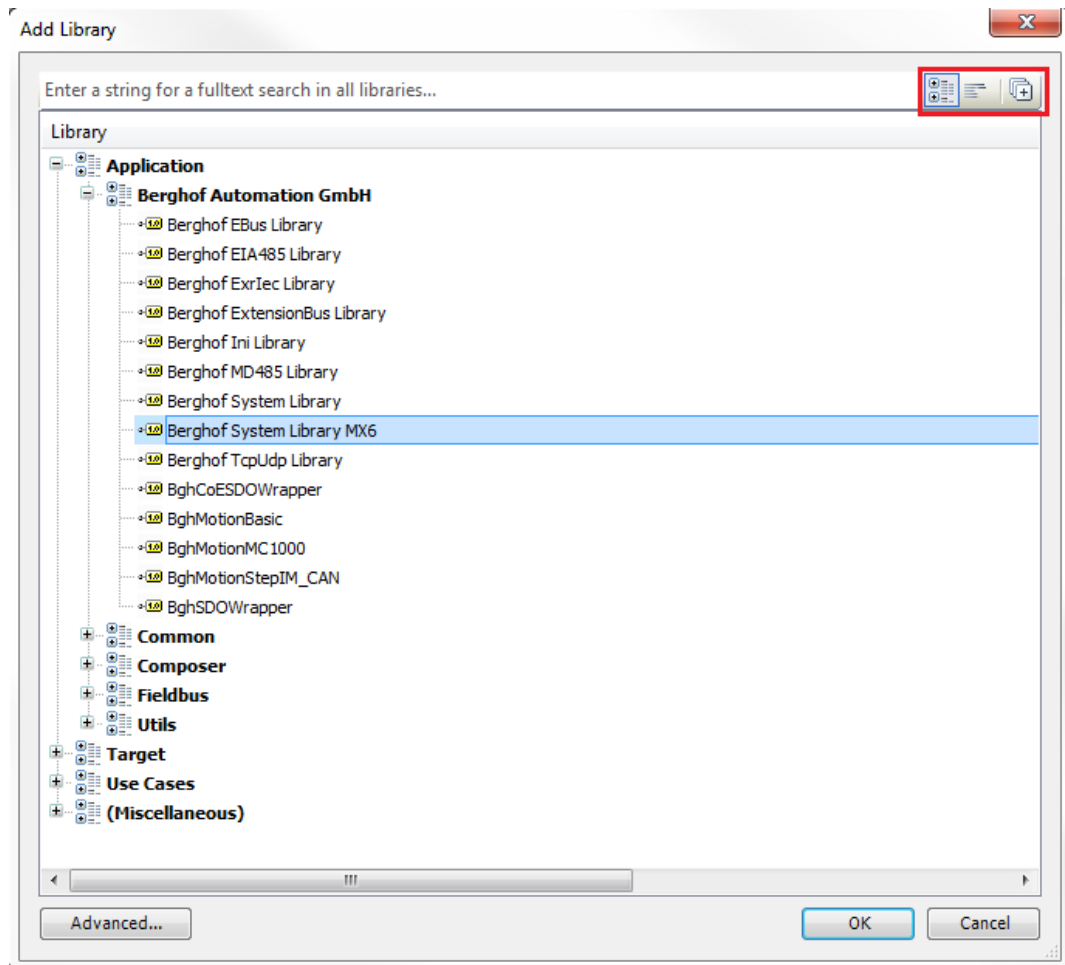
Picture 23: Add library

In the default settings, you get a categorized view displayed on the existing libraries.
Optionally, these can be switched to a list view with the buttons marked in the picture.
The button on the far right with the "plus" activates the extended view in the manager in which even system and low level libraries become visible. With the full-text search, you can search for libraries or even individual functions and other components in libraries. To integrate the desired library, mark it and trigger the integration process by clicking on the "OK" button.

---

i NOTE

It is strongly recommended that only advanced CODESYS V3 users work with the libraries in the extended view. The erroneous use of these libraries in a project can lead to unstable applications.
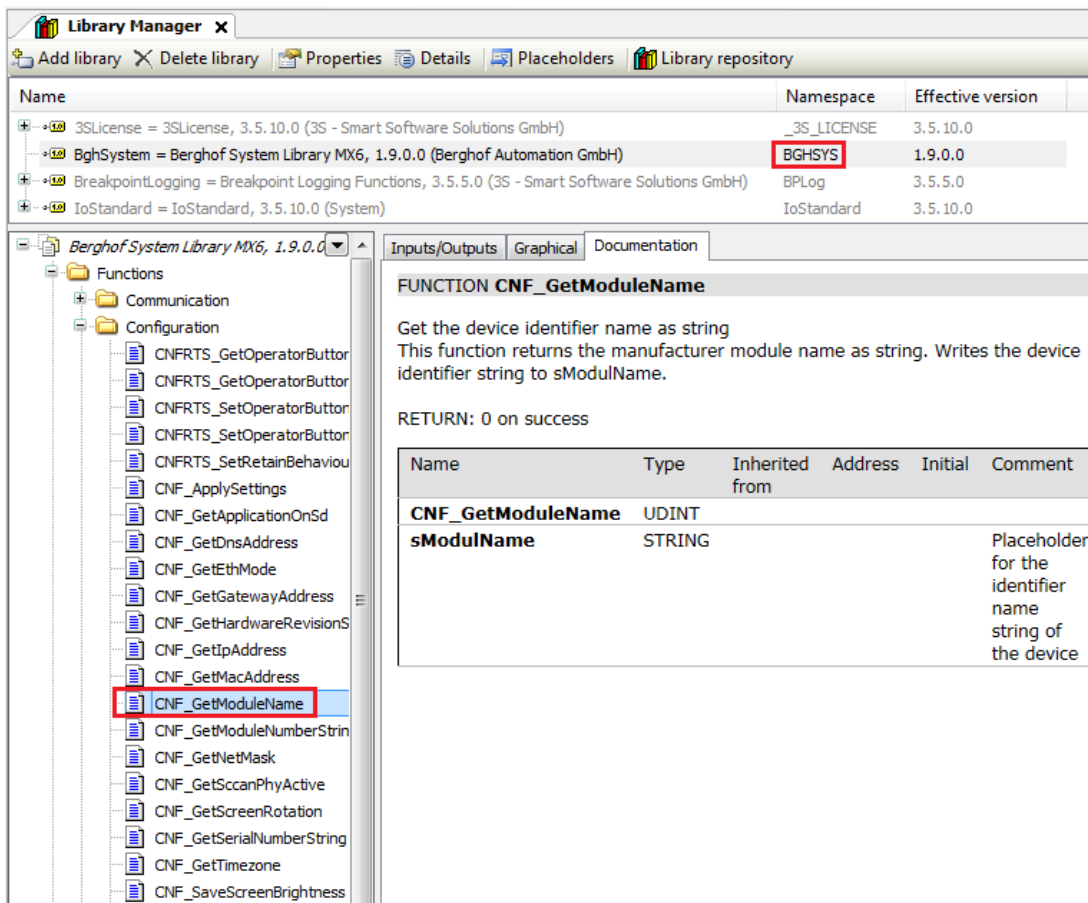
Picture 24: Library Manager

The selected library will then appear in black text in the overview of the library manager.
From now on, all components in this library can be used in the project. The components are recognized globally and can be used in any object you create yourself. If you select the library in the overview, the contents of the selected library appear in the lower half of the screen of the manager. The list of all components that can be used in the library is in the left half; if you mark an object here, information about this object such as interfaces, a graphical view and a short documentation appears on the right side of the screen.
If you now want to use the object, programs and functions can be called directly from the library. Function blocks must be instantiated in the project.
The call or instance statement of a module in the project always results from the Namespace of the library, a point, and the name of the object itself.

Syntax call:
<Namespace>.<Object name>(…);
Example based on the object selected in the picture.

```
1
2
3    BGHSYS.CNF_GetModuleName(sModulName := sName);
4
5
```

Syntax-FB Declaration
<Instance name> : <Namespace>.<Function Block name>;

Example using a timer from the standard library.

```
2    VAR
3        Timer   : Standard.TON;
4    END_VAR
```

6.11. Create visualization

A visualization in CODESYS V3 is a graphical user interface. There is basically a distinction between two different visualizations:
The **"Target visualization"** is needed to use the integrated screen on some Berghof controllers (DC2xxx series). It is also possible to use a target visualization on controllers without a screen; but this is then only provided via a VNC server running on the controller. The target visualization can then be displayed on one or more VNC clients, such as an E-terminal. It should be noted that the target visualization cannot differentiate between different clients. The same picture is output on all connected clients. It is also not possible to use several target visualizations at the same time.
On the other hand, the **"Web visualization"** starts a web server and provides a web page based on HTML5 and JavaScript per web visualization. This web visualization can then be displayed either in a browser or on an HTML5 compatible device. In contrast to target visualization, web visualization also differentiates between individual users who use the same visualization at the same time.
It is also possible to run multiple web visualizations simultaneously. For example, different screens can be controlled and evaluated by one controller.

6.11.1. Visualization Manager

In order to be able to use a visualization in an application, an object of the type **"Visualization Manager"** must first be added. This contains the configuration for the target and web visualization. To do this, select your "Application" object in the device view and right-click to open the context menu. First select "Add Object" and then "Visualization Manager".
After inserting an object of the type "Visualization Manager", a web visualization and a target visualization are automatically created. A new task named "VISU_TASK" will also be created. The graphic interface always runs independently of your other tasks. By default, the "VISU_TASK" has a cycle time of 100ms and a priority of 31. This means that the visualization has the lowest priority and can only get computing time if no other tasks demand it. This ensures that the interface cannot thwart tasks with real-time priority.
After inserting the "Visualization Manager", you can now create objects of type "Visualization". Open the context menu of your application again and add an object of type "Visualization".
Each object of this type represents a single page of the graphical user interface (UI).
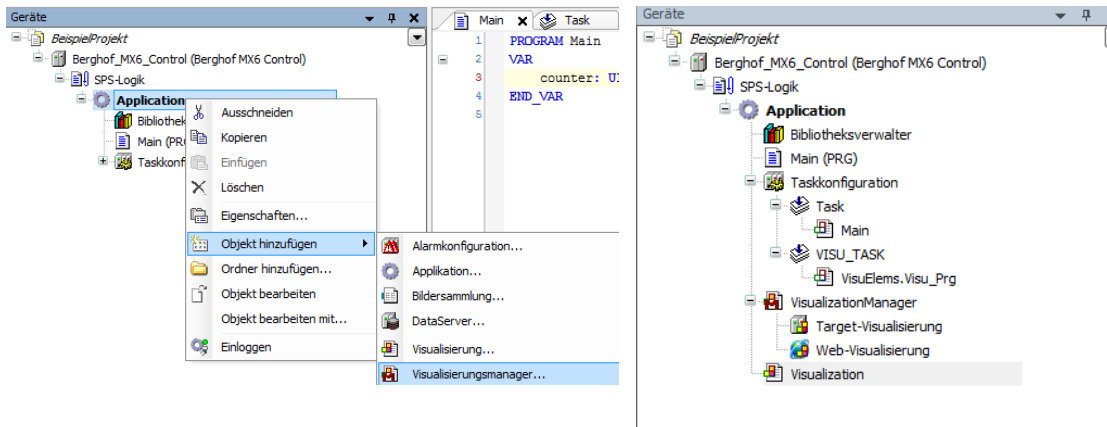
Picture 25: settings for the target and web visualization

Now you have to make some custom settings for the target and web visualization before you can use the newly added visualization.

6.11.2. Settings of target visualization

Open the settings for the target visualization by selecting the respective object below the visualization manager in the device tree and double-clicking.
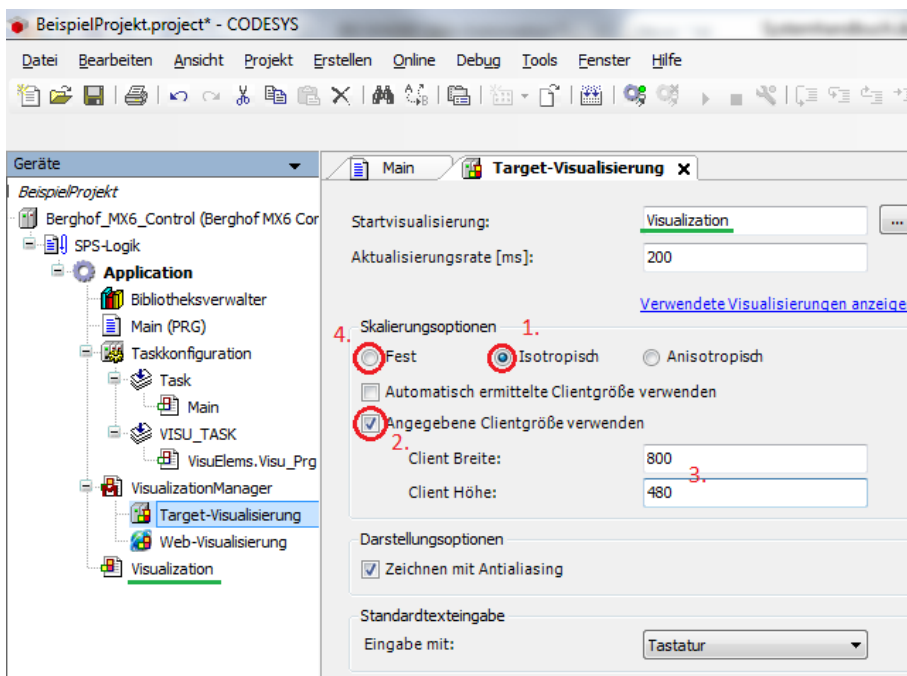
Picture 26: Settings for the target visualization

First we can choose a **start visualization**. This is the visualization that will be displayed when the program is started. Different start visualizations can be specified for target and web visualization. The only visualization that exists in our sample project so far is called "Visualization" (green underlined) so we enter this name in the text field or select it via the input help, which opens when the "..." button is pressed. In order for the visualizations to be displayed later in the correct size, you must specify the resolution of your display used. Therefore, edit the **scaling options** in the following order:

1. Set the scaling to **"Isotropic"**.
2. Activate the checkbox "Use specified client size".
3. Enter the resolution of your display (to find in the manual of the display/the controller).
4. Set the scaling back to **"Fixed"**.

With that, all your visualizations are the same size as your used display and do not need to be scaled, which can lead to distortion or blurring of the visualization elements.

6.11.3. Settings of web visualization

Now open the settings for the web visualization by selecting the corresponding object below the visualization manager in the device tree and double-clicking.
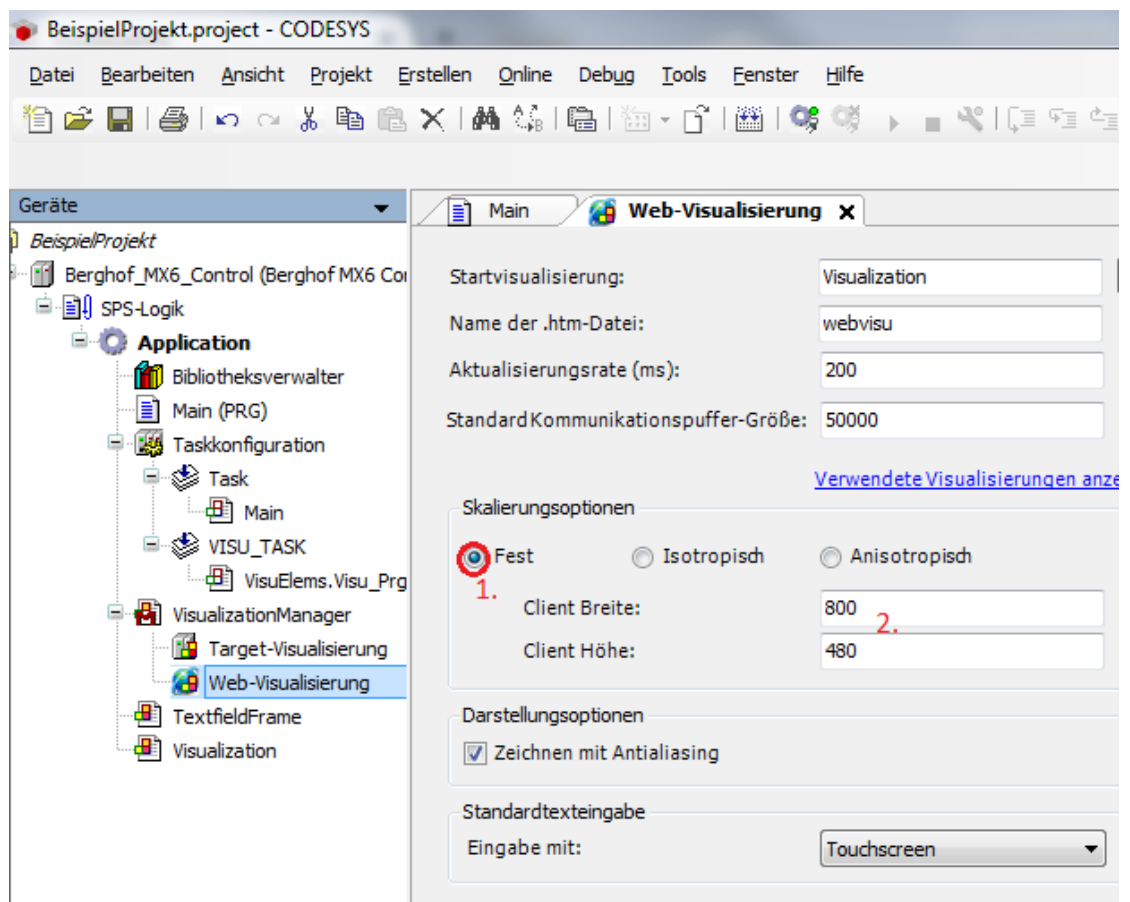
Picture 27: Settings of web visualization

As with the settings of the target visualization, a **start visualization** must also be specified for the web visualization. In this case we use the same visualization, but it is easily possible to use different visualizations for target and web visualization in order to react to different screen sizes or other requirements with that.

You can assign a **name** per web visualization. This name determines the URL under which the web visualization is accessible. The web server of the web visualization(s) runs on port 8080 of the controller. The default path for web visualization is:

http://[IP-Adresse der Steuerung]:8080/webvisu.htm

The **scaling options** allow you to specify how the visualization is displayed in the browser. If the web visualization is to be displayed in the browser in the same size as on the controller or the display, the scaling option **"Fixed"** is the best option. Set the "client width" and "client height" to the same values that you have specified in the settings of the target visualization. The **"Isotropic"** and **"Anisotropic"** options cause the visualization to be scaled in the browser. With the **"Isotropic"** option, the visualization is scaled to the entire width of the browser window, and then the height of the visualization is adjusted while maintaining the aspect ratio. The **"Anisotropic"** option scales the visualization to the full size of the browser window, regardless of the aspect ratio. Under certain circumstances, texts and graphics are then distorted or blurred.

6.12. Visualization editor

After inserting the visualization object, mark it in the device tree and open it with a double-click to edit it. Now a new editor window should open with the visualization editor.
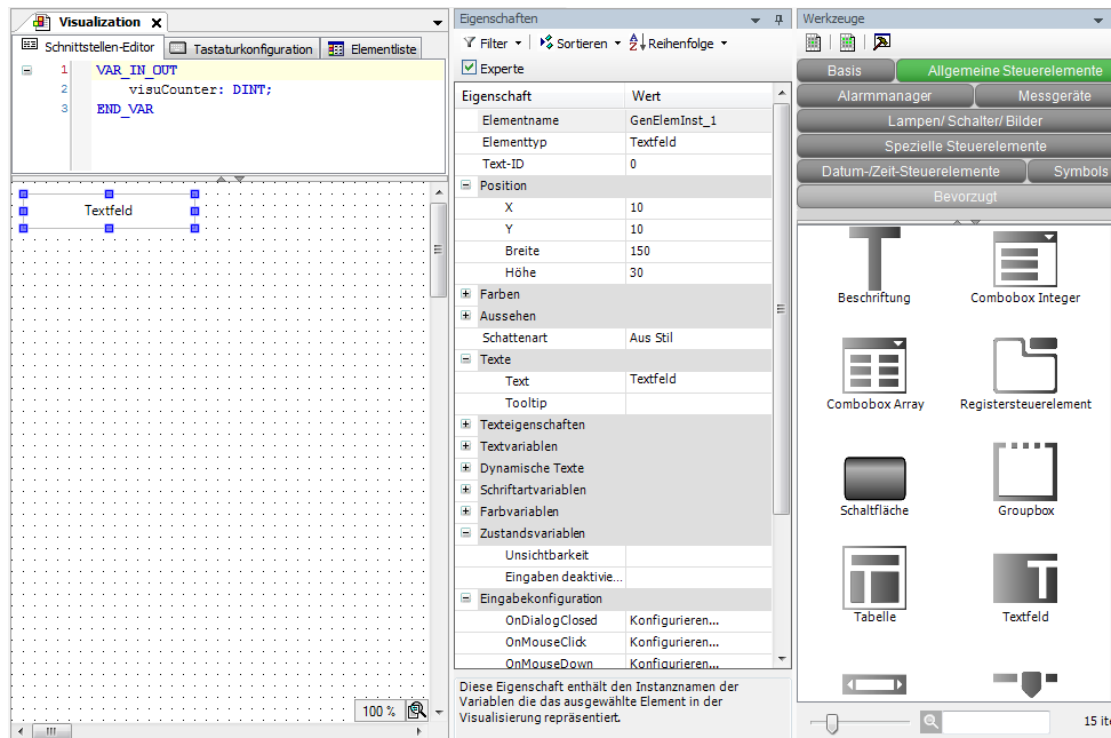
Picture 28: Visualization editor

In contrast to the normal editor window (for programs, modules and functions), the visualization editor has additional sub-windows. The largest part of the window is the visualization editor itself (dotted background), which displays the visualization with all its elements. Above this editor window, there are three windows cumulated in a group of tabs:

The **interface editor** to define variables that are used when displaying the current visualization in a frame.

The **keyboard configuration** to assign specific actions to specific keys or shortcuts.

The **element list** in which all elements in the visualization are displayed in tabular form.

To the right of the editor window, there is another group of tabs with two windows. In the figure above, however, these are shown side by side for reasons of clarity.

The **"Tools"** window in which all ready-made visualization elements are displayed according to categories. From this window, single elements can be simply dragged into the editor window with the mouse and then arranged there as desired.

The **"Properties"** window where all properties of the currently selected element can be viewed and edited. You can also select several elements (using the editor window or the element list) using the control key (Ctrl). Now if you change a property, you will try to change this property for all selected elements.

> **i NOTE**
>
> In the CODESYS V3 visualization, the user is allowed to add and use own pictures. How this works and which elements are required for this is described in detail in the "CODESYS Online Help" under "Using the Image Collection" and the "Visualization Element Image".
>
> Berghof controllers support standard formats such as BMP and JPG as well as transparent formats such as PNG and SVG tiny. SVG standard must be converted to the SVG tiny format before integration; the Freeware Tool 'SVG Converter' is suitable for this.
>
> For performance reasons, it is recommended to include images in the resolution in which the images should also be displayed and not to actively scale these within the application.

6.13. Expand project

The next goal is to create a visualization with several text fields to display some variables from our main program. Thus, the reusability of individual visualizations can be demonstrated right away; we use frames. This allows one visualization to be embedded in another visualization and used multiple times. Subsequently, the internal inputs and outputs of the controller should be used and these can also be controlled via the visualization.

To realize these goals, we first have to expand our main program, that is our "Main" object of the type "Application". In the "Main" program, first define other variables:

→ A variable of type BOOL named "counter_enabled" to activate the first counter variable.

→ Another counter variable of type INT with the name "counter2".

→ A variable of type BOOL named "counter2_enabled" to activate the second counter variable.

→ A variable of type BOOL named "input" to store the value of a digital input.

→ A variable of type BOOL named "output" to set the value of a digital output.

Next, expand the actual program so that when executed, the first counter variable increments (increases the value by 1) and the second counter variable decrements (decreases the value by 1), if the corresponding activation variable has the value "TRUE".

For this, use an IF statement to check the current value of the variable and then execute the corresponding operation. The following figure shows the complete "Main" program.
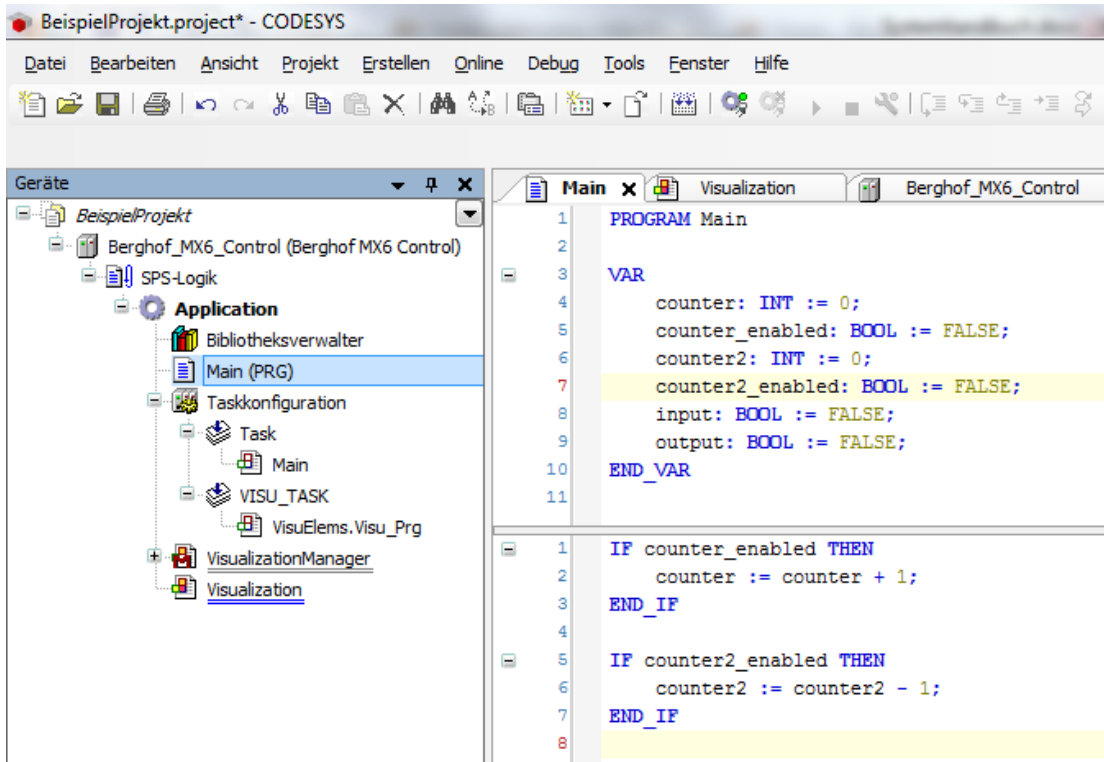
Picture 29: expand the actual program 1.1

In the next step, we now create a second visualization and call this "TextfieldFrame" (1). In this visualization, we now add a **text box** (2, found in the toolbar under the category "General Controls") and then another **toggle switch** (3, found in the toolbar under the category "Lamps/Switches/Images"). In order to be able to equip these two elements with additional functionality, we additionally define two IN-OUT variables in the "interface editor" of the visualization: A variable of the type INT with the name "counter" and a variable of the type BOOL with the name "enabled" (4). After the variables have been so defined, they can be used within the visualization.
So first, we will use the "enabled" variable to set the state of the toggle switch. The toggle switch as a separate element also has the advantage that, apart from this one variable, it requires no further configuration to change its state with a mouse click or finger pressure (on touch displays).
Select the toggle switch in the visualization editor and now set the property "Variable" to the value "enabled" in the "Properties" window on the right edge. So the toggle switch always has the same state as variable "enabled" (5).

Next, select the text field (6) and set the property "Text" (7, to be found in the "Texts" tree in the properties) to the value "%s". This is a so-called placeholder which is replaced by the contents of the text variable when executing the program. This placeholder can also be embedded within a text. Then you have to select the text variable you want to use. To do this, set the "Text variable" property (8) of the text field to the value "counter". Thus, the placeholder is replaced with the value of the variable "counter". Different variable types (such as STRING and INT) can be converted into text using the placeholder.
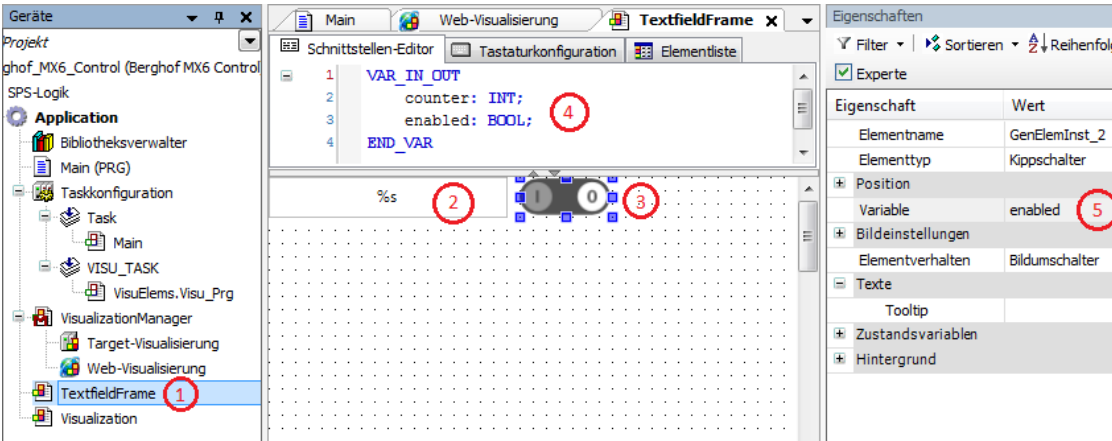
Picture 30: expand the actual program 1.2
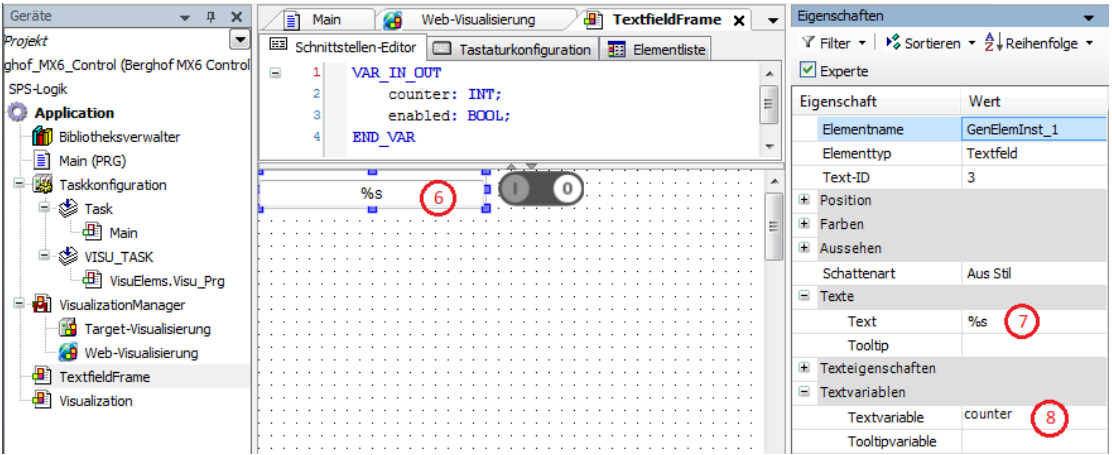
Picture 31: expand the actual program 1.3

6.13.1. Integrate visualization as a frame

Since two different counters were defined in the sample project, you also need two text boxes and two switches to display the counter readings and to activate or deactivate the counters. This is achieved by integrating two frames in another visualization (called "visualization" here), which then each display a "TextfieldFrame" visualization. Multiple nesting is possible with the help of frames. This means that frames can be embedded again in additional frames. So you can use many simple elements to develop a modular and at the same time user-friendly interface.
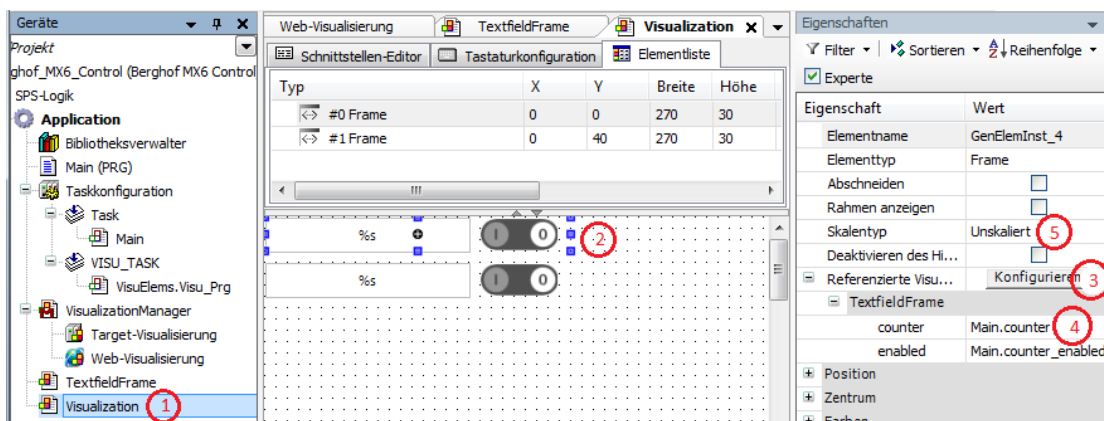
Picture 32: Integrate visualization

Edit the "Visualization" visualization (1) and insert a visualization element "Frame" from the "Basic" category in the "Tools" window. Then select the frame (2) and go to the "Properties" window. Next, search for the "Referenced Visualizations" property and open the dialog for referencing the visualizations with a mouse click on the "Configure..." button(3).
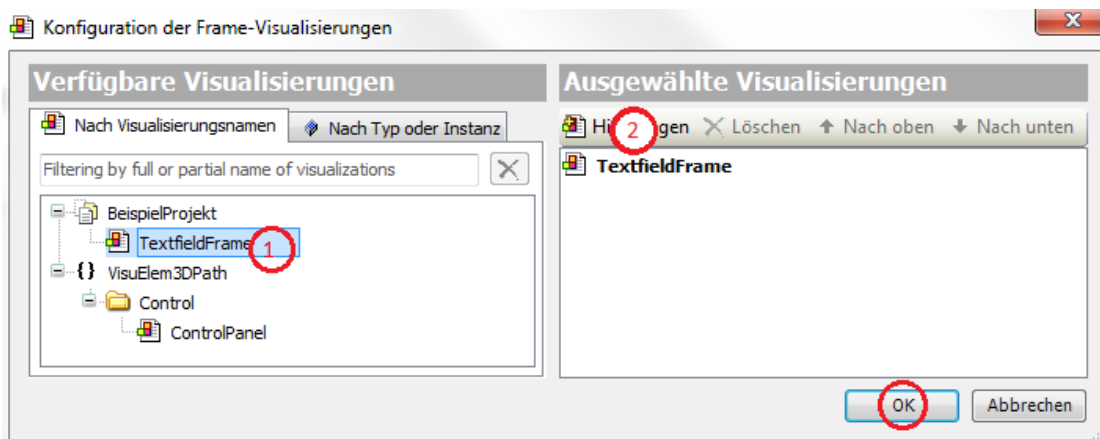
Picture 33: Select the visualization

Select the visualization "TextfieldFrame" in your project (1) and click on "Add" (2). With this you have referenced the visualization with the frame and it is now displayed in the frame. It is also possible to combine multiple visualizations with a single frame. These can then be assigned to a variable of type INT via

the property "Switch variable". If you now change the value of this variable, the corresponding visualization is displayed in the frame. The first visualization has the ID 0, the next ID 1 and so on. Now close the configuration dialog with the "OK" button, then the selected visualization should be displayed in the frame.

Next, you must configure the variables that the frame should pass to the visualization. For each visualization referenced with the frame, all available interface variables are displayed (4).
In our case, we have only two interface variables defined, "counter" (INT) and "counter_enabled" (BOOL). Enter "Main.counter" or "Main.counter_enabled" as the value for the interface variables. This will use the variables from the "Main" program with the name "counter" and "counter_enabled".
Before the frame can now be used, the display options have to be adjusted:
Set the property "Scale Type" to "Unscaled" (5). This displays the entire contents of the frame and does not scale. The "anisotropic" and "isotropic" options cause the content to be adjusted to the size of the frame and scaled up or down accordingly. With "isotropic" the aspect ratio is maintained, but not with "anisotropic".
This completes the configuration of the first frame. You can now copy the frame in the editor window using the key combination "CTRL+C" and then paste it with "CTRL+V". As an alternative to the key combination, you can of course also use the "Edit" menu in the menu bar.
After inserting a copy of the frame, drag the frame to the right place with the mouse. Please note that the variables "Main.counter2" and "Main.counter2_enabled" must be entered as interface variables in the properties of the frame. If you skip this step, the second frame will also show the values of the first counter.

6.14. Use internal inputs and outputs of the controller

All devices of the Dialog Controller, CC Slim and CC-Prime device families have several integrated digital and analog inputs and outputs. The following chapter shows how to integrate, configure, and use these I/Os in CODESYS V3.

ℹ NOTE

Connection instructions can be found in the relevant manual of the corresponding controller.

6.14.1. Integrate inputs and outputs

The integration of hardware into the controller works in the same way as the integration of software objects in the application. By right-clicking on the control unit in the device tree, navigate to the item "Add device" in the opened menu. This will open a new window "Add device".

ℹ NOTE

If the integrated inputs or outputs are not used, it is recommended not to

include them. The real-time behavior of the controller is thereby improved.
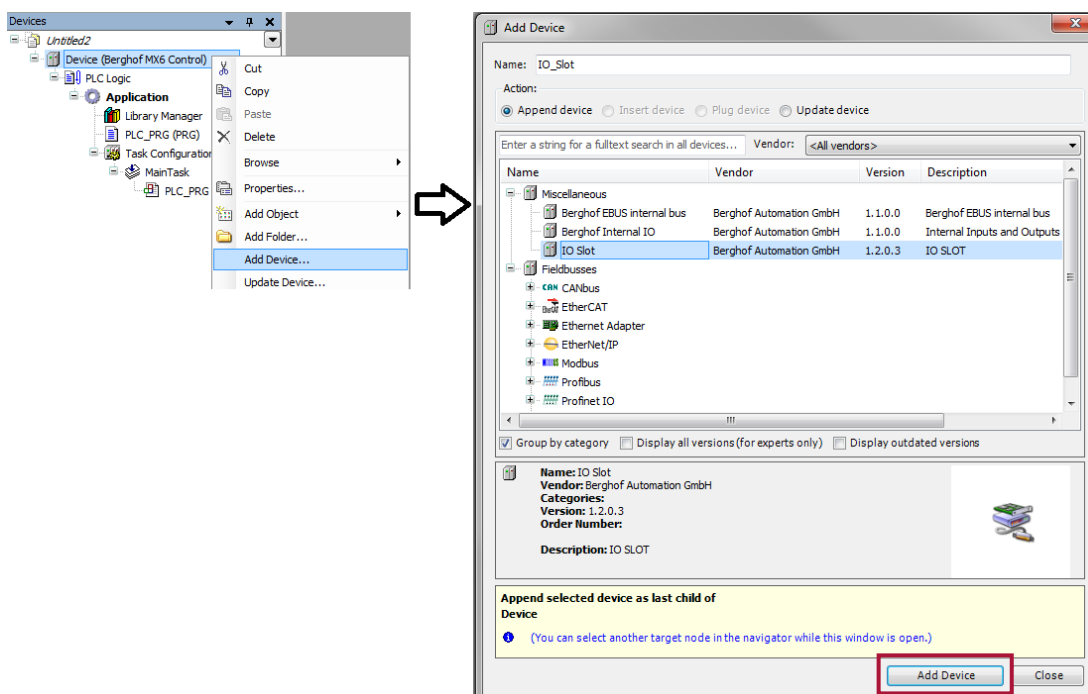
Picture 34: Hardware integration

In the "Add device" window, you now have a categorized overview of all the devices that you can connect to the controller. Among them, all of the fieldbus masters installed in CODESYS V3 and their slaves and the devices for the integrated I/Os.

---

i NOTE

Not all fieldbus masters displayed in "Add device" work with Berghof controllers. Some require an extra license (see chapter 7.1), others are not implemented in the system. If you have any questions, please contact technical support.

In order to now include the integrated I/Os, mark the device "Extension Slots" (depending on the set language, the device can also be called IO Slot) in the category "miscellaneous" and add the device by executing the button "Add Device". The device is now attached to the control unit in the device tree.
However, the connection process is not yet completed, the slot device is only the interface, the actual I/Os will be integrated in the next step.
In order to connect the actual I/Os, mark the first bracket which is attached below the slot device and right-click. In the menu, navigate to the item "Plug Device" and execute it. This will open a window "Plug Device", which shows the available I/O modules, which can be integrated.
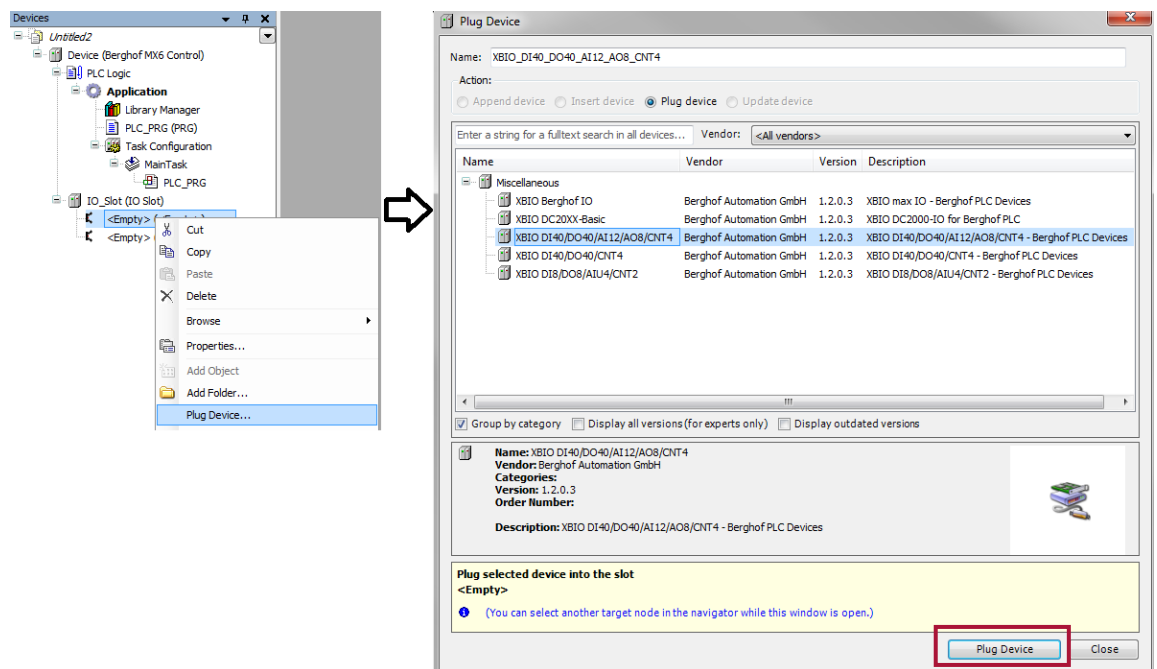
Picture 35: include the integrated I/Os

---

i NOTE

The second bracket below the slot device is included only in the device
description and not realized on the hardware and therefore without function;
it must therefore be left blank.

Overview modules:

XBIO DI40 DO40 AI12 AO8 CNT4:   Suitable for CC-Primedevices with digital and analog inputs and outputs

XBIO DI40 DO40 CNT4:   Suitable for CC-Primedevices only with digital inputs and outputs

XBIO DI8 DO8 AIU4 CNT2:   Suitable for Dialog Controller and CC-Slim devices

XBIO Berghof IO: Identical to XBIO DI40 DO40 AI12 AO8 CNT4, other term for compatibility to older target versions

XBIO DC20XX Basic: Identical to XBIO DI8 DO8 AIU4 CNT2, other term for compatibility
to older target versions

ℹ NOTE

In general, you can use each module for each controller. All versions are based on the fully equipped module XBIO DI40 DO40 AI12 AO8 CNT4, and are only shortened for reasons of clarity in the device description; the core remains the same.

It is generally possible to use the module XBIO DI40 DO40 AI12 AO8 CNT4 for all Berghof MX6 controllers. If inputs or outputs existing in the software are used, which are not present on the hardware side, the system recognizes that and nothing happens.

Mark the module XBIO DI40 DO40 AI12 AO8 CNT4 and connect it to the first bracket of the slot device by executing the "Insert device" button.

6.14.2. Configure inputs and outputs

If the desired device has been successfully integrated, it must be configured before first use.
The configuration window opens by double-clicking on the device, in this case XBIO DI40 DO40 AI12 AO8 CNT4 in the first bracket of the slot device. Here, you can configure the analog as well as counter and encoder inputs.
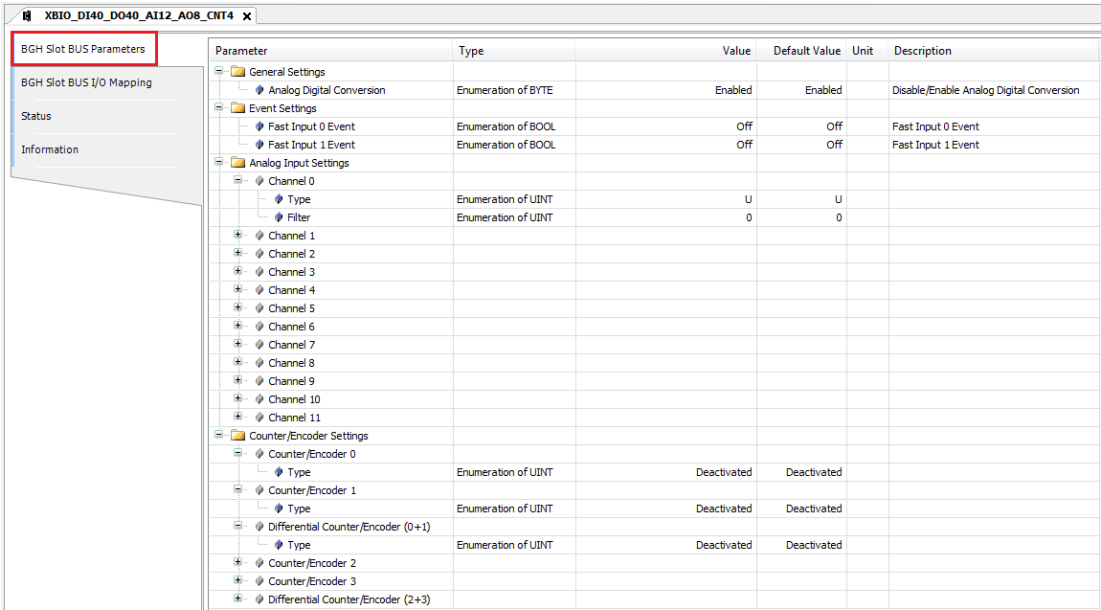
Picture 36: Configure inputs and outputs

The settings for the analogs as well as counter and encoder inputs can be found in the tab "BUS Parameter".

Overview of settings:

Analog Digital Conversion:
Activates the analog inputs and outputs; a drop-down menu opens with one click in the corresponding table cell in the column "value".
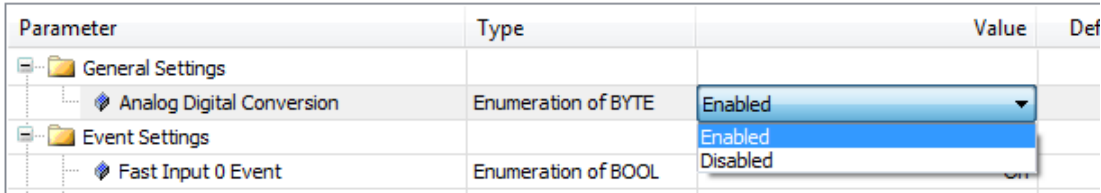
Picture 37: parameter

ℹ **NOTE**

If the analog inputs or outputs are not used, it is recommended to disable them. The real-time behavior of the controller is thereby improved.

Fast Input 0/1 Event:

Fast-switching inputs which can trigger a Trigger system event in the application. Clicking on the corresponding table cell in the "Value" column opens a drop-down menu.
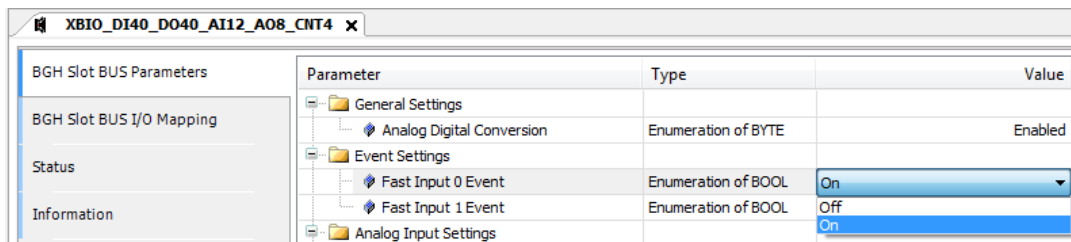
Picture 38: Fast Input

Afterwards, a system event of type "FastIn0/1" must be defined in the configuration task (see "CODESYS Online Help").
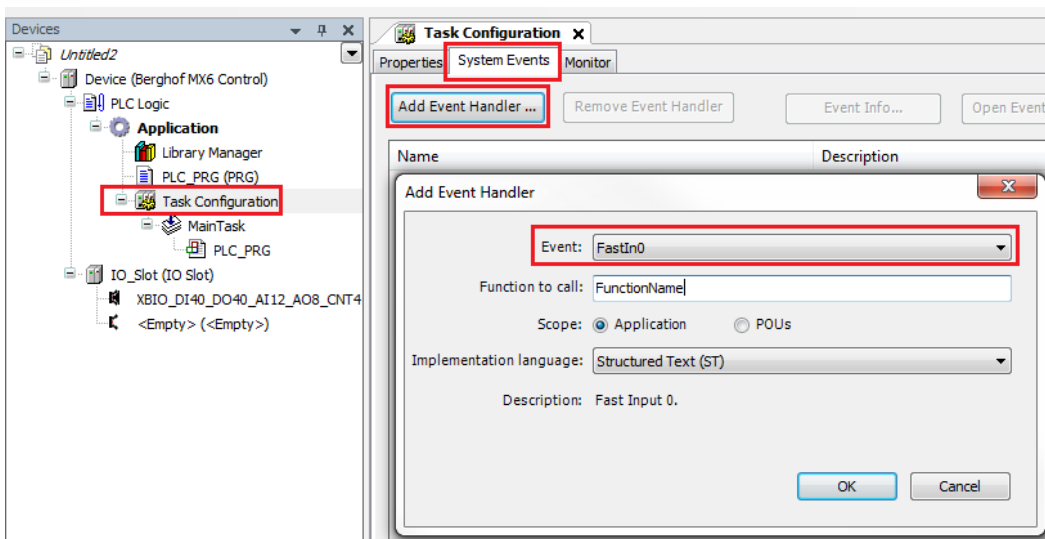
Picture 39: Task Configuration

ℹ **NOTE**

If this functionality is used, the Berghof library "Berghof ExtensionBus Library" must be integrated into the application.

Channel 0-11 Input Settings:

Settings for analog inputs, selection of the signal type and setting of a filter.

Clicking on the corresponding table cell in the "Value" column opens a drop-down menu.

Under "Type", there is the choice between voltage, current and temperature (2 and 3-wire).
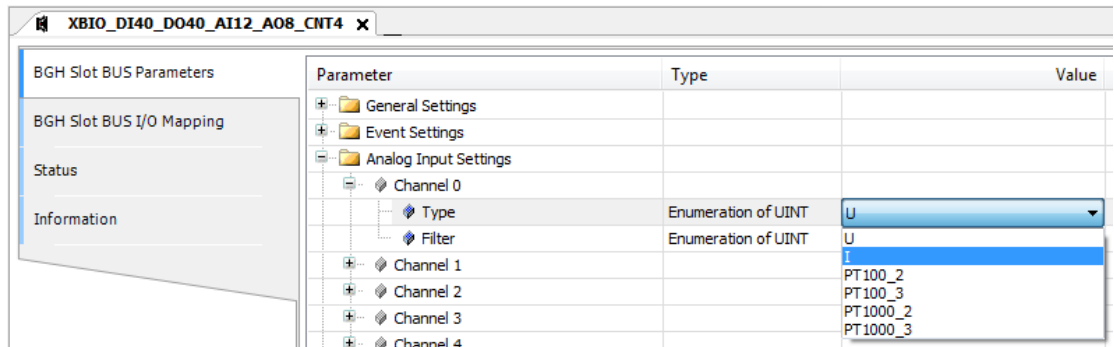
Picture 40: Analoge input type

Under "Filter", you choose between a 10Hz, 100Hz, 1000Hz or no filter. Here, you can set whether an averaging 10, 100 or 1000 times the second is executed or not at all. Clicking on the corresponding table cell in the "Value" column opens a drop-down menu.
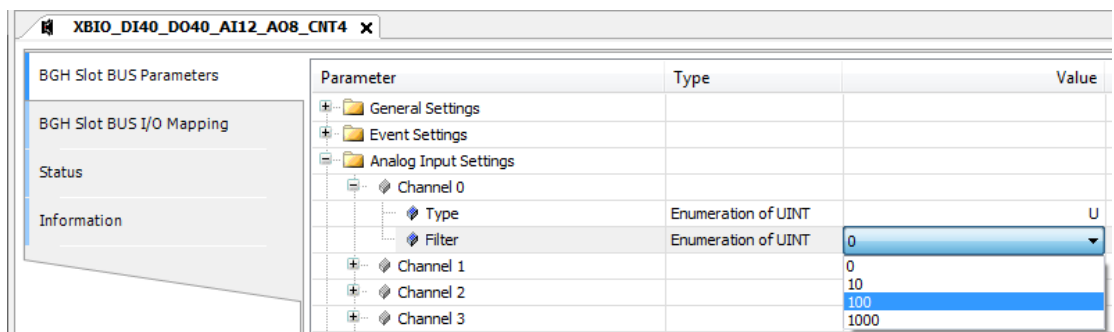
Picture 41: Analoge input filter

Counter/Encoder 0-4 Settings:

A few points must be observed in the counter settings.

In order to use the counter functionality, an EC Compact controller with the minimum version 0200 is required; in addition, the counters must be activated via a paid license (see chapter 7.1). On the hardware, Counter 0 and Counter 1 are available. Each counter is controlled by two fast inputs; for better understanding these are called Counter_In1 and Counter_In2.

More detailed information on the Pinout for the two counters can be found in the manual of the EC Compact controllers.
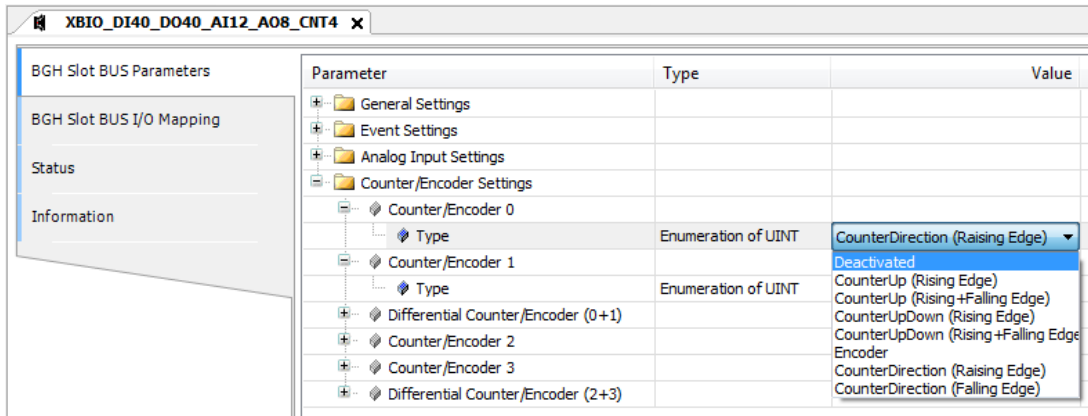
Picture 42: Encoder input type

Clicking on the corresponding table cell in the "Value" column opens a drop-down menu. Under "Type", there is the choice between the different types of counters.

Deactivated: deactivates the counter.

CounterUp (Rising Edge): If input Counter_In1, has a rising edge, the counter value is incremented by 1.

CounterUp (Rising+Falling Edge): If input Counter_In1 has a rising or falling edge, the counter value is incremented by 1.

CounterUpDown (Rising Edge): If input Counter_In1 has a rising edge, the counter value is incremented by 1. If a rising edge is detected at the input Counter_In2, the counter value is decremented by 1.

CounterUpDown (Rising+Falling Edge): If input Counter_In1 has a rising or falling edge, the counter value is incremented by 1 each time. If a rising or falling edge is detected at the input Counter_In2, the counter value is decremented by 1 each time.

CounterDirection (Rising Edge): Depending on the state of input Counter_In2, the counter value is either incremented or decremented by 1 when detecting a rising edge in the input Counter_In1. If Counter_In2 FALSE is decremented, Counter_In2 TRUE is incremented.

CounterDirection (Rising Edge): Depending on the state of input Counter_In2, the counter value is either incremented or decremented by 1 when detecting a falling edge in input Counter_In1. If Counter_In2 FALSE is decremented, Counter_In2 TRUE is incremented.

In addition to the function set in the parameter, Counter_In2 of Counter1 is also the capture input of Counter0. A capture function for Counter1 does not exist.

6.14.3. Use inputs and outputs

If the configuration of the inputs and outputs has been adapted to your own wishes, the connection be-tween the devices and the application is now established. This is done by mapping the automatically as-signed inputs and outputs to variables inside the application.
The individual addresses of the I/Os can be found in the tab "BGH Slot BUS I/O Mapping"
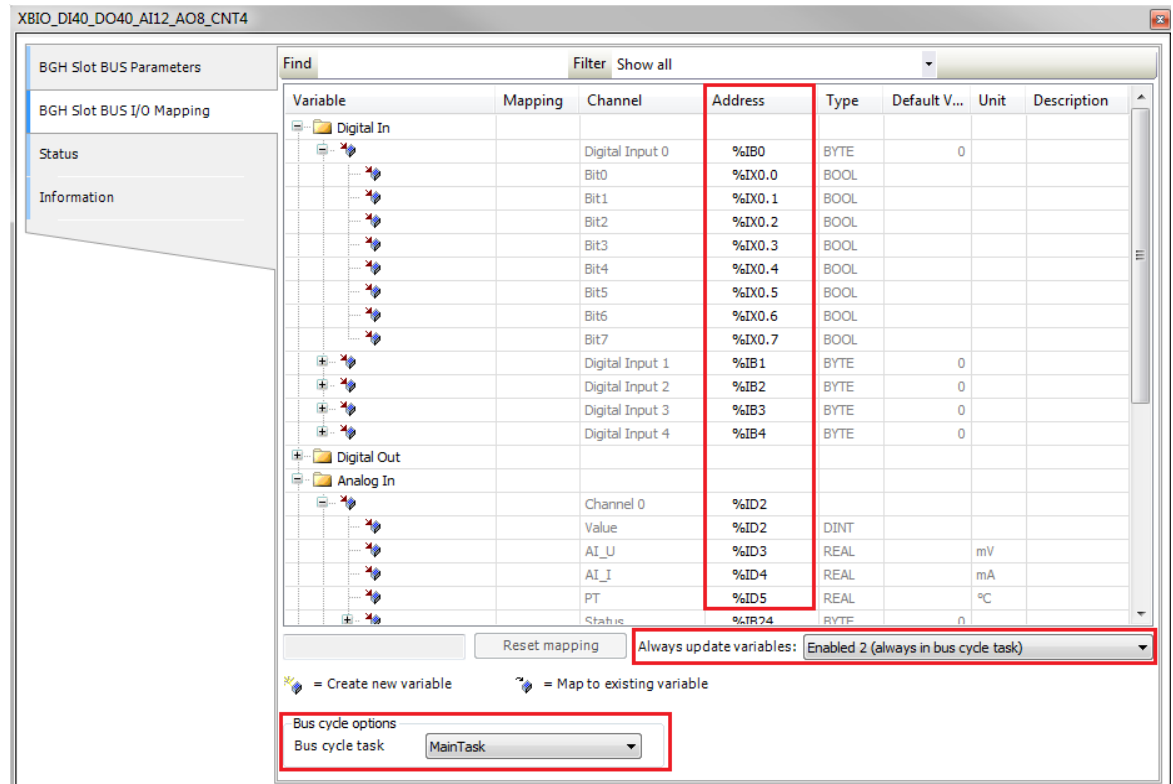
Picture 42: I/O Mapping

Before starting to map the variables, it is recommended to set the I/O update to a self-defined bus cycle task. The bus cycle task defines how fast CODESYS V3 should cyclically update the I/Os. Whether you create a new task or use an existing one is up to the user. Through the drop-down menu in the "Bus cycle options", you can get all tasks defined in the project and can select the desired one. If you have selected the bus cycle task, it is recommended to set the option "Always update variables" via the drop down menu to "Enabled 2" so that all variables are updated cyclically; otherwise, CODESYS V3 only updates the addresses which are mapped to a variable and this variable is also used actively in the project.
Now, in order to map a variable with the device, you must assign the variable the value of the address in the I/O mapping tab. The simplest method for this is the so-called "AT declaration". Here, you assign the address to the corresponding variable in the declaration with an "AT" command. The appropriate data type is taken from the Type column in the I/O Mappings tab. It is possible to declare variables with as-signed addresses in local as well as global objects. In common practice, however, the I/O variables are declared in global variable lists.

ℹ NOTE

As described in chapter 6.14.1, you should not be confused by the greater number of inputs and outputs than available on the hardware.

The link always begins with the first input or output type in the description and links the addresses according to the series, as long as the hardware is available.

for example, in a device that has 16 digital inputs, if you link the address of the 17th input with a variable, then the variable in the software will work normally and in the hardware this will lead to nothing.

Syntax AT Declaration:

<Variable name>   AT %I/Q<Address value> : <Data type>;

%I stands for input addresses, %Q for output addresses

If the variable with the "AT" assignment has been declared correctly, the linked variables can be used like any other variable in the project.

Example of digital inputs:

```
VAR_GLOBAL
    gxDI0   AT  %IX0.0  :   BOOL;
    gxDI1   AT  %IX0.1  :   BOOL;
    gxDI2   AT  %IX0.2  :   BOOL;
    gxDI3   AT  %IX0.3  :   BOOL;
    gxDI4   AT  %IX0.4  :   BOOL;
    gxDI5   AT  %IX0.5  :   BOOL;
    gxDI6   AT  %IX0.6  :   BOOL;
    gxDI7   AT  %IX0.7  :   BOOL;
END_VAR
```

Variable	Mapping	Channel	Address	Type
📁 Digital In				
🔧		Digital Input 0	%IB0	BYTE
🔧		Bit0	%IX0.0	BOOL
🔧		Bit1	%IX0.1	BOOL
🔧		Bit2	%IX0.2	BOOL
🔧		Bit3	%IX0.3	BOOL
🔧		Bit4	%IX0.4	BOOL
🔧		Bit5	%IX0.5	BOOL
🔧		Bit6	%IX0.6	BOOL
🔧		Bit7	%IX0.7	BOOL

Picture 43: Mapping digital inputs

Example of digital outputs:

```
VAR_GLOBAL
    gxDO0   AT  %QX0.0  :   BOOL;
    gxDO1   AT  %QX0.1  :   BOOL;
    gxDO2   AT  %QX0.2  :   BOOL;
    gxDO3   AT  %QX0.3  :   BOOL;
    gxDO4   AT  %QX0.4  :   BOOL;
    gxDO5   AT  %QX0.5  :   BOOL;
    gxDO6   AT  %QX0.6  :   BOOL;
    gxDO7   AT  %QX0.7  :   BOOL;
END_VAR
```

📁 Digital Out				
🔧		Digital Output 0	%QB0	BYTE
🔧		Bit0	%QX0.0	BOOL
🔧		Bit1	%QX0.1	BOOL
🔧		Bit2	%QX0.2	BOOL
🔧		Bit3	%QX0.3	BOOL
🔧		Bit4	%QX0.4	BOOL
🔧		Bit5	%QX0.5	BOOL
🔧		Bit6	%QX0.6	BOOL
🔧		Bit7	%QX0.7	BOOL

Picture 44: Mapping digital outputs

Example of counter inputs:

```
VAR_GLOBAL
    gdiCountVal     AT %ID62    : DINT;
    gdiCapVal       AT %ID63    : DINT;
    gudiCapEventVal AT %ID64    : DINT;
END_VAR
```

Counter/Encoder			
Counter/Encoder 0		%ID62	
Counter Value		%ID62	DINT
Capture Value		%ID63	DINT
Capture Event Counter		%ID64	UDINT
Status		%IB260	BYTE
Counter/Encoder 1		%ID66	

Picture 45: Mapping encoder inputs

Example of analog inputs:

Analog In			
Channel 0		%ID2	
Value		%ID2	DINT
AI_U		%ID3	REAL
AI_I		%ID4	REAL
PT		%ID5	REAL
Status		%IB24	BYTE
Channel 1		%ID7	
Value		%ID7	DINT
AI_U		%ID8	REAL
AI_I		%ID9	REAL
PT		%ID10	REAL
Status		%IB44	BYTE
Channel 2		%ID12	
Value		%ID12	DINT
AI_U		%ID13	REAL
AI_I		%ID14	REAL
PT		%ID15	REAL
Status		%IB64	BYTE

```
VAR_GLOBAL
    grAI0_Voltage   AT %ID3     : REAL;
    grAI1_Temp      AT %ID10    : REAL;
    grAI2_Current   AT %ID14    : REAL;
END_VAR
```

Picture 46: Mapping analoge inputs

---

i NOTE

When mapping to the addresses of the analog inputs, the setting of the signal type must be observed. Depending on the setting, only the address corresponding to the relevant signal type is output in the analog channel.

e.g. in my channel 0, I've specified the input type as a voltage, so in my I/O mappings I need to link my variable with the address for AI_U.

In this, the system automatically converts the signal values into the corresponding signal type, and you get either mV, mA or °C as the read-in value of the addresses.

Example of analog outputs:

```
VAR_GLOBAL
    grAO0_Voltage   AT %QD2 : REAL;
    grAO1_Voltage   AT %QD3 : REAL;
    grAO2_Voltage   AT %QD4 : REAL;
END_VAR
```

Analog Out			
Channel 0		%QD2	REAL
Channel 1		%QD3	REAL
Channel 2		%QD4	REAL
Channel 3		%QD5	REAL

Picture 47: Mapping digital outputs

6.15. Use inputs and outputs via EtherCAT™

Integrating EtherCAT™ hardware into the controller works in the same way as including the integrated I/Os. By right-clicking on the control unit in the device tree, navigate to the item "Add device" in the opened menu. This will open a new window "Add device".
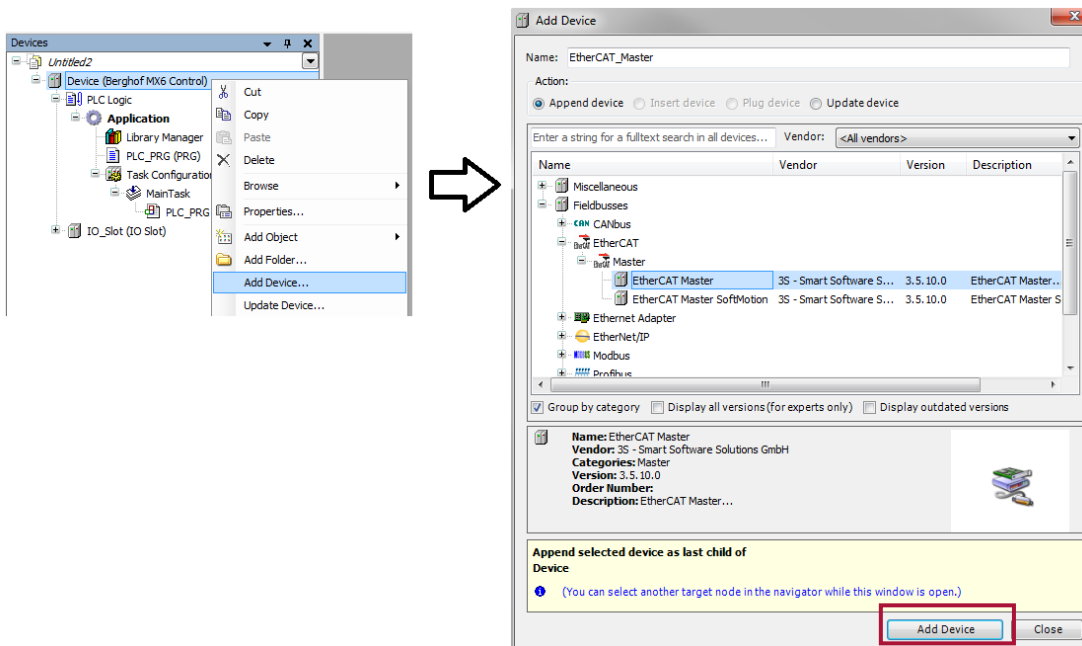
Picture 48: Add Device

In the "Add device", you now have a categorized overview of all devices which you can connect to the controller. Including all of the fieldbus masters installed in CODESYS V3 and their slaves and the devices for the integrated I/Os.

---

i   NOTE

The Berghof EtherCAT™ modules and other EtherCAT™ modules must be installed in the repository prior to integration. Usually, a hardware description file in XML format is used for this. This file is always provided by the manufacturer of your modules.

To integrate the EtherCAT™ I/Os, you first need the EtherCAT™ master. To do this, mark the device "EtherCAT Master" in the category "Fieldbus-> EtherCAT-> Master" and add the device by executing the "Add device" button. The device is now attached to the control unit in the device tree. Necessary libraries integrate the system into the background in the project. In addition, another POU call is automatically added in the MainTask and the cycle time is set to 4 ms. This is a default setting and can be changed afterwards. One can adjust the cycle time or e.g. create a separate task for the EtherCAT™ and shift the POU call in it, but calling this POU is absolutely necessary.

6.15.1. Configure EtherCAT™ Master

Before additional devices can be attached to the EtherCAT master, it must be configured. Double-clicking on the EtherCAT™ Master device in the device tree opens the configuration window.
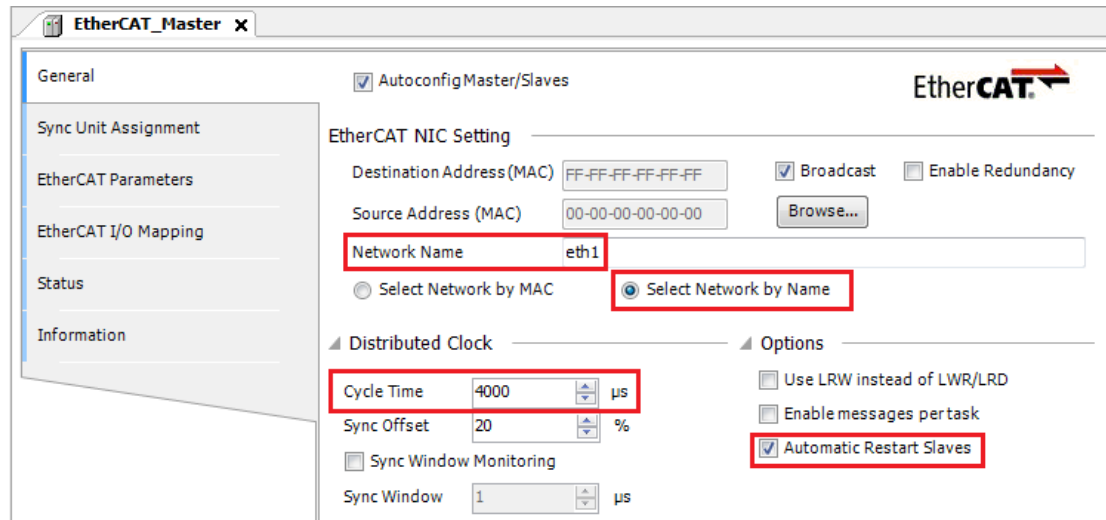
Picture 49: Configure EtherCAT™ Master

In the standard configuration only the network interface, which CODESYS is going to use to run the EtherCAT master, must be defined. To do this, select the option "Select network by name" in the settings; then the text field can be edited.

Now enter "eth1" in the text field. For the EtherCAT™ to work, in the network settings on the web interface of the controller, EtherCAT mode must be selected on eth1.

With the selection of the network interface, the master is functional, but it is recommended to make a few more settings. The "Cycle Time" for the "Distributed Clock" should be the same value or a multiple of the EtherCAT bus cycle task which calls the EtherCAT™ POU.

In the "Options", you can configure that EtherCAT™ slaves are automatically restarted in the event of lost communication, by setting a checkmark for "Automatic restart slaves".

The last setting is in the "EtherCAT I/O Mapping" tab. It is recommended to set the task as bus cycle task in the bus cycle options, in which the EtherCAT™ POUs are also called, newer versions of CODESYS will add an EtherCAT Task and select the bus cycle task automatically.
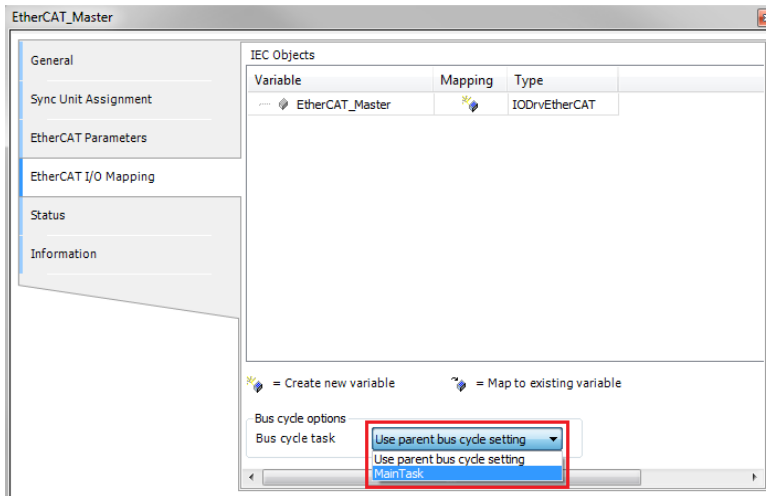
Picture 50: EtherCAT Master bus cycle task

When the configuration is closed, the settings are applied and EtherCAT™ slaves can now be added.

6.15.2. Add and use EtherCAT™ slaves

The integration of EtherCAT slaves works with the same principle that has been used in this manual.
When integrating EtherCAT™ slaves, it must be ensured that the order of the slaves in the control config-uration and the order of the connected modules is identical.
Now, to connect the actual EtherCAT™ slaves, mark the EtherCAT™ master and right-click. In the menu, navigate to the item "Add Device" and run it. A "Add device" window then opens, showing the available EtherCAT™ slave modules that can be integrated.
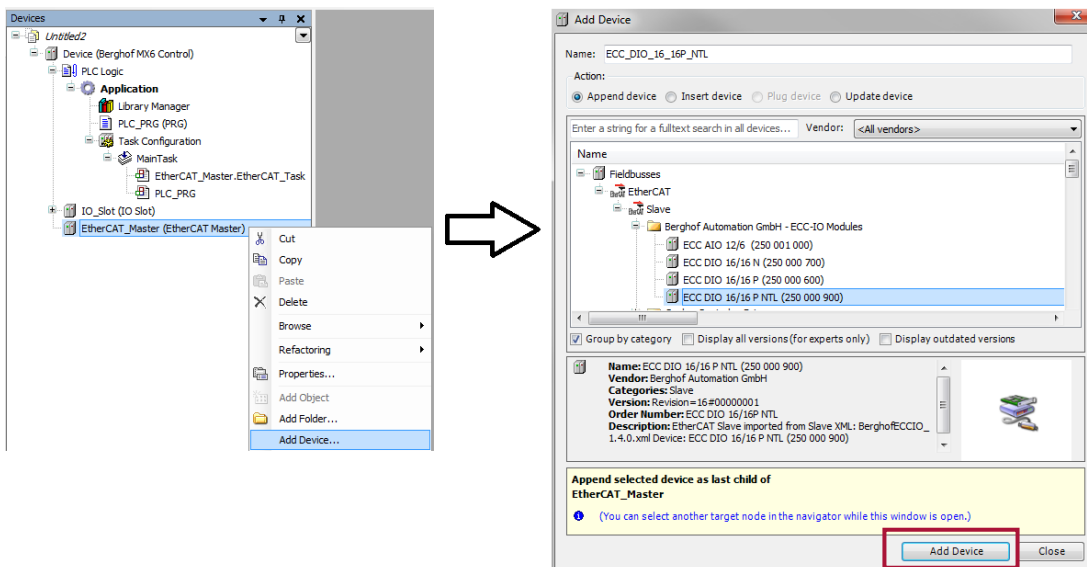
Picture 51: Add and use EtherCAT™ slaves

Navigate to the desired EtherCAT™ slave; in this example, a Berghof CC-DIO; mark this device and press the "Add Device" button to attach the device to the EtherCAT™ master.

ℹ NOTE

If you are using a B-Nimis MC PLC make sure that the XML device description file for the MC-IO is installed. The first device that must be added under the EtherCAT Master is the "onboard Buscoupler". After that is added it is possible to add the MC-IO module into the EtherCAT configuration.

ℹ NOTE

The attachment of fieldbus masters and their slave modules works on the same principle for all supported fieldbuses. You insert the main interface and the desired master and make the basic settings there. Subsequently, the slave modules are attached to the respective fieldbus master and configured. Almost all slaves must first be installed into the repository via a device description file so that they can be used in CODESYS V3. Each inserted device will have a tab I/O mapping in its configuration window, listing the inputs and outputs and their addresses. The mapping to project variables always works the same way, preferably with the AT declaration (see chapter 6.14.3).

If the desired devices have been added in the project, they can already run in their standard settings. Like every device, the configuration window can be opened by double-clicking on the EtherCAT™ device in the device tree; no further settings are required for standard use. As also described in chapter 6.14.3, it is recommneded that the option "Always update variables" located in the "EtherCAT IO Mappings" tab, is set to "Enabled 2" so that all I/O variables are cyclically updated by the bus cycle task.
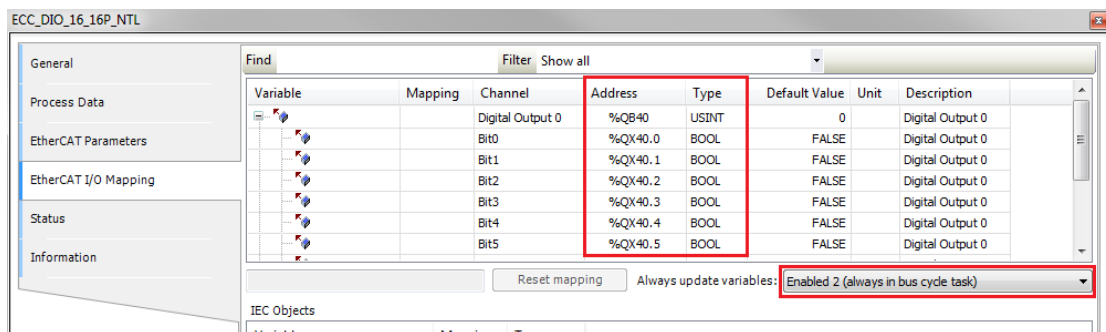
Picture 52: Configure EtherCAT™ slaves

The mapping of project variables to the I/Os is the same for the EtherCAT™ slaves as described in chapter 6.14.3; look in the "EtherCAT™ I/O Mappings" for the address and the data type of the input or output and declare a corresponding variable with the AT method in its project.

6.15.3. Automatic insertion of EtherCAT™ Slave modules

As an alternative to manual integration that is already explained, there is also a more convenient way to integrate modules. It is possible to search for these and integrate them into the project automatically. However, an executable and compilable application must already be available, and all desired slave modules must already be connected to the controller and switched on, as well as installed in the repository.

If the application can be compiled completely without errors, add an **"EtherCAT_Master"** to the project as in the section described above and assign the correct network name **"eth1"**. Additional devices do not need to be added. Then log in to the controller via **"Online -> Login"** and load the application onto the controller. The application can stay in stop and does not have to be started.
Right-clicking on **"EtherCAT_Master -> Scan for devices"**, opens a new window.
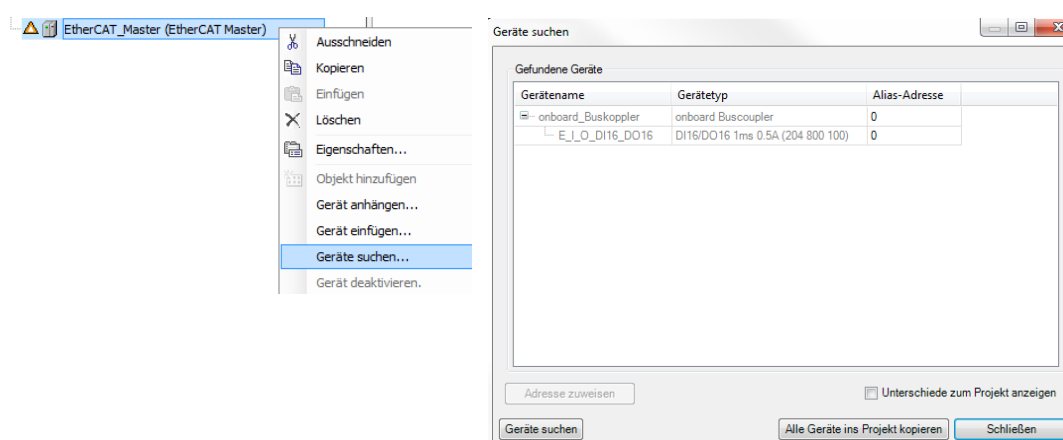
Picture 53: Device search

The new window is blank at first and automatically searches for connected devices. Wait for the search process to be finished and the connected devices that are found appear in the window. With a click on the button **"Copy all devices to the project"**, all found devices are appended below the **"EtherCAT_Master"**.
Then log out once with **"Online -> Logout"** and log in again. The application must then be reloaded once again. The additional modules have thus been successfully integrated.

6.16. Online Mode and Debugging

The logged-in state of CODESYS V3 while the controller is running is also called online mode. The online mode offers a number of additional possibilities, such as the automatic addition of EtherCAT™ modules, the control of the application without the visualisation, monitoring and debugging.
In online mode, variables can be observed and even changed directly by the user, so you can influence the program flow without direct input via the visualization.

6.16.1. Monitoring and control in online mode

If an application is controlled by a visualization or I/Os, only the variables that are linked to the respective visu elements or I/Os are changed in the application. In online mode, however, these variables can also be changed directly by the user, and thus it is possible to manipulate processes in the application without direct inputs via the visualizations or readout of I/Os.
Open **"PLC_PRG"**, leave the editor window open and log into the controller. When the application is fully loaded, start it. Also, activate the flow control under **"Debug -> Flow control"** in the menu bar. The flow control displays the lines in green in the editor window of **"PLC_PRG"**, which are currently being executed. This view of **"PLC_PRG"** shows the current values of the monitorable variables contained in it, in a table in the declaration part and also in the implementation part.

➔ The upper part shows the monitorable variables of the object in a table, this means, the corre-
sponding variables with data type and current value.

Expression	Type	Value
◆ udiCPUload	UDINT	35
◆ xRed	BOOL	FALSE
◆ xYellow	BOOL	FALSE
◆ xGreen	BOOL	FALSE
◆ xPowerOn	BOOL	FALSE
◆ uiStep	UINT	10
◆ xStart	BOOL	FALSE
⊞ ◆ tmrBlink	TON	
◆ tBlink	TIME	T#1s

➔ In the lower part of the view, you can see the code lines as entered in the offline mode, supple-
mented by the small windows of the inline monitoring behind each variable, which show their actual
value.

```
udiCPUload[   35   ] := SchedGetProcessorLoad(0)[ 122455(

IF xPowerOn[FALSE] = TRUE THEN    // When Rockerswitch
    xGreen[FALSE] := TRUE;          //green Lamp On
    gxOutGreen[FALSE] := TRUE; //Set output QX0.0

    //Blink Algorithm
    IF xStart[FALSE] = TRUE THEN    // When Pushswitch
            CASE uiStep[ 10 ] OF //Case Instruction ju
                10:             //State 10 -> Red is on
                    xRed[FALSE] := TRUE;
                    xYellow[FALSE]:=FALSE;
                    gxOutYellow[FALSE] := TRUE;
                    gxOutRed[FALSE] := FALSE;
                    tmrBlink(IN[FALSE]:=TRUE, PT[
                    IF tmrBlink.Q[FALSE] = TRUE THEN
                        tmrBlink(IN[FALSE] := FALSE);
                        uiStep[ 10 ]:= 20;
                        gdwCnt[   0   ] := gdwCnt[
            END_IF
```

6.16.2. Debugging

Debugging is finding the error and rectifying it in the program code if the program does not work as in-
tended or if it does not assume desired states. In addition to writing arbitrary variables to cause different
states and to control the application, it is important to stop the application at any point and then process it
line by line. This will allow the programmer to keep a close eye on what's really going on in the event of
an error in the program.
In online mode, breakpoints can be set, from where the execution of the program must be stopped. When
a breakpoint is reached, the program can also be executed step by step. At each breakpoint or step, the
current value of the variable in the monitoring views can be checked.

Basic debugging features:

F5	Start program
F9	Set/reset breakpoint
F8	Perform single step
F10	Block-wise single step
STRG+F7	Write prepared value in variables
F7	Force prepared value into variables
ALT+F7	Cancel forcing

> ## i NOTE
>
> The following examples are from an arbitrary application and this to illustrate some debug functionalities. The illustrated procedure can thus be used in any application.

Write variables

For example, you can write or force a **"prepared value"** for the **"xPowerOn"** and **"uiStep"** variables on the controller, which means that these variables will receive the value on the controller at the beginning of the next cycle and will be retained during forcing.

To do this, select the field in the **"Prepared value"** column in **"uiStep"** in the declaration section, double-click to open the input field, enter a value of 20 and close it with the Enter key or a mouse click outside the field. For variables with numeric values or strings, a new value must always be entered in the input field.
The situation is different for the data type **"Bool"**. Click here into the input field and the **"prepared value"** takes the equivalent of the current value.

Expression	Type	Value	Prepared value
xPowerOn	BOOL	FALSE	
uiStep	UINT	10	

Expression	Type	Value	Prepared value
xPowerOn	BOOL	FALSE	TRUE
uiStep	UINT	10	20

Now execute the command **"Write Values"** with a click under **"Debug -> Write Values"** or by pressing **"CTRL"** and **"F7"**. You can see the newly written result in column **"Value"**.

Expression	Type	Value	Prepared value
xPowerOn	BOOL	TRUE	
uiStep	UINT	20	

"Inline monitoring" in the program code shows, in addition to the table form of the declaration section, that **"xPowerOn"** and **"uiStep"** have been written as desired. In addition, the program was operated in this way without visualization. The process control shows that the condition is fulfilled and that the corresponding code is executed.

```
IF xPowerOn TRUE = TRUE THEN     // When Rockerswitch pushed -> xPowerOn:= TRUE
    xGreen TRUE := TRUE;         //green lamp On
    gxOutGreen TRUE := TRUE; //Set output QX0.0

    //Blink Algorithm
IF xStart FALSE = TRUE THEN     // When Pushswitch pushed -> xStart:= TRUE
    CASE uiStep 20 OF //Case Instruction jumps from state 10 to 20 and back
```

96

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
01-0099-82-00-00-000 | Berghof_Control_Systemmanual_1.5_EN.docx

Now do the same with the variable **"xStart"**. In the declaration section, click in the column **"prepared value"** so that **"TRUE"** appears. Write the new value. Then you should see in the code how the inline monitoring value has changed from **"xStart"** to **"TRUE"**. Due to the process control, the change should be visible every second between steps 10 and 20.

```
//Blink Algorithm
IF xStart TRUE = TRUE THEN    // When Pushswitch pushed -> xS
        CASE uiStep  20  OF //Case Instruction jumps from sta
            10:           //State 10 -> Red is on and Timer r
                xRed FALSE := TRUE;              // re
                xYellow TRUE :=FALSE;            // ye
                gxOutYellow FALSE := TRUE;       // Se
                gxOutRed TRUE := FALSE;          //
                tmrBlink(IN TRUE :=TRUE, PT      T#1s
                IF tmrBlink.Q FALSE = TRUE THEN         // Wh
                    tmrBlink(IN TRUE := FALSE);         //
                    uiStep  20  := 20;            // Ju
                    gdwCnt    4    := gdwCnt    4    +1;
                END_IF
            20:           //State 20 -> Yellow is on and Timer
                xRed FALSE := FALSE;             // re
                xYellow TRUE :=TRUE;             // ye
                gxOutYellow FALSE := FALSE;      // Re
                gxOutRed TRUE := TRUE;           //
                tmrBlink(IN TRUE :=TRUE, PT      T#1s
                IF tmrBlink.Q FALSE = TRUE THEN         // Wh
                    tmrBlink(IN TRUE := FALSE);         //
                    uiStep  20  := 10;            // Ju
                    gdwCnt    4    := gdwCnt    4    +1;
                END_IF
```

So you can see that you can correctly control the program via the online mode and the arbitrary variable writing even without the visualization. These options are especially helpful in troubleshooting.

Set breakpoint

To set the breakpoint, perform a reset cold by executing **"Online -> Reset"**. Breakpoints can also be set in the current program, the reset in this example is for simplicity.

As long as the application is in the stop state, click on the line **"IF xStart = TRUE THEN"**, so that the pointer is in this line. If the pointer is in the line, press the **"F9"** key. The line is now completely highlighted in red and a red dot appears in the line at the left edge. The breakpoint has been set successfully.

```
IF xPowerOn FALSE = TRUE THEN     // When F
    xGreen FALSE := TRUE;         //green l
    gxOutGreen FALSE := TRUE; //Set output

    //Blink Algorithm
IF xStart FALSE = TRUE THEN       // Wher
        CASE uiStep   0  OF //Case I:
            10:              //State 10
                xRed FALSE := TRUE;
```

Activate the flow control and start the application. You can see with the help of process control that the process reaches the query of **"xPowerOn"**. Also, the program does not reach the breakpoint.

```
IF xPowerOn FALSE = TRUE THEN     // When Rockerswit
    xGreen FALSE := TRUE;         //green Lamp On
    gxOutGreen FALSE := TRUE; //Set output QX0.0

    //Blink Algorithm
IF xStart FALSE = TRUE THEN       // When Pushswi
        CASE uiStep  10  OF //Case Instructio:
            10:              //State 10 -> Red i:
                xRed FALSE := TRUE;
```

To reach the breakpoint, you have to set **"xPowerOn := TRUE"**. If the variable is set, the breakpoint can be reached. Upon reaching the breakpoint, the line is highlighted in yellow, the red dot on the left edge has a yellow arrow in its interior and the program is stopped. You also set **"xStart := TRUE"** so that you can step through the case instructions step by step.

```
IF xPowerOn FALSE = TRUE THEN     // When Rocl
    xGreen TRUE := TRUE;          //green Lamp
    gxOutGreen TRUE := TRUE; //Set output Q.

    //Blink Algorithm
IF xStart FALSE = TRUE THEN       // When Pl
        CASE uiStep  10  OF //Case Inst.
            10:              //State 10 ->
                xRed FALSE := TRUE;
```
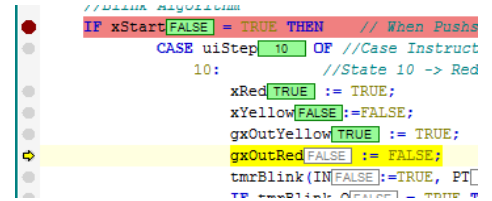
Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

97

Now, by pressing **"F8"** again, which corresponds to the command **"Debug -> Single step"**, processing can be done step by step. In this, the current line being processed is always highlighted in yellow, the yellow arrow on the left edge always moves with it. It should be noted, however, that when executing functions or function block instances with a single step (**"F8"**), they are jumped into. The editor window of the function or the function block instance opens and this is also processed step by step.

If you want to skip the function run, you can use **"F10"** instead of **"F8"**, which corresponds to the command **"Debug -> Procedure step"**. This will skip functions optically at step, but the code will be executed.

With **"F5"**, the program is restarted from the current step and continues automatically until the next breakpoint is reached. This allows you to go through complete individual cycles. It should be noted that the breakpoint positions are stored even when logged out of the controller. The next time you log in, they will be displayed as faint red markers and can be reactivated.

To remove a breakpoint, click with the mouse in the line in which the breakpoint was set. With the **"F9"** key, the breakpoint is removed and the line no longer has a colored background. With the **"F5"** key, the program is started again and continues normally as long as there are no further breakpoints in the program.

6.17. Project archive

Sharing a project file with a third party may be problematic under certain circumstances. This is because CODESYS V3 links devices, libraries, visu styles, etc. only to the system repository in a project. All these CODESYS V3 additions are not part of the project file.

In order for the transfer of a project file to work smoothly, all systems must have an identical repository, otherwise the project will open with missing objects and then report errors when the project is compiled. To prevent this problem, CODESYS V3 gives you the option to create a project archive.

A project archive allows you to extend a conventional project file with additions, it is thus possible to include embedded libraries, devices, etc. in the project file. When you open the project archive, the additions stored in the archive are automatically installed in the repository so that the project can then be opened without errors.

Via *File -> Project archive -> Save/send archive (in German: Datei -> Projektarchiv -> Archiv speichern)* a project archive can be created with the desired options.
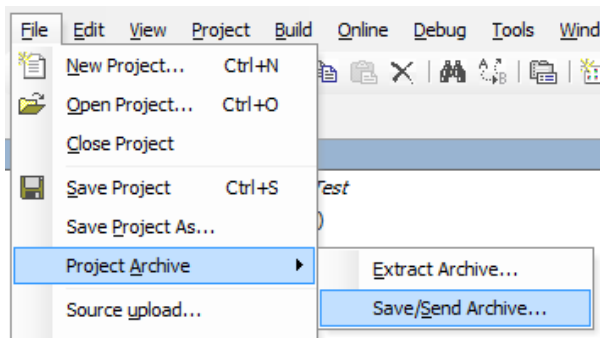
Picture 54: Project archive 1.1

A new window opens in which you can now select which additional objects are to be loaded into the project archive. For an error-free translation of the project, after opening the archive, it is recommended to select at least all referenced objects.

> **i NOTE**
>
> Libraries that are unprotected, i.e. not available as "compiled-library", are not automatically included in the project archive by CODESYS V3, for reasons of know-how protection. If you explicitly select such a library to be included in the list of information, you will receive a corresponding warning.
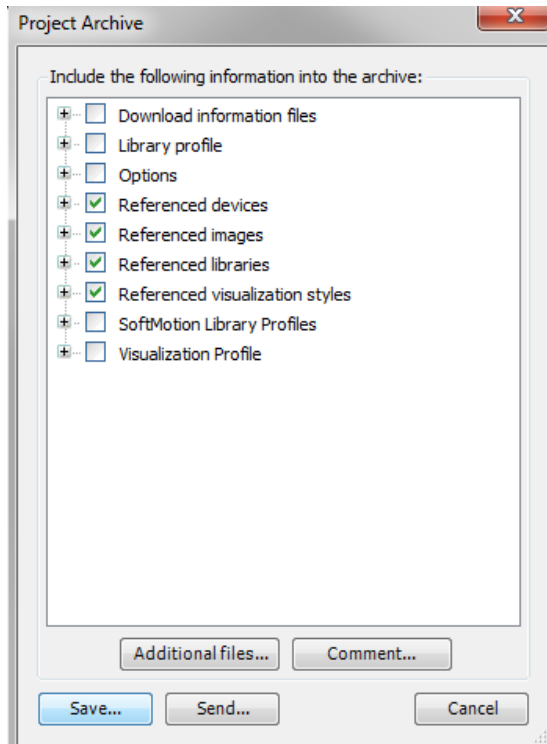
Picture 55: Project archive 1.2

Otherwise, it is up to the user to decide if he wants to include all objects or only a part. It is also possible to attach non-project files such as PDF through the button "additional files". Once the selection has been completed, the project archive is saved to the desired storage location by clicking on the "Save..." button.

6.18. Upgrade / downgrade a CODESYS V3 project

Upgrading a CODESYS V3 project to a new combination of CODESYS V3/Firmware/Target may be complex in some circumstances and may lead to new errors or problems. If you perform such an upgrade, the entire system must be re-tested in any case. Even with updates of the patch level, incompatibilities may occur, so that even such an update may not be used without tests in a productive environment; but as a rule, no problems are caused by such minor updates.

6.18.1. Example: Upgrade

You have a CC-Slim controller with firmware version 1.10.4. You can see from the table in chapter 5.2 that the suitable version is CODESYS V3.5 SP8 Patch 4. Now you want to upgrade to a new Berghof release, e.g. the release 1.13.0 based on SP9 patch 4. In this case, you must first update the controller to the SP9-compatible firmware 1.13.0. Afterwards, you can install CODESYS V3.5 SP9 Patch 4 and now have to install the target intended for the firmware (1.13.0). Now you can open your old project with CODESYS V3 SP9 Patch 4, but it is strongly recommended to create a backup of the project file first.
Wait until the project has finished loading and it will automatically open a window with the name "Project environment"; this window can also be opened manually under the context menu -> Project -> Project environment. The project environment checks the entire project when it is called and matches the versions

of the CODESYS V3 components of the project with the versions installed in the repository. If older components were found in the project, they will be displayed in the project environment with the suggestion to update to a new version.

The project environment typically recognizes the compiler version, visualization profile version, visualization style version, visualization symbol version, library versions not integrated by the placeholder, and the SoftMotion version.
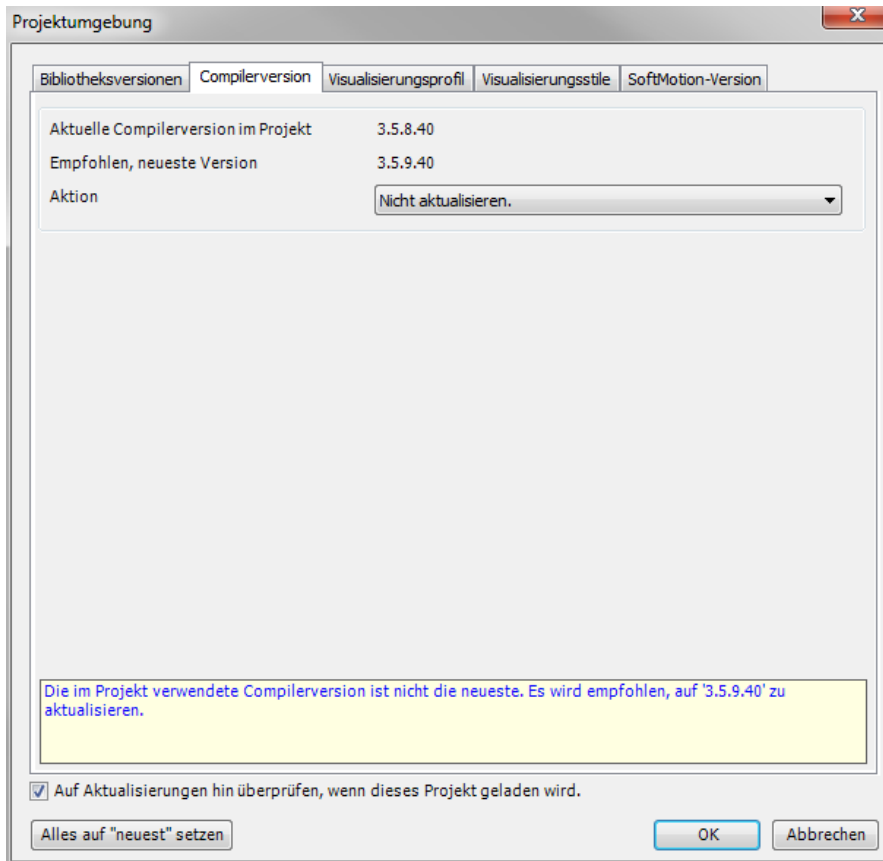
Picture 56: Project environment

As a rule, you can update the entire project to the latest version by clicking on the "Set all to newest" button. However, if many different CODESYS V3 versions are installed on the PC, it is recommended to go through all the tabs in the project environment and to set the versions manually exactly as if you want to update only parts of a project.

The versions displayed in the project environment can also be changed later in the project settings under the Context menu -> Project -> Project Settings.

Updating the project with the project environment was now the first step, as the project environment does not update any devices inserted in the project; they must be updated manually afterwards. Devices means hardware integrated in the CODESYS V3 in the control configuration and CODESYS V3 fieldbus modules, such as the controller itself, the EtherCAT™ master, the EtherNet interface with attached ModBusTCP, etc. After installing a new target version, the Berghof device must always be updated; in case of CODESYS V3 fieldbus modules, new versions of CODESYS V3 are not always new versions of all devices here. This must be checked and updated manually. The device version currently integrated in the project can be found by opening the device editor by double-clicking on the device itself; the version can be found in the "Information" tab.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

101
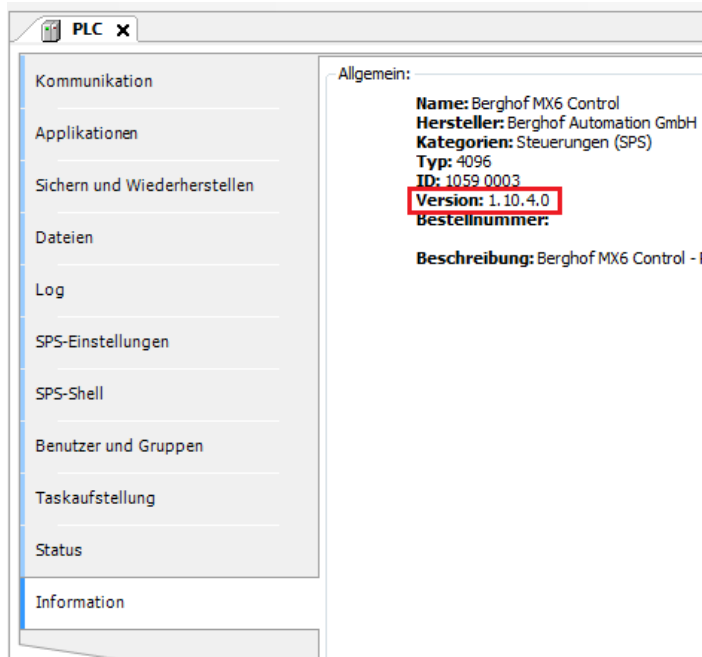
Picture 57: PLC

If the integrated version is known, you can now check if a newer version is available in the CODESYS system.
To do this, right-click on the device and select "Update device".
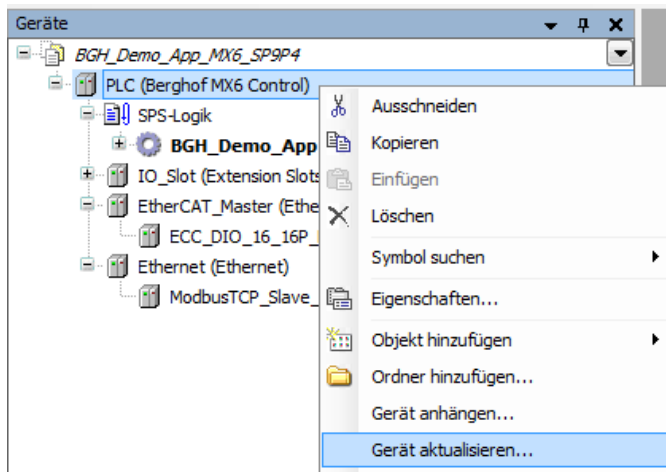
Picture 58: devices

A new window opens in which a device list is displayed, the marking automatically jumps to the device with which you have called the update process. With the default settings, it shows only the latest version of the device; optionally you can check the box in the setting "Show all versions" to show the full overview of all installed versions of this device type.
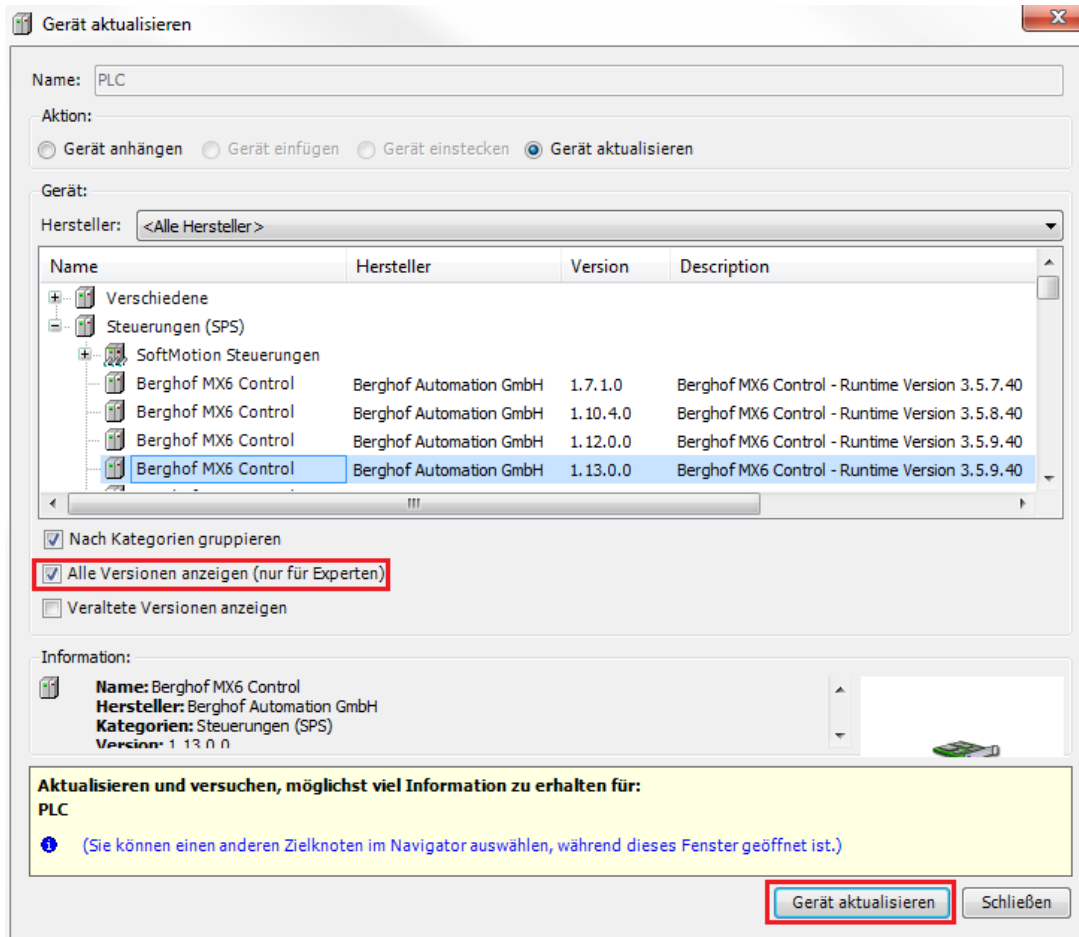
Picture 59: Update devices

To update the device, mark the desired version and press the button "Update device".
In the information tab of the device editor, you can check if the correct version is now attached.
This process is now repeated for all devices connected in the project.
The step sequence is always the same, you just have to be careful to mark the right device. Settings in the editors will be retained if the device is the same.
As an example, an update of the EtherCAT™ Master is shown.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

103
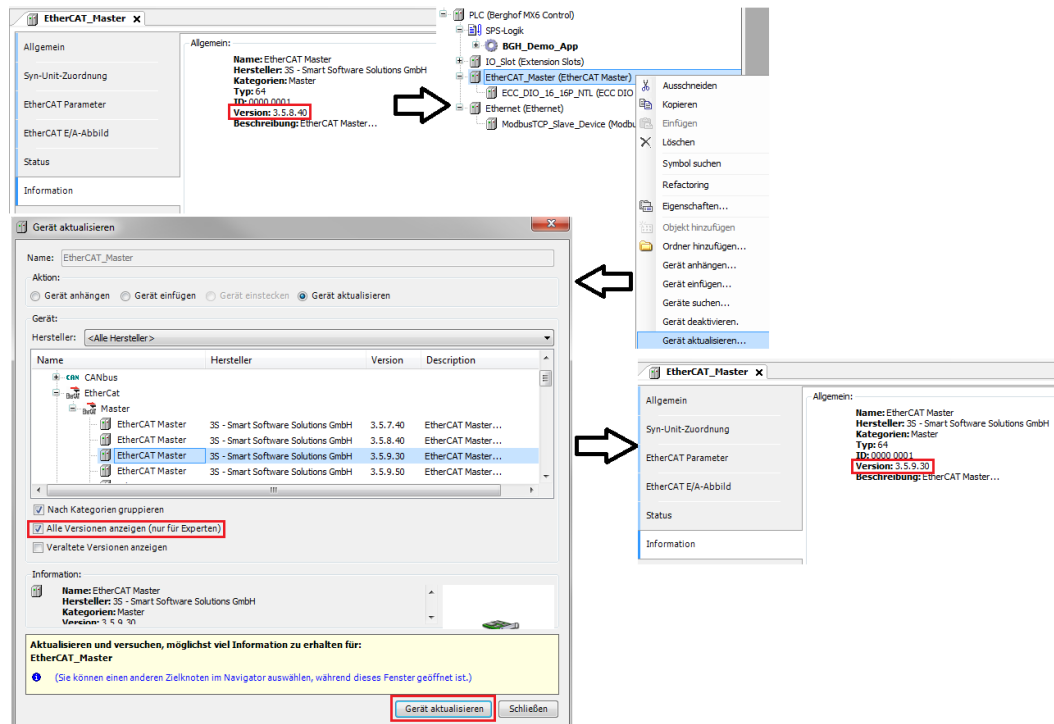
Picture 60: Save project

If the first part of the project has been updated automatically and the devices manually via the project environment and thus the second part of the project, then the project must be cleaned up once under the Context menu -> Create -> Clean up everything and recompile without errors under the Context menu -> Create - > Generate Code; the project can now be saved and loaded onto the controller with the updated firmware.

6.18.2. Example: Downgrade

It is also possible to downgrade only one project from a newer version of CODESYS V3 to an older one. Prerequisite is that the older version on the PC with which the downgrade is performed.
In principle, it is the same as with a project upgrade, but you cannot use the project environment here. Start in the project settings and first set the compiler, Visu profile and possibly SoftMotionversion to the desired older version and confirm the choice with the "OK" button. Then, for all project devices, such as during the upgrade, the device list must be opened for updating, all versions must be displayed and the appropriate device must be selected in the older version. If project settings and devices are adjusted, open the library manager and check if all libraries inserted by the user (not grayed out) are included in the correct version; if necessary, also include the correct versions here. Placeholders or libraries implicitly linked by the system automatically convert to the correct version when the project settings and devices are rearranged. If the project is internally adapted to the older version, the project file must still be finally saved in the correct project profile. To do this, save the project under File -> Save as. In the save dialog, you can now select the appropriate profile version under File type. If the desired profile version is not in the list, select the next older profile version.

7. Best practices Berghof and CODESYS V3

7.1. Berghof Software options

The functionality of the Berghof control system can be easily extended with software options.
New features can be easily activated by purchasing licenses for the controller. With a new order, it is possible to get a controller with preinstalled licenses, as well as to install any license later on the controller.
It is possible to combine licenses when ordering, according to your wishes. If licenses are subsequently installed, the customer will always receive a license file containing all software options ordered. This file is conveniently installed via the update functionality in the web interface on the respective controller. The unlockable software options differentiate between options in the firmware and options for CODESYS V3; the individual options are shown in the following chapters.

---

i   NOTE

If a license is ordered to be subsequently installed on a controller, please
always send us a screenshot of the System Info page from the web interface of the controller on which the license is to be installed. Through the System Info page, it is possible to see all the information which is important for the correct licensing at a glance.

7.1.1.   Software options for CODESYS V3 Features

The software options in CODESYS V3 are software features integrated in CODESYS V3. These features are mainly different fieldbus protocols or special features like Softmotion and CNC functionality.

---

i   NOTE

If you include one of these options in your project without having a license installed on your controller, this is not a problem. When starting the application, the embedded feature starts in demo mode. In demo mode, the feature runs in full functionality for a predefined time (depending on the feature 30-120min); when the time is over, the feature automatically disables. If you restart the controller, the demo mode is restarted. Thus, a license does not have to be purchased to test and evaluate a CODESYS V3 software option. An exception is Softmotion; here a separate demo license must be installed on the controller; this demo license is available for free from Berghof.

The following features are included with every Berghof d control system.

TargetVisu	S104 Ether-CAT™ Master	S103 CANOpen Manager
✓	✓	✓*

* When the device has a CAN interface

The following software options can be installed via a paid license on any compatible Berghof control system.

S100 ModBus TCP/RTU	S101 BACnet	S102 J1939	S105 Profinet™ Device	S106 OPCuA Server
O	O	O	O	O

S107 EtherNet IP Scanner	S108 EtherNet IP Adapter	S110 IIOT Libraries	S114 WebVisu	S115 Profinet Controller	S116 KNX IP
O	O	O	O	O	O

S117 OPCuA Client	S118 OPCuA PubSub	S300 Softmotion	S301 Softmotion CNC+Robotics
O	O	O	O

Further information on the individual software options can be found in the respective product data sheets. The product data sheets can be found in the customer download area of the Berghof Automation homepage.

7.1.2. Software options for Firmware Features

The software options in firmware are firmware features that extend the functionality of the controller and, in principle, are independent of CODESYS V3 for your use.

---

i NOTE

For software options in the firmware there is no demo mode and no demo licenses; these must always be purchased. If interested, please contact the Sales Department.

The following firmware features are included with every Berghof control system.

S002 USB Serial	S003 USB Ethernet	S004 USB Keyboard	S111 FTP Server	S112 NTP Client
✔	✔	✔	✔	✔

S501 Backup Restore	S502 Display Rotation	S503 Onboard config. Protection		
✔	✔	✔		

The following software options can be installed via a paid license on any compatible Berghof control system.

S000 Counter Encoder	S001 Hardware Watchdog	S006 Fast Event IO	S109 VPN / IXON	S306 Berghof Motion Basic
O	O	O*	O	O

S400 Powertrack Communication	S401 Powertrack Service Tool	S402 Powertrack Software	S500 PDF Creation Library	S505 Application Protection
O	O	O	O	O

* Does not work with CC-Slim, DC-Pro 4/7

Further information on the individual software options can be found in the respective product data sheets. The product data sheets can be found in the customer download area of the Berghof Automation homepage.

7.2. Cyber Security

To connect Berghof controller to the Internet and to use remote access options the following procedures are recommended.

- Our controllers must not be connected directly to the Internet without external protection through suitable firewalls.
- Change the default passwords given at delivery. .
- Use the security mechanisms of VPN server portals (e.g. IXON) to which the controller can actively connect via VPN or comparable mechanisms
- Use https instead of http.
- Disable the services you do not need, such as (FTP/SSH/...).

Contact person for cybersecurity of Berghof products:
Berghof Automation Product Security Incident Response Team
psirt.automation@berghof.com

7.3. Multicore Quickstart

Berghof provides several product families that come equipped with either a dual or quad core CPUs. These controllers allow you to create more complex applications with fast visu responses while maintaining low and stable cycle times.

7.3.1. Requirements

- Install Multicore compatible CODESYS Development System (3.5.16.xx or newer)
- Install Multicore compatible Target Package
    1.24.x or newer for MX6 based Duo devices
    1.1.x or newer for Rasberry Pi based devices
- Check that your controller firmware is matching with the installed target package

7.3.2. Lock EtherCAT interface to Core

With a Multicore controller it is possible to lock the EtherCAT port to a CPU core (recommended for Soft-Motion) to enable, load the network page in the Webinterface.
By default, "eth1" is set to ethercat. The "ethercat" settings automatically locks the selected interface to the last CPU core available on the Multicore controller. Which is Core1 on a Dualcore and Core3 on Quadcore PLC.
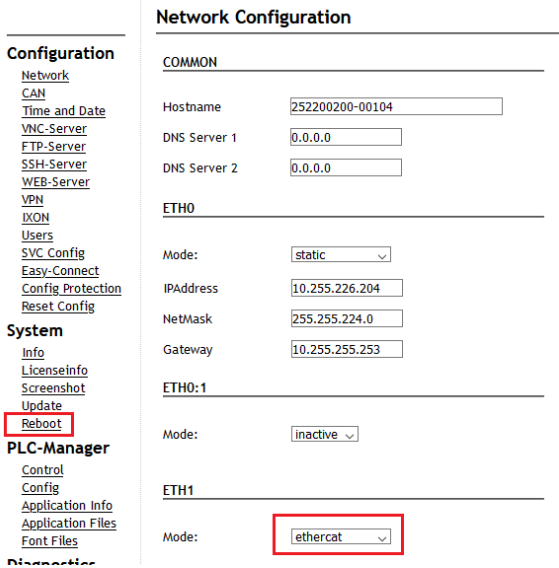
Picture 61: EtherCAT settings

If it is not required that the EtherCAT is locked to an isolated core select an unconfigured mode like "dhcp" or "profinet device". CODESYS will do the correct configuration of the interface on startup.
Be aware that by not selecting "ethercat" mode, the real time behavior of EhterCAT on interface might differ. If EtherCAT is not required at all select the required mode.
After the settings are done, save the configuration and reboot the PLC.

7.3.3. New project (optional)

Open CODESYS 3.5.16.xx or newer and create a Standard Project and Select a Berghof MX6 or MC-Pi device Description (with or without Softmotion)
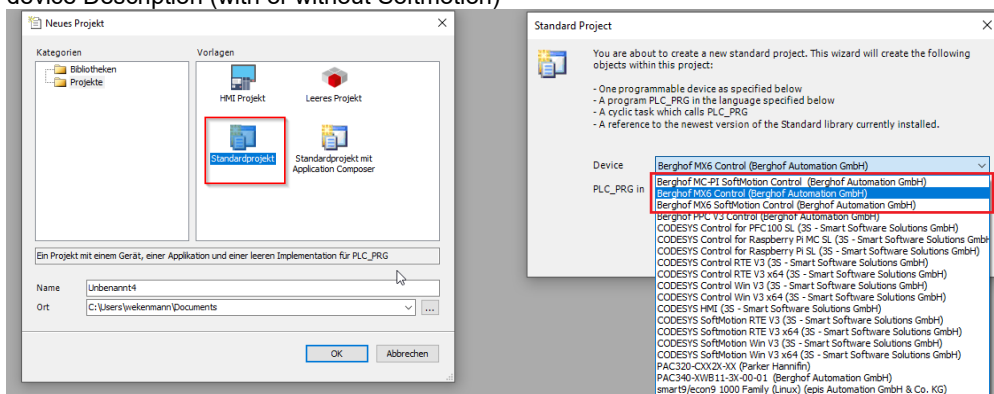
Picture 62: Create new project and select PLC

The newest installed device will be used automatically, and the project will have the matching project settings.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

109

7.3.4. Open and upgrade existing project

Open an already existing project with CODESYS 3.5.16.xx or newer and update the PLC device to a Multicore compatible version.
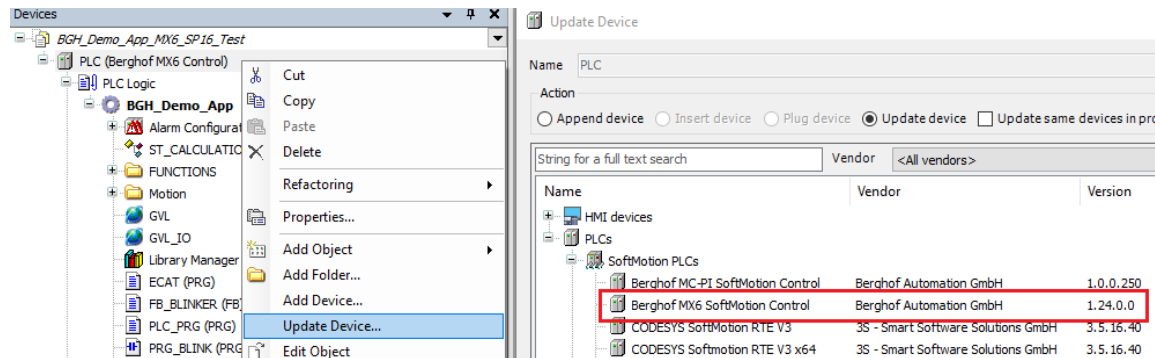
Picture 63: Update PLC device

If the existing project is based on an older CODESYS Version, please upgrade the rest of the project properly to the installed CODESYS Version. Besides the PLC device please check the sub devices like the EtherCAT Master, CANOpen Manager etc, the configured Compiler and Visu Profile Version in the project settings.
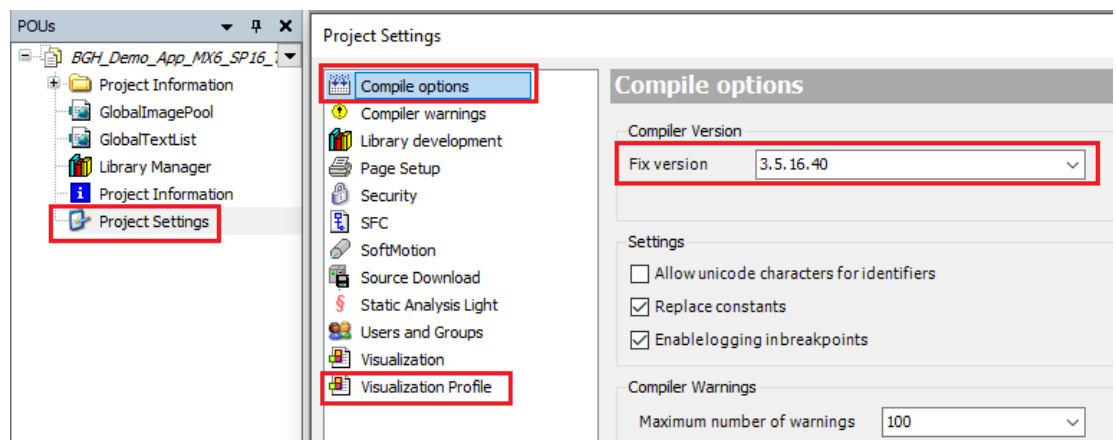
Picture 64: Project settings

As well as manually added libraries in the library manager.

7.3.5. Open and upgrade existing project

Under „Task Configuration-> Task Groups" you can add additional task groups and select the scheduling algorithm for the task group. Then tasks can be added to a task group.
When using EtherCAT that is locked to a core, create a new Task group and add the EtherCAT task to it.
Other Tasks that are of high priority or linked to the EtherCAT task may be added as well.
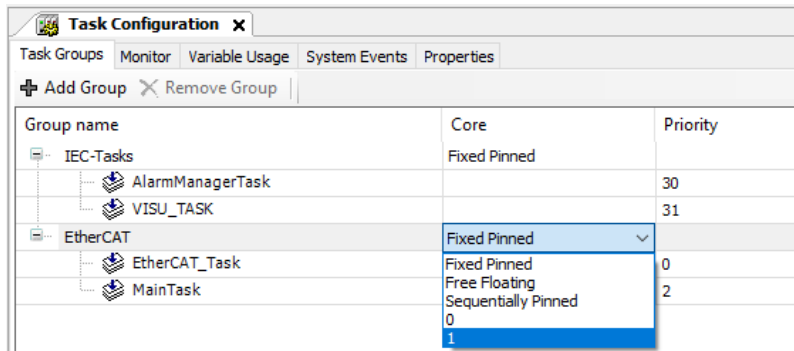
Picture 65: Task configuration create task group

Select the last Core (Core 1 or Core 3) on the EtherCAT Task group and make sure the EtherCAT Task has the priority 0. It is possible to create multiple task groups, but it is not recommended to create more task groups than actual cores.
On a Dualcore it is enough to select Core 0 for the other Task group if no high priority tasks are used in that group. On a Quadcore PLC the configuration is more flexible. But for an easy to use start we recommend having one more task group besides the EtherCAT task group and set it to "Free Floating".
If there is no core locked to EtherCAT the best practice is to go with one task group and select "Free Floating".
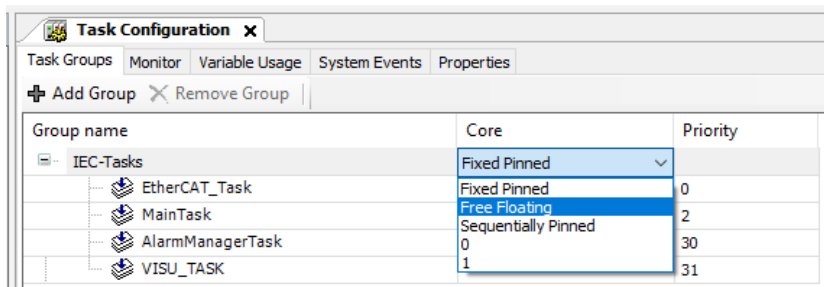
Picture 66: Task configuration assign core to task group

In "Free Floating" mode the system is responsible for scheduling the tasks over the two or four cores. More information about the multicore features and the different scheduling modes can be found in the CODESYS help: https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_obj_task_config_task_groups.html

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

111

In case of manually assigning the cores on either Dualcore or Quadcore PLCs it is recommended to only pin low priority tasks (Prio 16 or higher) to Core 0 and to use the remaining Cores 1-3 to pin high priority tasks (15 or lower) to, as Core0 is additionally running the OS and the Runtime.

Be aware that Multicore systems perform real multitasking on separate cores, and thus race conditions can occur on unoptimized applications that were previously running on SingleCore systems.
Please make sure that different task groups are not writing the same variables or lock parallel writing with the help of state machines and semaphores.

7.4. Intall updates on the controller

As mentioned in chapter 4.2.2. the controller has an integrated update function in the webinterface.
This chapter shows a step by step how to update the firmware as an example.
However, all update files (firmware/licenses/application) are installed with the same steps

Connect to the webinterface of your controller using a compatible browser.
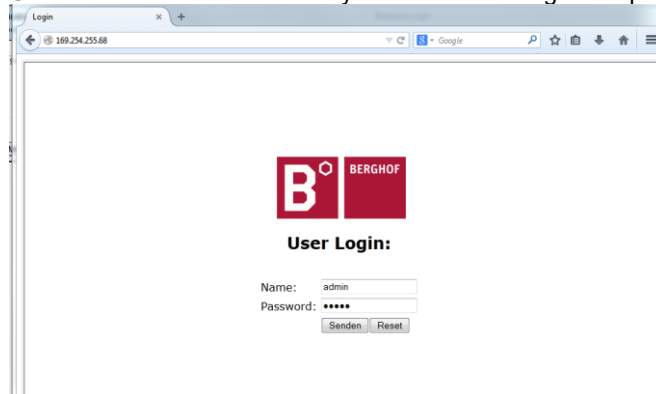
If the controller is running an application, go to "PLC-Manager->Control" and stop the application.
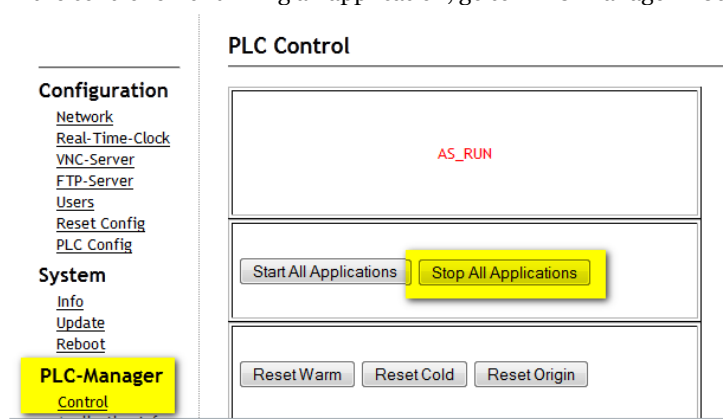
When the controller is stopped navigate to "System->Update" and click the button "browse" (may vary depending on the local language of the PC). Search for your firmware file and open it.
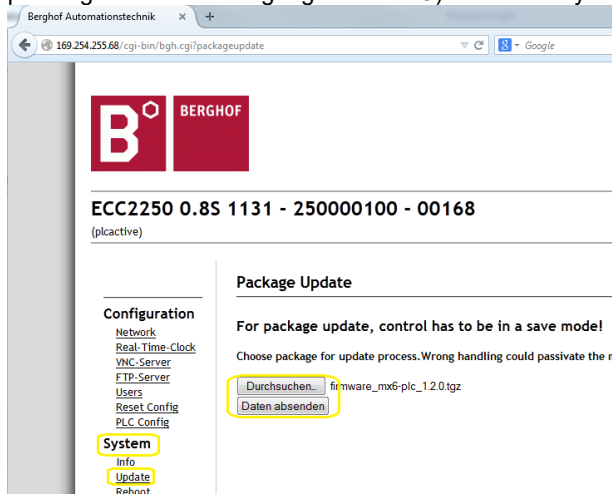
The selected file is shown in the browser and by clicking "Send" it will be loaded onto the controller.
After the file is fully loaded onto the controller the package information, which is a brief content overview will be shown.
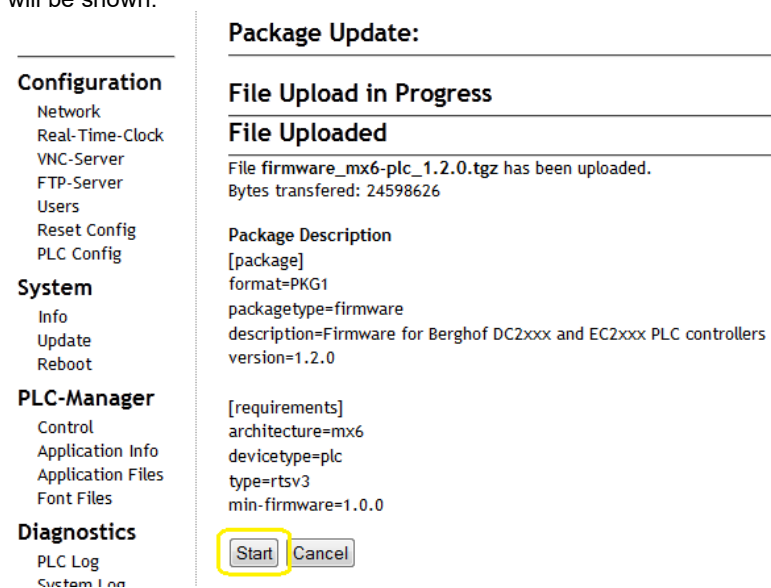
With a click on the "Start" button, the update process is now started. The process is fully automated and cannot be interrupted. The webinterface will show a progress log. Wait until the log shows "Update SUCCEEDED" and the prompt to reboot the controller.

---
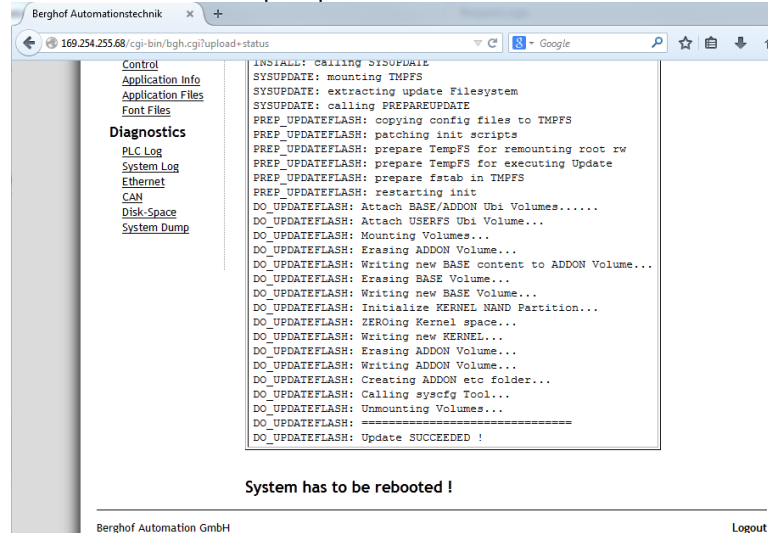
i NOTE

Due to the high CPU load that is caused by an update, it might happen that the browser will have a connection timeout. In that case do not restart the controller, wait a few seconds, and refresh the browser and the process log will reappear.

For the update to take effect, navigate to "System->Reboot" and restart the controller with clicking the button "Reboot Module"
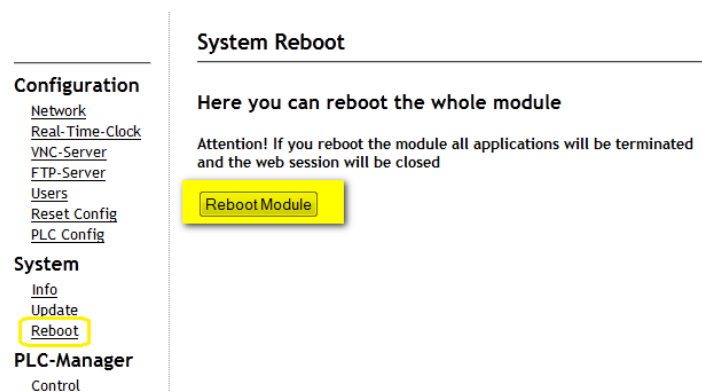
⚠ CAUTION

When performing an update make sure that the controller has a stable power source. Never take power away or restart the controller while updating as it could damage the software system and prevent the bootup. If for whatever reason the update state is unclear, wait at least 5 minutes before restarting the controller.

7.5. Berghof System Library

The Berghof System Library is an integral part of the Berghof Target Pro Package and includes functionality to configure the controller from within the application and perform hardware diagnostics. The Berghof System Library is self-contained; for individual functions DUTs such as structures or Enums are used; these are included in the library.

> **i NOTE**
>
> The following list of functions is based on version 1.10.0.0 of the Berghof System Library MX6.

7.5.1. Read boot cause (FUN DGN_GetBootReason)

Description: Reads the cause of the last boot.

Parameter	Value	Description	
Output parameters: DGN_GetBootReason	0..3	0:	Unknown reason
		1:	Device has been switched on
		2:	Device had a soft reboot from the application or by the user
		3:	Device had a Watchdog Reset

Graphical display:
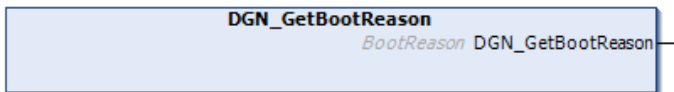
Sample call in ST:

enReason := DGN_GetBootReason();

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

115

7.5.2. Read system temperature (FUN DGN_GetAmbientTemperature)

Description: Reads the current system temperature values

	Parameter	Value	Description
Input parameters:	pTempValues	-	Pointer to the Temperature Structure Temperatures are written in here
Output parameters:	DGN_GetAmbientTemperature 0..1	0:	Successfully executed 1: Error occurred

Graphical display:
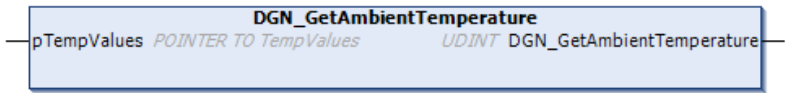
Sample call in ST:

udiReturn := DGN_GetAmbientTemperature(pTempValues:=ADR(stTemperatureValues));

7.5.3. Read CPU temperature (FUN DGN_GetDieTemperature)

Description: Reads the current CPU temperature values

	Parameter	Value	Description
Input parameters:	pTempValues	-	Pointer to the Temperature Structure Temperatures are written in here
Output parameters:	DGN_GetDieTemperature	0..1	0: Successfully executed 1: Error occurred

Graphical display:
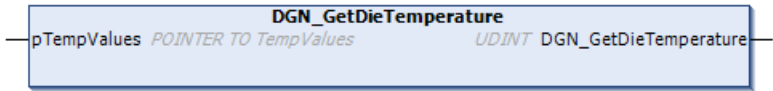
Sample call in ST:

udiReturn := DGN_GetDieTemperature(pTempValues:=ADR(stTemperatureValues));

7.5.4. Read operating hours (FUN DGN_GetOperationHours)

Description: Reads the operating hours of the controller

Parameter	Value	Description
Output parameters: DGN_GetOperationHours	-1..x -1:	Error occurred
	x:	Device has been switched on

Graphical display:

DGN_GetOperationHours
DINT DGN_GetOperationHours
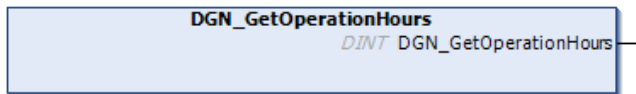
Sample call in ST:

diHours := DGN_GetOperationHours ();

7.5.5. Read Retain Status (FUN DGN_GetRetainStatus)

Description: Reads the retain memory status of the controller

Parameter	Value	Description
Output parameters: DGN_GetRetainStatus 0..2	0:	UNDEFINED - Retain Status not defined
	1:	OK - Retain Status OK
	2:	OLDDATA – Retain data is available but could be obsolete

Graphical display:

DGN_GetRetainStatus
RetainStatus DGN_GetRetainStatus
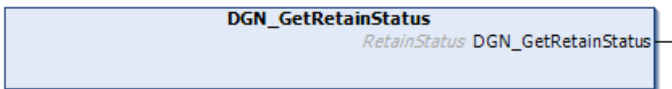
Sample call in ST:

enRetainStatus := DGN_GetRetainStatus ();

7.5.6. Apply settings (FUN CNF_ApplySettings)

Description: If settings are changed via CNF_Set or CNF_Save functions, these settings are saved temporarily. Only with the execution of the Apply Settings function, the settings are saved. After settings have been saved with Apply Settings, the controller must be restarted for the settings to take effect.

	Parameter	Value	Description	
Output parameters:	CNF_ApplySettings	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:
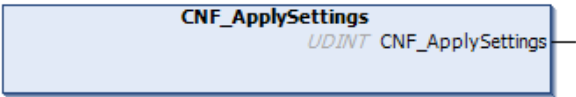
Sample call in ST:

udiReturn := CNF_ApplySettings ();

7.5.7. Read screen brightness (FUN SCN_GetBrightness)

Description: Reads the current brightness value of the controller.
Works only on controllers with display.

	Parameter	Value	Description	
Input parameters:	pbyBrightness	-	Pointer to the brightness variable	
			Brightness value is written in here	
Output parameters:	SCN_GetBrightness	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:
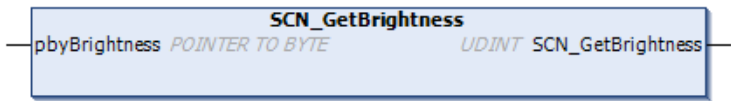
Sample call in ST:

udiReturn := SCN_GetBrightness(pbyBrightness :=ADR(byBrightness));

7.5.8. Set screen brightness on start (FUN SCN_SetBrightness)

Description: Writes the desired brightness value in the controller.
                     Works only on controllers with display.

	Parameter	Value	Description
Input parameters:	byBrightness	0..8	Brightness value for the screen
			0: Off
			8: Maximum brightness
Output parameters:	SCN_SetBrightness	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
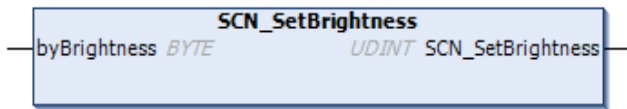
Sample call in ST:

udiReturn := SCN_SetBrightness(byBrightness:=bySetBrightness);

7.5.9. Start set screen brightness (FUN CNF_SaveScreenBrightness)

Description: Writes the desired brightness value temporarily in the controller as start value
                     Parameters must be permanently saved with CNF_ApplySettings.
                     The controller will then have this brightness value at each bootup.
                     Works only in controllers with display.

	Parameter	Value	Description
Input parameters: screen	byBrightness	0..8	Brightness value for the
			0: Off
			8: Maximum brightness
Output parameters:	SCN_SaveScreenBrightness	0..1	0: Successfully executed
			1: Error occurred

Graphical display:

Sample call in ST:

udiReturn := CNF_SaveScreenBrightness (byBrightness:=byStartBrightness);

7.5.10. Read screen rotation (FUN CNF_GetScreenRotation)

Description: Reads out the set degree of rotation of the display from the controller.
Works only in controllers with display.

	Parameter	Value	Description
Input parameters:	pwScreenRotation	-	Pointer to the degree variable Degree is written in here
Output parameters:	CNF_GetScreenRotation 0..1		0: Successfully executed
			1: Error occurred

Graphical display:
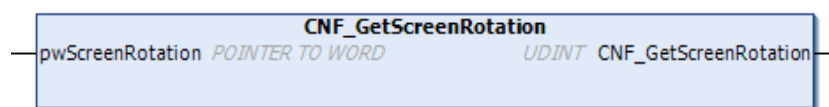
Sample call in ST:

udiReturn := CNF_GetScreenRotation (pwScreenRotation:=ADR(wSetRotation));

7.5.11. Set screen rotation (FUN CNF_SetScreenRotation)

Description: Allows you to rotate the displayed image by 0, 90, 180, 270 degrees counter clockwise.
Writes the desired degree temporarily in the controller.
Parameters must be permanently saved with CNF_ApplySettings.
Works only in controllers with display.

	Parameter	Value	Description	
Input parameters:	wScreenRotation	0,90,180,270	0:	Display remains unchanged
			90:	Display is rotated by 90°
			180:	Display is rotated by 180°
			270:	Display is rotated by 270°
Output parameters:	CNF_SetScreenRotation		0..1	Successfully executed
			1:	Error occurred

Graphical display:
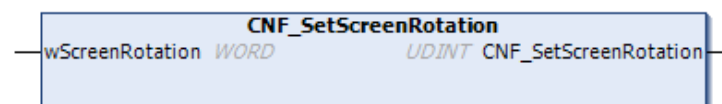
Sample call in ST:

udiReturn := CNF_SetScreenRotation (wScreenRotation:=wSetRotation);

7.5.12. Read out time zone (FUN CNF_GetTimezone)

Description: Reads the time zone set in the web interface from the controller.

	Parameter	Value	Description
Input parameters:	sTimezone	-	IN_OUT String to the time zone variable Time zone is rewritten to transferred variable
Output parameters:	CNF_GetTimezone	0..1	0: Successfully executed 1: Error occurred

Graphical display:
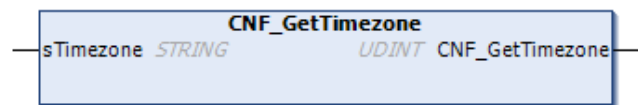
Sample call in ST:

udiReturn := CNF_GetTimezone (sTimezone:=sGetTimezone);

7.5.13. Set time zone (FUN CNF_SetTimezone)

Description: Writes the desired time zone temporarily into the controller.
                           Parameters must be permanently saved with CNF_ApplySettings.

	Parameter	Value	Description
Input parameters:	sTimezone	e.g. 'UTC'	String for time zone variable The correct description of the individual time zones can be found in the list of time zone selections in the Web Interface
Output parameters:	CNF_SetTimezone	0..10:	0: Successfully executed 1: Error occurred

Graphical display:
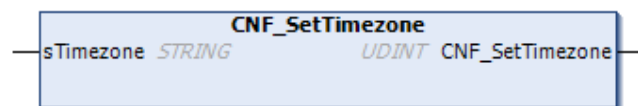
Sample call in ST:

udiReturn := CNF_SetTimezone (sTimezone:=sSetTimezone);

7.5.14. Read SCCAN Mode (FUN CNF_GetSccanPhyActive)

Description:

Reads out the status of the selected CAN interface and reports if SCCAN is active.
Only functions on the CCPU 8/8/4 MX6

Input parameters:	Parameter	Value	Description
	sInterface	'can0','can1'	String for desired CAN Interface

Output parameters:	Parameter	Value	Description	
	CNF_GetSccanPhyActive	-x..1	-x:	Error occurred
			0:	SCCAN inactive
			1:	SCCAN active

Graphical display:

```
              CNF_GetSccanPhyActive
─── sInterface STRING       INT  CNF_GetSccanPhyActive ───
```
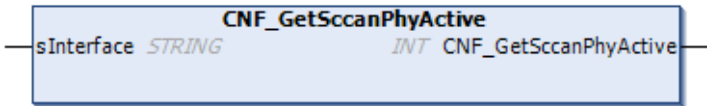
Sample call in ST:

iMode := CNF_GetSccanPhyActive (sInterface:=sCANInterface);

7.5.15. Set SCCAN Mode (FUN CNF_SetSccanPhyActive)

Description:

Reads out the status of the selected CAN interface and reports if SCCAN is active.
Parameters must be permanently saved with CNF_ApplySettings.
Only functions on the CCPU 8/8/4 MX6

Input parameters:	Parameter	Value	Description	
	sInterface	'can0','can1'	String for desired CAN Interface	
	xActiv	TRUE,FALSE	TRUE	sets SCCAN active
			FALSE:	sets SCCAN inactive

Output parameters:	Parameter	Value	Description	
	CNF_ SetSccanPhyActive	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:

```
              CNF_SetSccanPhyActive
─── sInterface STRING       INT  CNF_SetSccanPhyActive ───
─── xActiv BOOL
```
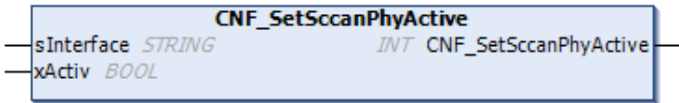
Sample call in ST:
udiReturn := CNF_SetSccanPhyActive (sInterface:=sCANInterface,xActiv:=xActivateSC);

7.5.16. Read Ethernet Mode (FUN CNF_GetEthMode)

Description: Reads out the set mode of the desired Ethernet interface.

	Parameter	Value			Description
Input parameters:	sInterface	'eth0',	'eth1'		String for desired Ethernet interface
	eMode	0..3			IN_OUT Enum for Ethernet Mode
				0:	INACTIVE – Ethernet interface Inactive
				1:	STATIC – Operate Ethernet interface with static IP
				2:	DHCP – Operate Ethernet interface dynamically with DHCP
				3:	ETHERCAT – Operate Ethernet interface in the EtherCAT Mode
Output parameters:	CNF_GetEthMode	0..1		0:	Successfully executed
				1:	Error occurred

Graphical display:
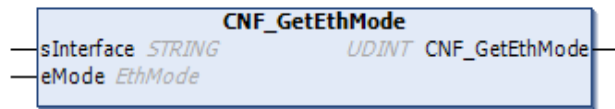
Sample call in ST:

udiReturn := CNF_GetEthMode (sInterface:=sEthInterface,eMode:=enEthMode);

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

123

7.5.17. Set Ethernet Mode (FUN CNF_SetEthMode)

Description:

Sets the set mode in the desired Ethernet interface.
Parameters must be permanently saved with CNF_ApplySettings.

	Parameter	Value	Description	
Input parameters:	sInterface	'eth0', 'eth1'	String for desired Ethernet interface	
	eMode	0..3	Enum for Ethernet Mode	
			0:	INACTIVE – Ethernet interface Inactive
			1:	STATIC – Operate Ethernet interface with static IP
			2:	DHCP – Operate Ethernet interface dynamically with DHCP
			3:	ETHERCAT – Operate Ethernet interface in the EtherCAT™ Mode
Output parameters:	CNF_SetEthMode	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:
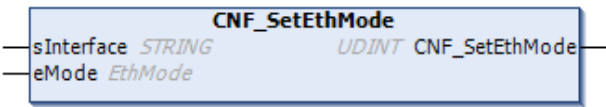
Sample call in ST:

udiReturn := CNF_SetEthMode (sInterface:=sEthInterface,eMode:=enEthMode);

7.5.18. Read IP address (FUN CNF_GetIpAddress)

Description: Reads the set IP address from the system configuration.
Reads only the static IP address, dynamically set IP addresses are not read.

	Parameter	Value	Description
Input parameters:	sInterface	'eth0', 'eth1'	String for desired Ethernet interface
	abyIpAddress	-	IN_OUT Array of Byte [0..3] Array of four byte fields, each field corresponds to a bit octal from the IP

	Parameter	Value	Description	
Output parameters:	CNF_GetIpAddress	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:
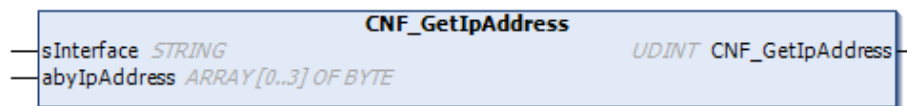
Sample call in ST:

udiReturn := CNF_GetIpAddress(sInterface:=sEthInterface,abyIpAddress:=abyIpGet);

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

125

7.5.19. Set IP address (FUN CNF_SetIpAddress)

Description: Sets the set IP address from the system configuration.
                             Only sets the static IP address, dynamically set IP addresses cannot be set
                             manually.
                             Parameters must be permanently saved with CNF_ApplySettings.

	Parameter	Value	Description
Input parameters:	sInterface	'eth0', 'eth1'	String for desired Ethernet interface
	abyIpAddress		Array of Byte [0..3] Array of four byte fields, each field corresponds to a bit octal from the IP.

Output parameters:	CNF_SetIpAddress	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
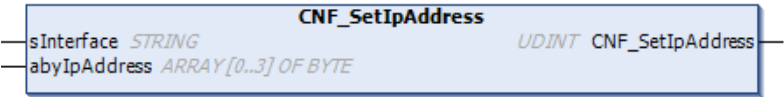
Sample call in ST:

udiReturn := CNF_SetIpAddress(sInterface:=sEthInterface,abyIpAddress:=abyIpSet);

7.5.20. Read Netmask (FUN CNF_GetNetMask)

Description: Reads the set Netmask from the system configuration.
Only reads the static Netmask, dynamically set Netmasks are not read.

	Parameter	Value	Description
Input parameters:	sInterface	'eth0', 'eth1'	String for desired Ethernet interface
	abyNetMask	-	IN_OUT Array of Byte [0..3] Array of four-byte fields, each field corresponds to a bit octal from the netmask.

	Parameter	Value	Description	
Output parameters:	CNF_GetNetMask	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:
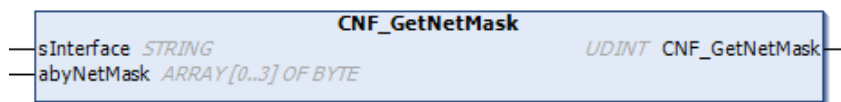
Sample call in ST:

udiReturn := CNF_GetNetMask(sInterface:=sEthInterface, abyNetMask:= abyNetMaskGet);

7.5.21. Set Netmask (FUN CNF_SetNetMask)

Description: Set the set Netmask in the system configuration.
Only sets the static Netmask, dynamically set Netmasks cannot be set manually.
Parameters must be permanently saved with CNF_ApplySettings.

	Parameter	Value	Description
Input parameters:	sInterface	'eth0', 'eth1'	String for desired Ethernet interface
	abyNetMask	-	Array of Byte [0..3] Array of four-byte fields, each field corresponds to a bit octal from the netmask.
Output parameters:	CNF_SetNetMask	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
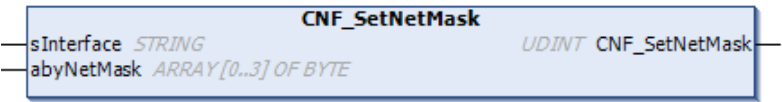
Sample call in ST:

```
udiReturn :=
CNF_SetNetMask(sInterface:=sEthInterface, abyNetMask:= abyNetMaskSet);
```

7.5.22. Read Gateway address (FUN CNF_GetGatewayAddress)

Description: Reads the default gateway address set for interface eth0 from the system configuration. If eth1 is transferred as an interface, eth0 is nevertheless read out,

as eth1 has no default gateway. Reads only the static gateway address, dynamically set gateway addresses are not read out.

Input parameters:

Parameter	Value	Description
sInterface	'eth0', 'eth1'	String for desired Ethernet interface
abyGatewayAddress	-	IN_OUT Array of Byte [0..3] Array of four-byte fields each field corresponds to a bit octal from the gateway address.

Output parameters:

CNF_GetGatewayAddress	0..1	0: Successfully executed
		1: Error occurred

Graphical display:

Sample call in ST:

udiReturn := CNF_GetGatewayAddress(sInterface:=sEthInterface,
abyGatewayAddress:=abyGatewayGet);

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

129

7.5.23. Set Gateway address (FUN CNF_SetGatewayAddress)

Description:

Sets the default gateway address set for interface eth0 in the system configuration. If eth1 is transferred as an interface, eth0 is nevertheless set, as eth1 has no default gateway. Only sets the static gateway address, dynamically set IP addresses cannot be set manually.
Parameters must be permanently saved with CNF_ApplySettings.

Input parameters:

Parameter	Value	Description
sInterface	' eth0', 'eth1'	String for desired Ethernet interface
abyGatewayAddress	-	Array of Byte [0..3] Array of four-byte fields, each field corresponds to a bit octal from the gateway address.

Output parameters:

CNF_SetGatewayAddress	0..1	0:	Successfully executed
		1:	Error occurred

Graphical display:

Sample call in ST:

udiReturn := CNF_SetGatewayAddress(sInterface:=sEthInterface,
abyGatewayAddress:=abyGatewaySet);

7.5.24. Read DNS address (FUN CNF_GetDnsAddress)

Description: Reads the set DNS Address from the system configuration.
Only reads the static DNS Address, dynamically set DNS Addresses are not read.

Input parameters:

Parameter	Value	Description
sInterface	'dns0', 'dns2'	String for desired DNS setting
abyDnsAddress	-	IN_OUT Array of Byte [0..3] Array of four byte fields, each field corresponds to a bit octal from the DNS Address.

Output parameters:

CNF_GetDnsAddress	0..1	0:	Successfully executed
		1:	Error occurred

Graphical display:
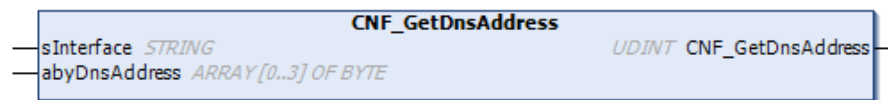
Sample call in ST:

udiReturn :=
CNF_GetDnsAddress(sInterface:=sEthInterface, abyDnsAddress:=abyDnsGet);

7.5.25. Set DNS address (FUN CNF_SetDnsAddress)

Description:

Sets DNS Address in the system configuration.
Only sets the static DNS Address, dynamically set DNS Addresses cannot be set manually.
Parameters must be permanently saved with CNF_ApplySettings.

Input parameters:

Parameter	Value	Description
sInterface	'dns0', 'dns2'	String for desired Ethernet interface
abyDnsAddress	-	Array of Byte [0..3] Array of four byte fields, each field corresponds to a bit octal from the DNS Address.

Output parameters:

CNF_SetNetMask	0..1	0:	Successfully executed
		1:	Error occurred

Graphical display:
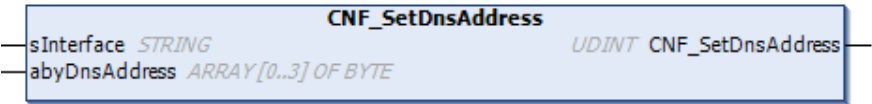
Sample call in ST:

```
udiReturn :=
CNF_SetDnsAddress(sInterface:=sEthInterface, abyDnsAddress:=abyDnsSet);
```

7.5.26. Read Mac address (FUN CNF_GetMacAddress)

Description: Reads the MAC Address from the respective Ethernet interface.

Parameter	Value	Description
Input parameters: sInterface	'eth0', 'eth1'	String for desired Ethernet interface
abyMacAddress	-	IN_OUT Array of Byte [0..5] Array from six byte fields, each field corresponds to a bit octal from the MAC Address.

	Parameter	Value	Description
Output parameters:	CNF_GetDnsAddress	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
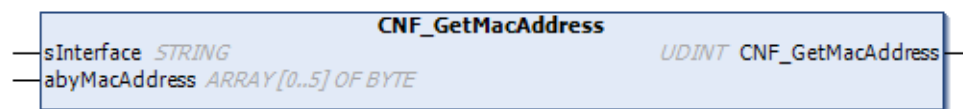
Sample call in ST:

udiReturn :=
CNF_GetMacAddress(sInterface:=sEthInterface, abyMacAddress:=abyMacGet);

7.5.27. Read storage location of the application (FUN CNF_GetApplicationOnSd)

Description: Reads the 'ApplicationOnSd' option from the system configuration to determine
if

the application is stored on the internal memory or SD card.

Parameter	Value	Description
Input parameters:		None

	Parameter	Value	Description
Output parameters: card	CNF_GetApplicationOnSd	TRUE, FALSE	FALSE: Not on the SD
			TRUE: On the SD card

Graphical display:
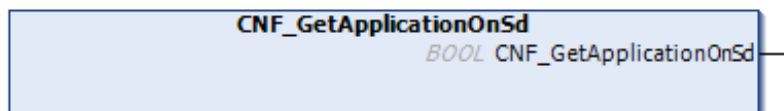
Sample call in ST:

xAppOnSd := CNF_GetApplicationOnSd();

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

133

7.5.28. Read name of the controller (FUN CNF_GetModuleName)

Description: Reads the module name displayed in the web interface from the controller.

Parameter	Value	Description
Input parameters:	sModulName	-	IN_OUT String to the time zone variable
			Module name is rewritten to transferred variable
Output parameters:	CNF_GetModuleName	0..1	0: Successfully executed
			1: Error occurred

Graphical display:

```
                        CNF_GetModuleName
— sModulName STRING          UDINT  CNF_GetModuleName —
```
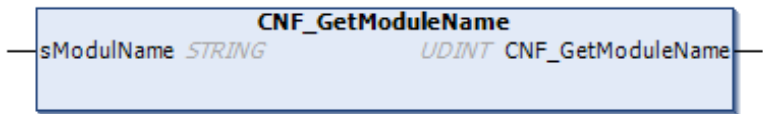
Sample call in ST:

udiReturn := CNF_GetModuleName(sModulName:=sGetModuleName);

7.5.29. Read article number of the controller (FUN CNF_GetModuleNumberString)

Description: Reads the article number displayed in the web interface from the controller.

Parameter	Value	Description
Input parameters: zone	sModulNumber	-	IN_OUT String to the time variable
			Article name is rewritten to transferred variable
Output parameters:	CNF_GetModuleNumberString	0..1	0: Successfully executed
			1: Error occurred

Graphical display:

```
                        CNF_GetModuleNumberString
— sModulNumber STRING          UDINT  CNF_GetModuleNumberString —
```
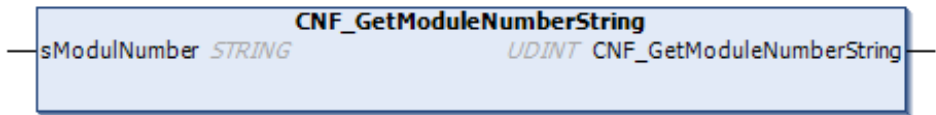
Sample call in ST:

udiReturn := CNF_GetModuleNumberString(sModulNumber:=sGetModuleNumber);

7.5.30. Read serial number of the controller (FUN CNF_GetSerialNumberString)

Description: Reads the serial number displayed in the web interface from the controller.

	Parameter	Value	Description
Input parameters:	sSerial	IN_OUT	String to the time zone variable
			Serial name is rewritten in transferred variable
Output parameters:	CNF_GetSerialNumberString	0..1	0: Successfully executed
			1: Error occurred

Graphical display:

```
                CNF_GetSerialNumberString
—sSerial STRING              UDINT  CNF_GetSerialNumberString—
```
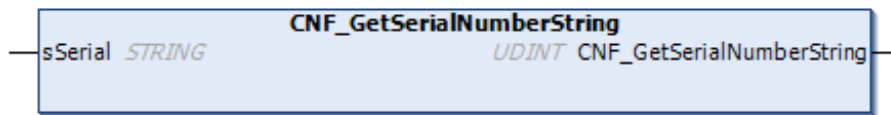
Sample call in ST:

udiReturn := CNF_GetSerialNumberString(sSerial:=sGetSerialNumber);

7.5.31. Read hardware version of the controller (FUN CNF_GetHardwareRevisionString)

Description: Reads the hardware version displayed in the web interface from the controller.

	Parameter	Value	Description
Input parameters:	sHwRevision	-	IN_OUT String to the timezone variable
			Hardware version is rewritten to transferred variable
Output parameters:	CNF_GetHardwareRevisionString0..1		0: Successfully executed
			1: Error occurred

Graphical display:

```
                CNF_GetHardwareRevisionString
—sHwRevision STRING          UDINT  CNF_GetHardwareRevisionString—
```
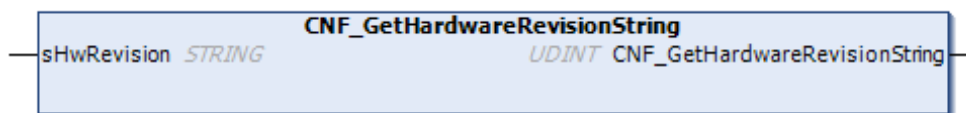
Sample call in ST:

udiReturn := CNF_GetHardwareRevisionString(sHwRevision:=sGetHwRevision);

7.5.32. Set retain behavior (FUN CNFRTS_SetRetainBehaviourOnOldData)

Description:	Sets the behavior of the controller upon detecting mismatched data in the retain memory. Parameters must be permanently saved with CNF_ApplySettings.

	Parameter	Value	Description
Input parameters:	eOption	0..1	0: DELETE_RETAIN_DATA Delete retain memory 1: KEEP_RETAIN_DATA Do not delete retain memory
Output parameters:	CNFRTS_**SetRetainBehaviour-OnOldData**	0..10: Successfully executed 1: Error occurred	

Graphical display:
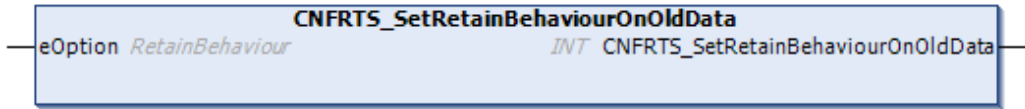
Sample call in ST:

iReturn:= CNFRTS_**SetRetainBehaviourOnOldData(eOption:=enRetainOption);**

7.5.33. Read status of the user switch S1 (FUN CNFRTS_GetOperatorButtonDisable)

Description: Reads whether the user switch S1 to start, stop and reset the application is activated

or deactivated.

Parameter	Value	Description
Output parameters: CNFRTS_**GetOperatorButtonDisable**	TRUE, FALSE	FALSE: S1 activated TRUE: S1 deactivated

Graphical display:
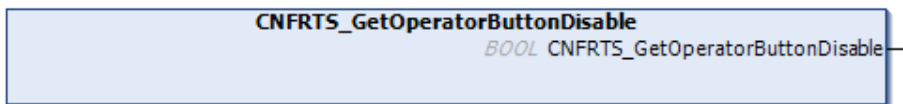
Sample call in ST:

xS1ButtonDisabled := CNF_**GetOperatorButtonDisable**();

7.5.34. Set status of the user switch S1 (FUN CNFRTS_SetOperatorButtonDisable)

Description: Enables or disables the user switch S1 to start, stop, and reset the application. Parameters must be permanently saved with CNF_ApplySettings.

Parameter	Value	Description
Input parameters: bDisable	TRUE,FALSE	TRUE: Deactivate S1 FALSE: Activate S1
Output parameters: CNFRTS_SetOperatorButtonDisable	0..10:	0: Successfully executed 1: Error occurred

Graphical display:
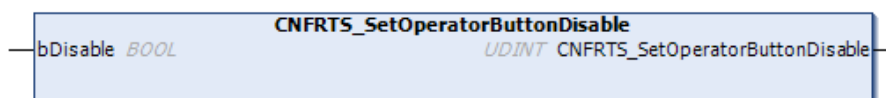
Sample call in ST:

udiReturn := CNF_**SetOperatorButtonDisable**(bDisable:=xS1ButtonDisable);

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

137

7.5.35. Read reset mode of the user switch S1 (FUN CNFRTS_GetOperatorButtonResetMode)

Description: Read which reset mode is executed when operating the user switch S1.

	Parameter	Value	Description
Input parameters:	peButtonResetMode	x	Pointer to the Enum of the Reset Mode. Current mode is rewritten in transferred enum.
			0: COLD – cold Reset
			1: WARM – warm Reset
Output parameters:	CNFRTS_**GetOperatorButtonResetMode**	0..1	0: Successfully executed
			1: Error occurred

Graphical display:

Sample call in ST:

udiReturn:=
CNF_**GetOperatorButtonResetMode** (peButtonResetMode:=ADR(enGetResetMode));

7.5.36. Set reset mode of the user switch S1 (FUN CNFRTS_SetOperatorButtonResetMode)

Description: Sets which reset mode is executed when operating the user switch S1.
                       Parameters must be permanently saved with CNF_ApplySettings.

	Parameter	Value	Description
Input parameters:	eButtonResetMode	-	Enum of the Reset Mode.
		0:	COLD – cold Reset
		1:	WARM – warm Reset
Output parameters:	CNFRTS_**SetOperatorButtonResetMode**	0..1	0: Successfully executed
		1:	Error occurred

Graphical display:
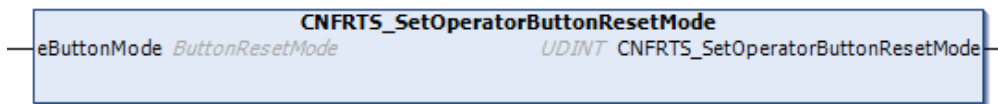
Sample call in ST:

udiReturn:= CNF_**SetOperatorButtonResetMode** (eButtonResetMode:= enSetResetMode);

7.5.37. Read USBUpdate behavior (FUN CNF_GetSkipUsbUpdateFlag)

Description: Reads whether the automatic execution of the USB update is skipped or not.

	Parameter	Value	Description
Output parameters:	**CNF_GetSkipUsbUpdateFlag** 0..2	0:	USB Update is not skipped
		1:	USB Update is skipped
		2:	Error occurred

Graphical display:
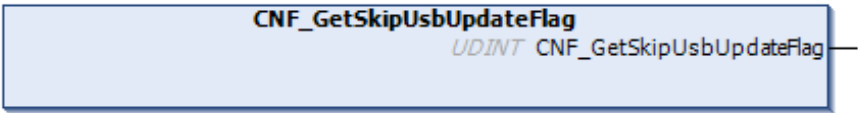
Sample call in ST:

udiUsbUpdateState := CNF_**GetSkipUsbUpdateFlag**();

7.5.38. Set USBUpdate behavior (FUN CNF_SetSkipUsbUpdateFlag)

Description: Sets whether the automatic execution of the USB update is skipped or not.

	Parameter	Value	Description
Input parameters:	xSkipUSBUpdate	TRUE,FALSE	TRUE: Skip USB Update
			FALSE:Do not skip USB Update
Output parameters:	**CNF_SetSkipUsbUpdateFlag**	0..2	0: USB Update is not
			1: USB Update is skipped
			2: Error occurred

Graphical display:
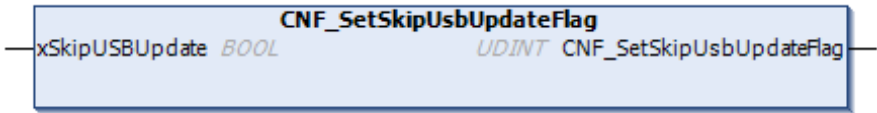
Sample call in ST:

udiUsbUpdateState := CNF_**SetSkipUsbUpdateFlag** (xSkipUSBUpdate:=xSkipUSB);

7.5.39. Generate backup of the application (FUN PLC_SaveApplicationToFile)

Description: Creates an archive with the backup of the application folder at the selected
storage path
Warning: Depending on the application size, this function may take several seconds; so this function may only be executed in low-priority tasks.

	Parameter	Value	Description
Input parameters:	sDstFilename	-	String for memory path, e.g. '/media/usb1/appbackup.tgz'
Output parameters:	**PLC_SaveApplicationToFile**	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
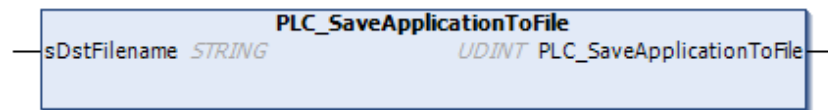
Sample call in ST:

udiReturn := **PLC_SaveApplicationToFile**(sDstFilename:=sFilePath);

7.5.40. Read available memory (FUN FS_DiskTotal)

Description: Reads the total available memory of a data medium.

	Parameter	Value	Description
Input parameters:	sPath	-	String for memory path, e.g. '/flash/' oder '/media/usb1/'
Output parameters:	FS_DiskTotal	0..x	Available memory in KB

Graphical display:
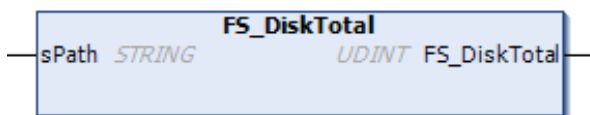
Sample call in ST:

udiTotalSpace := FS_DiskTotal(sPath:=sDiskPath);

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

141

7.5.41. Read free memory (FUN FS_DiskFree)

Description: Reads the free memory of a data medium.

	Parameter	Value	Description
Input parameters:	sPath	-	String for memory path, e.g. '/flash/' oder '/media/usb1/'
Output parameters:	FS_DiskFree	0..x	Free memory in KB

Graphical display:

```
           FS_DiskFree
─| sPath STRING      UDINT FS_DiskFree |─
```
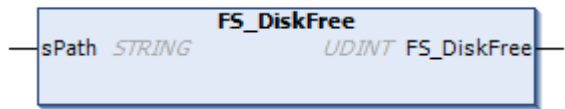
Sample call in ST:

udiFreeSpace := FS_DiskFree(sPath:=sDiskPath);

7.5.42. Read USB Status (FUN USB_GetPlugStatus)

Description: Reads whether a USB device is plugged in on the USB interface.

	Parameter	Value	Description
Input parameters:	nUsbDevice	0..x	UINT for USB device, e.g. 0 for 'usb1' or 1 for 'usb2' etc.
Output parameters:	USB_GetPlugStatus	0..1	0: no device inserted 1: Device inserted

Graphical display:

```
          USB_GetPlugStatus
─| nUsbDevice UINT    INT USB_GetPlugStatus |─
```
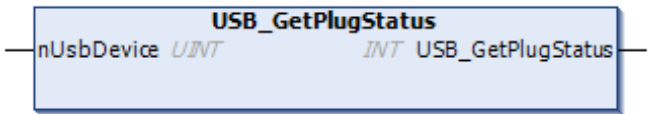
Sample call in ST:

iPlugStatus := USB_GetPlugStatus(nUsbDevice:=uiUsbDevice);

7.5.43. Read USB Mount Status (FUN USB_GetMountStatus)

Description: Reads whether the inserted USB device has been correctly mounted by the Berghof controller. A USB device can only be used after a successful mount.

	Parameter	Value	Description
Input parameters:	nUsbDevice	0..x	UINT for USB device, e.g. 0 for 'usb1' or 1 for 'usb2' etc.
Output parameters:	USB_GetMountStatus	0..1	0: Device not integrated
			1: Device integrated

Graphical display:
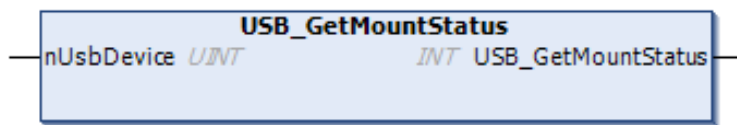
Sample call in ST:

iMountStatus := USB_GetMountStatus(nUsbDevice:=uiUsbDevice);

7.5.44. Mount USB device (FUN USB_MountDisk)

Description: Normally, USB devices are automatically mounted by the Berghof system. With this function, then mounting process is manually executed. A USB device can only be used after a successful mount.

	Parameter	Value	Description
Input parameters:	nUsbDevice	0..x	UINT for USB device, e.g. 0 for 'usb1' or 1 for 'usb2' etc.
	dwOptions	0..x	Enum for options 0: Default – Standard settings
Output parameters:	USB_MountDisk	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
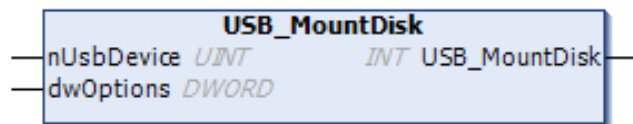
Sample call in ST:

iReturn := USB_MountDisk(nUsbDevice:=uiUsbDevice, dwOptions:=0);

7.5.45. Unmount USB device (FUN USB_UMountDisk)

Description:
and

Ends the connection from the USB device to the Berghof system. To avoid data loss

on

damage to the file system, it is recommended to perform this function before removing the USB device. Furthermore, before unmounting, all open file and directory handles on

the USB device must be closed.

	Parameter	Value	Description
Input parameters:	nUsbDevice	0..x	UINT for USB device, e.g. 0 for 'usb1' or 1 for 'usb2' etc.
Output parameters:	USB_UMountDisk	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
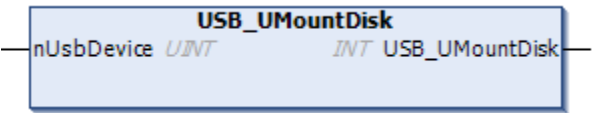
Sample call in ST:

iReturn := USB_UMountDisk(nUsbDevice:=uiUsbDevice);

7.5.46. Use Buzzer (FUN SND_Buzzer)

Description:

Controls the buzzer on the controller with the volume transferred.
Only works with controllers with built-in buzzer.

	Parameter	Value	Description
Input parameters:	byVolume	0..4	Volume value for the buzzer
			0: Off
			4: Maximum volume
	udiFrequency	0..x	0: Standard
			Must be supported by hardware
Output parameters:	SND_Buzzer	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
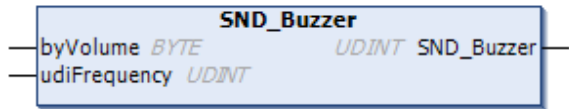
Sample call in ST:

udiReturn := SND_Buzzer(byVolume:=byBuzVolume, udiFrequency:=0);

7.5.47. Play back audio file (FUN SND_PlaySoundFile)

Description: Plays back the audio file specified in the path. The file must be in MP3 format
on

the controller.
Works only on controllers with built-in audio chip. So far no Berghof device with
an audio chip is available. Function for internal purposes only.

	Parameter	Value	Description
Input parameters:	sSoundFile	-	String for memory path e.g. '/home/plc/applications/alarm.mp3'
Output parameters:	SND_PlaySoundFile	0..1	0: Successfully executed 1: Error occurred

Graphical display:

```
                 SND_PlaySoundFile
— sSoundFile STRING        UDINT  SND_PlaySoundFile —
```
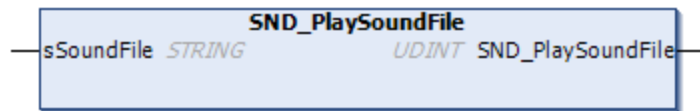
Sample call in ST:

udiReturn := SND_PlaySoundFile (sSoundFile:=sPathAlarm);

7.5.48. Stop playback of the audio file (FUN SND_StopSoundFile)

Description: Stops playback of the currently playing audio file.
Works only on controllers with built-in audio chip. So far no Berghof device with
an audio chip is available. Function for internal purposes only.

	Parameter	Value	Description
Output parameters:	SND_StopSoundFile	0..1	0: Successfully executed 1: Error occurred

Graphical display:

```
           SND_StopSoundFile
               UDINT  SND_StopSoundFile —
```
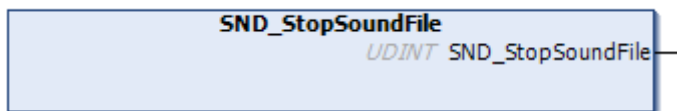
Sample call in ST:

udiReturn := SND_StopSoundFile();

7.5.49. Read the playback volume (FUN SND_GetSoundVolume)

Description:	Reads the set playback volume. Works only on controllers with built-in audio chip. So far no Berghof device with an audio chip is available. Function for internal purposes only.			

	Parameter	Value	Description	
Input parameters:	byChannel	0	0:	Default – Master Channel
	ByVolume	0..100	IN_OUT Byte for volume Volume value is rewritten in transferred variable	
			0:	Audio off
			100:	max. volume
Output parameters:	SND_GetSoundVolume	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:

```
                 SND_GetSoundVolume
—byChannel BYTE              UDINT SND_GetSoundVolume—
—byVolume BYTE
```
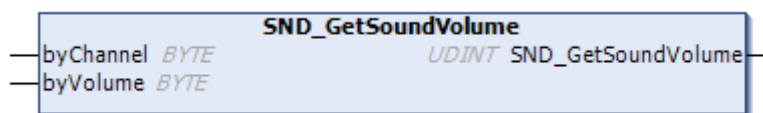
Sample call in ST:

udiReturn := SND_GetSoundVolume(byChannel:=0,byVolume:=byGetVolume);

7.5.50. Set the playback volume (FUN SND_SetSoundVolume)

Description:	Sets the transferred value for playback volume. Works only on controllers with built-in audio chip. So far no Berghof device with an audio chip is available. Function for internal purposes only.			

	Parameter	Value	Description	
Input parameters:	byChannel	0	0:	Default – Master Channel
	ByVolume	0..100	0:	Audio off
			100:	max. volume
Output parameters:	SND_SetSoundVolume	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:

```
                 SND_SetSoundVolume
—byChannel BYTE              UDINT SND_SetSoundVolume—
—byVolume BYTE
```
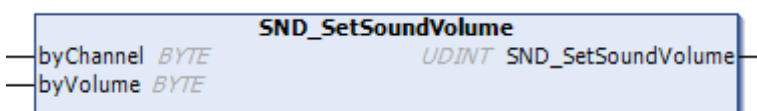
Sample call in ST:

udiReturn := SND_SetSoundVolume(byChannel:=0,byVolume:=bySetVolume);

7.5.51. Read out received bytes in multi-drop mode (FUN COM_GetAddrBytesRx)

Description: Reads out the number of received bytes at the selected serial connection.
                      Only works in multi-drop mode (9 bit set).
                      Warning: Recommended only for advanced users; can lead to unstable application.

	Parameter	Value	Description
Input parameters:	hCom	-	ComPort Handle by ComOpen retrieval of the SysCom Lib
Output parameters:	COM_GetAddrBytesRx	0..x	Number of received bytes in the Multidrop Mode (9 Bit Set)

Graphical display:
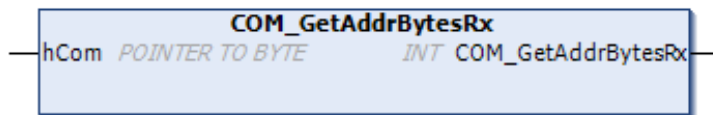
Sample call in ST:

iBytesRec := COM_GetAddrBytesRx (hCom:=hComPort2);

7.5.52. Set bytes to be sent in Multidrop Mode (FUN COM_SetAddrBytes)

Description: Sets the number of bytes to be sent for the next data exchange with a 9-bit set
                      on the transferred serial connection.
                      Only works in multi-drop mode (9 bit set).
                      Warning: Recommended only for advanced users; can lead to unstable application.

	Parameter	Value	Description
Input parameters:	hCom	-	ComPort Handle by ComOpen retrieval of the SysCom Lib
	uiCnt	0..x	Number of bytes to be sent
Output parameters:	COM_SetAddrBytes	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
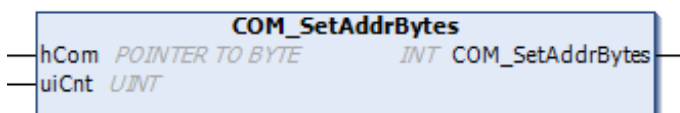
Sample call in ST:

iBytesRec := COM_SetAddrBytes(hCom:=hComPort2,uiCnt:=uiTxBytes);

7.5.53. Read statistics of a serial interface (FUN COM_GetSioTimes)

Description: Reads advanced statistic values from the selected serial connection.
Only relevant in the Multidrop Mode (9 Bit Set)
Warning: Recommended only for advanced users; can lead to unstable application.

	Parameter	Value	Description
Input parameters:	hCom	-	ComPort Handle by ComOpen retrieval of the SysCom Lib
	pSioTimes	-	IN OUT structure for the statistical values Statistical values are rewritten here
Output parameters:	COM_GetSioTimes	0..1	0: Successfully executed 1: Error occurred

Graphical display:
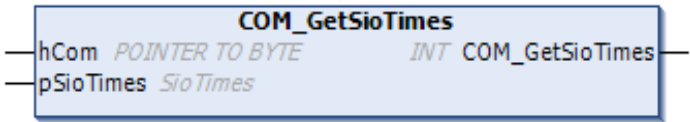
Sample call in ST:

iReturn := COM_GetSioTimes (hCom:=hComPort2,pSioTimes:=stComPortStats);

7.5.54. Set own baud rate for serial interface (FUN COM_SetCustomBaudrate)

Description: Sets a self-selected non-predefined baud rate on the selected serial interface.

Warning: Recommended only for advanced users; can lead to unstable application.

	Parameter	Value	Description
Input parameters:	hCom	-	ComPort Handle by ComOpen retrieval of the SysCom Lib
	uiBaudrate	0..x	Baud rate in Bits/s
Output parameters:	COM_SetCustomBaudrate	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
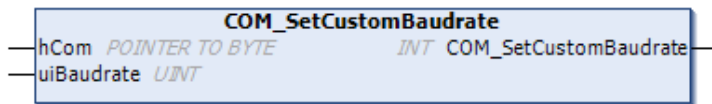
Sample call in ST:

iReturn := COM_SetCustomBaudrate(hCom:=hComPort2, uiBaudrate:=uiCustomRate_28k);

7.5.55. Set mode of the serial interface (FUN COM_SetMode)

Description: Sets the selected serial interface to the RS232 or RS485 Mode.

Warning: Works only on Powertrack controllers.

Warning: Recommended only for advanced users; can lead to unstable application.

	Parameter	Value	Description
Input parameters:	hCom	-	ComPort Handle by ComOpen retrieval of the SysCom Lib
	eMode	-	Enum of the ComMode.
			0: RS232
			1: RS485
Output parameters:	COM_SetMode	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
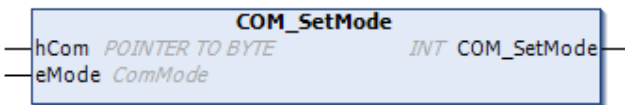
Sample call in ST:

iReturn := COM_ SetMode(hCom:=hComPort2, eMode:=enComMode);

7.5.56. Set Multidrop Mode on the serial interface (FUN COM_SetMode9Bit)

Description:

Sets the selected serial interface to Multidrop (9 bit set) mode.

Warning: Works only on RS485 interfaces.

Can only be reset by restarting the controller.

Warning: Recommended only for advanced users; can lead to unstable application.

	Parameter	Value	Description
Input parameters:	hCom	-	ComPort Handle by
ComOpen			
			retrieval of the SysCom Lib
Output parameters:	COM_SetMode9Bit	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
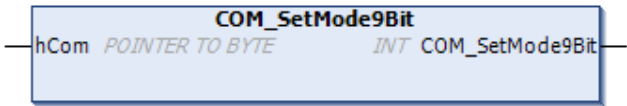
Sample call in ST:

iReturn := COM_SetMode9Bit(hCom:=hComPort2, eMode:=enComMode);

7.5.57. Read BusOff Recovery time of a CAN interface (FUN CAN_GetBusOffRecoveryCycletime)

Description: Reads the cycle time of the automatic BusOff Recovery at the selected CAN interface.

	Parameter	Value	Description
Input parameters:	sInterface	'can0','can1'	String for CAN interface
	restart_cycletime_ms	0..x	IN OUT UINT for time
			Time value is rewritten here
			0: Auto-Recovery deactivated
			>1: Auto-Recovery time in ms
Output parameters:	CAN_**GetBusOffRecoveryCycletime**	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
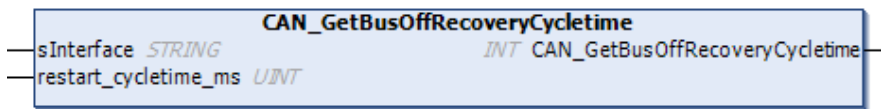
Sample call in ST:

iReturn := CAN_**GetBusOffRecoveryCycletime**
(sInterface:= sCanInt1, restart_cycletime_ms:=uiAutoRecCycle);

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

151

7.5.58. Set BusOff Recovery time of a CAN (FUN CAN_SetBusOffRecoveryCycletime)

Description:

Sets the cycle time of the automatic BusOff Recovery at the selected CAN interface.

Warning: When calling this function, the CAN interface is stopped and then restarted

	Parameter	Value	Description	
Input parameters:	sInterface	'can0','can1'	String for CAN interface	
	restart_cycletime_ms	0..x	0:	Deactivate Auto-Recovery
			>1:	Set Auto-Recovery time in ms
Output parameters:	CAN_**SetBusOffRecoveryCycletime** 0..1		0:	Successfully executed
			1:	Error occurred

Graphical display:
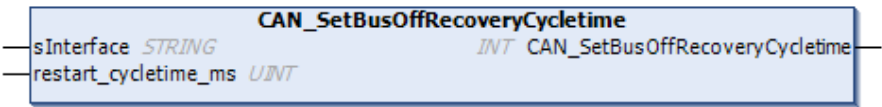
Sample call in ST:

iReturn := CAN_**SetBusOffRecoveryCycletime**(sInterface:= sCanInt1,
restart_cycletime_ms:=uiAutoRecCycle);

7.5.59. Read statistics of a CAN interface (FUN CAN_GetStatistics)

Description: Reads statistical values from the selected CAN interface.

	Parameter	Value	Description
Input parameters:	sInterface	'can0','can1'	String for CAN interface
	stats	-	IN OUT structure for statistical values
			Statistical values are rewritten here
Output parameters:	CAN_GetStatistics 0..1		0: Successfully executed
			1: Error occurred

Graphical display:
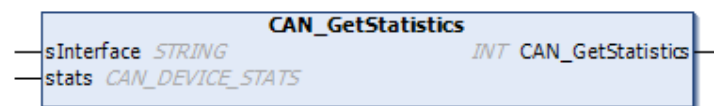
Sample call in ST:

iReturn := CAN_GetStatistics (sInterface:=sCanInt1, stats:=stCan1Stats);

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

153

7.5.60. Read statistics of a CAN interface (FUN CAN_GetStatus)

Description: Reads the status of the selected CAN interface.

	Parameter	Value	Description
Input parameters:	sInterface	'can0','can1'	String for CAN interface
	eStatus	0..5	IN OUT Enum for Status
			Statistical values are rewritten here
			0: CAN_STATE_ERROR_ACTIVE CAN active (<96 Error Frames)
			1: CAN_STATE_ERROR_WARNING CAN active (<128 Error Frames)
			2: CAN_STATE_ERROR_PASSIVE CAN inactive (<256 Error Frames)
			3: CAN_STATE_ERROR_BUS_OFF CAN off (>=256 Error Frames)
			4: CAN_STATE_STOPPED CAN stopped
			5: CAN_STATE_ERROR_SLEEPING CAN in Standby
Output parameters:	CAN_GetStatus	0..1	0: Successfully executed
			1: Error occurred

Graphical display:
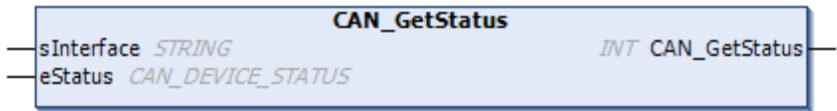
Sample call in ST:

iReturn := CAN_GetStatus(sInterface:=sCanInt1, stats:=stCan1Stats);

7.5.61. Restart of a CAN interface from the BusOff (FUN CAN_RestartFromBusOff)

Description: Executes a restart of the selected CAN interface.
                Only works if the CAN interface is in the BusOff.

	Parameter	Value	Description	
Input parameters:	sInterface	'can0','can1'	String for CAN interface	
Output parameters:	CAN_RestartFromBusOff	0..1	0:	Successfully executed
			1:	Error occurred

Graphical display:
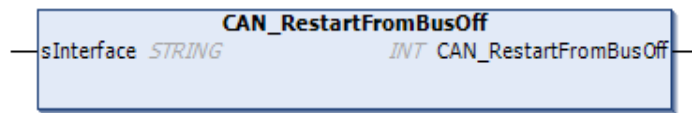
Sample call in ST:

iReturn := CAN_RestartFromBusOff (sInterface:=sCanInt1);

7.5.62. Read mode of the SC-CAN interface (FUN POWT_GetSCCanListenOnlyMode)

Description: Reads if the selected CAN interface is in Listen Only Mode.
                Warning: Only works with Powertrack controllers and only for the SC-CAN interface.

	Parameter	Value	Description	
Input parameters:	uiInterface	0..1	0:	CAN0
			1:	CAN1
Output parameters:	**POWT_GetSCCanListenOnlyMode**	0..x	0:	Listen Only Mode inactive
			1:	Listen Only Mode active
			x:	Error occurred

Graphical display:
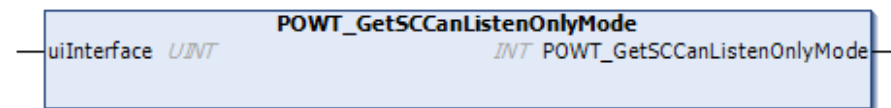
Sample call in ST:

iReturn := **POWT_GetSCCanListenOnlyMode**(uiInterface:=uiCAN0);

7.5.63. Set mode of the SC-CAN interface (FUN POWT_SetSCCanListenOnlyMode)

Description: Sets the selected CAN interface to Listen Only mode.
Warning: Only works with Powertrack controllers and only for the SC-CAN interface.

	Parameter	Value	Description	
Input parameters:	uiInterface	0..1	0:	CAN0
			1:	CAN1
	eState	-	Enum of the CAN Mode	
			0:	INACTIVE Deactivate Listen Only Mode
			1:	ACTIVE Activate Listen Only Mode
Output parameters:	**POWT_SetSCCanListenOnlyMode**	0..x	0:	Listen Only Mode inactive
			1:	Listen Only Mode active
			x:	Error occurred

Graphical display:
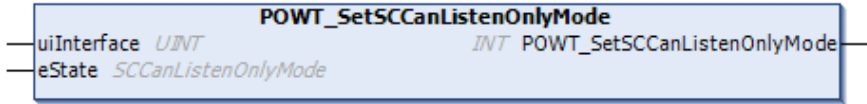
Sample call in ST:

iReturn := **POWT_SetSCCanListenOnlyMode**(uiInterface:=uiCAN0,eState:=enCanMode);

7.5.64. Read UPS activity (FUN POWT_GetUpsActive)

Description: Reads if the controller is powered by the UPS.
                      Warning: Works only in Powertrack controllers with integrated UPS.

	Parameter	Value	Description	
Output parameters:	**POWT_GetUpsActive**	0..x	0:	No voltage from UPS
			1:	Voltage from UPS
			x:	Error occurred

Graphical display:
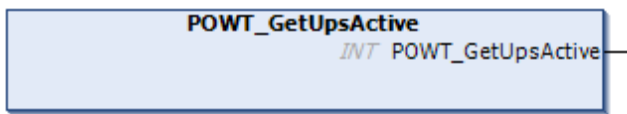
Sample call in ST:

iReturn := **POWT_GetUpsActive** ();

7.5.65. Read UPS charge (FUN POWT_GetUpsPowerGood)

Description: Reads whether the charge saved in the UPS is sufficient for operating the con-
trol-

                      ler.
                      Warning: Works only in Powertrack controllers with connected UPS.

	Parameter	Value	Description	
Output parameters:	**POWT_GetUpsPowerGood**	0..x	0:	UPS charge too low
			1:	UPS charge OK
			x:	Error occurred

Graphical display:
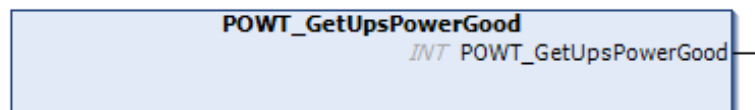
Sample call in ST:

iReturn := **POWT_GetUpsPowerGood**();

7.5.66. Set Active Transceiver at the RS485 interface (FUN POWT_SetGpioMode2EMosi)

Description:
Sets whether RS485 transceiver 0 or 1 is active on the hardware interface.
Warning: Only works for CCPU-SC-IMX (202800022) controllers.
Must not be used with other Powertrack controllers.

	Parameter	Value	Description	
Input parameters:	xValue	TRUE,FALSE	FALSE:	Transceiver 0 active
			TRUE:	Transceiver 1 active
Output parameters:	**POWT_SetGpioMode2EMosi** 0..x		0:	Successfully executed
			1:	Error occurred

Graphical display:
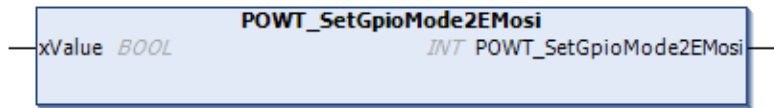
Sample call in ST:

iReturn := **POWT_S**e**tGpioMode2EMosi**(xValue:=xRS485Transceiver);

7.5.67. Activate SC-CAN outlet 0 (FUN POWT_SetGpioSc_En0)

Description:
Enables or disables the outlet 0 of the SC-CAN.
Warning: Only works for CCPU-SC-IMX (202800022) controllers.
Must not be used with other Powertrack controllers.

	Parameter	Value	Description	
Input parameters:	xValue	TRUE,FALSE	FALSE:	Outlet 0 inactive
			TRUE:	Outlet 0 active
Output parameters:	**POWT_SetGpioSc_En0**	0..x	0:	Successfully executed
			1:	Error occurred

Graphical display:
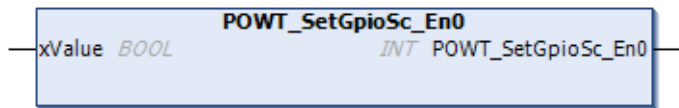
Sample call in ST:

iReturn := **POWT_SetGpioSc_En0**(xValue:=xSC0Activate);

7.5.68. Activate SC-CAN outlet 1 (FUN POWT_SetGpioSc_En1)

Description: Enables or disables the outlet 1 of the SC-CAN.
Warning: Only works for CCPU-SC-IMX (202800022) controllers.
Must not be used with other Powertrack controllers.

	Parameter	Value	Description	
Input parameters:	xValue	TRUE,FALSE	FALSE:	Outlet 1 inactive
			TRUE:	Outlet 1 active
Output parameters:	**POWT_SetGpioSc_En1**	0..x	0:	Successfully executed
			1:	Error occurred

Graphical display:
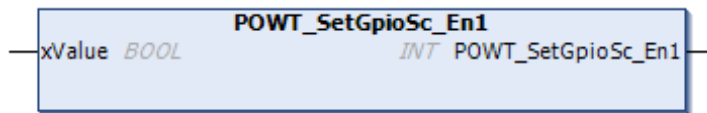
Sample call in ST:

iReturn := **POWT_SetGpioSc_En1**(xValue:=xSC1Activate);

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

159

8. Advanced configuration and access options

8.1. Access via FTP

A frequent use case of PLC is the storage of data on the controller's internal memory or SD card. To gain easy access to this data from outside, you can activate an FTP server in the Berghof controllers and then retrieve the data with an FTP client.

How the FTP server is activated via the web interface can be read in chapter 0. After restarting the controller, connect to the FTP server via the IP address of the controller and the standard FTP port 21. If you have not yet installed an FTP client, we recommend that you use the free software "FileZilla", which runs under Windows, MacOS X and Linux.

As a developer, you can use the "root" account to load files from or to the controller. In productive use, it is recommended to use the user groups "ftpuser" or "ftpreader" as they cannot leave the ftp directory (/flash/ftpupload) or the start directory set in the Customuser and "ftpreader" has no additional write access to the directory.
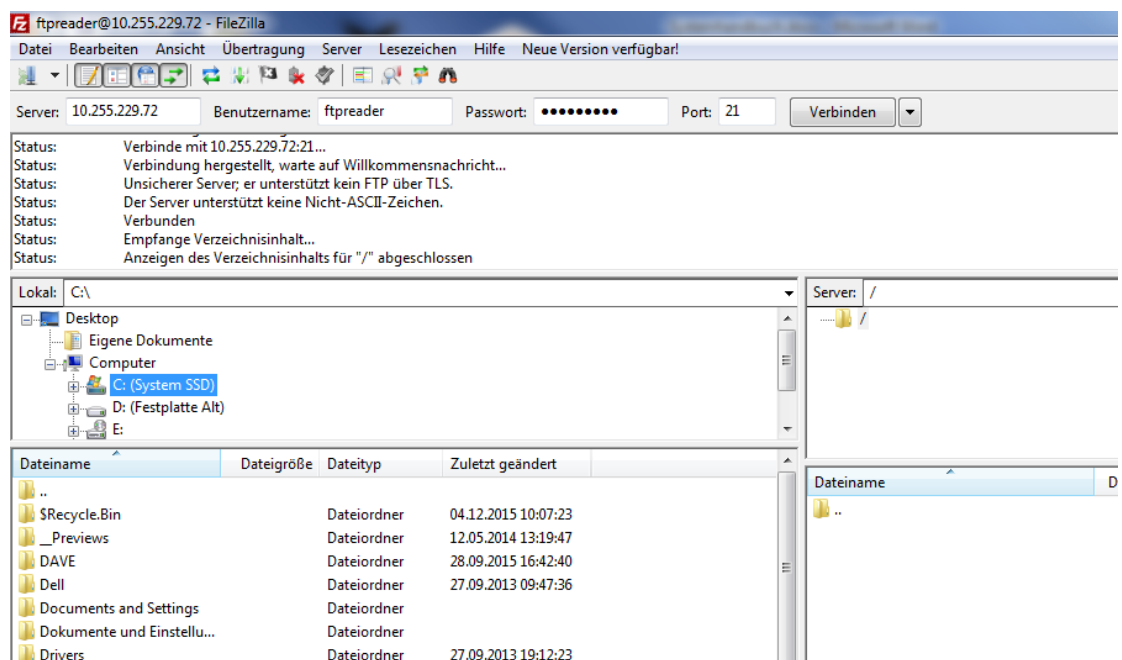
Picture 67: FTP-Reader

8.2. File system and folder structure

Most folders and files in the file system of the controller are not accessible to the user or can only be read for security reasons. However, there are a few directories that end users can describe and use to utilize certain functionalities of the controller.

Directory	Description
/flash	Internal Flash memory of the controller
/flash/plc	Standard directory of CODESYS V3. Also mounted at /home/plc.
/flash/plc/applications	Storage location of the CODESYS V3 Application
/flash/plc/applications/fonts	Location for own fonts, must be created once if own fonts are to be used.
/flash/ftpupload	Standard directory of the FTP user. Here, data read or written via FTP should be stored.
/media/sd	External flash memory (SD card), only available if there is an SD card.
/media/usb1	External USB storage, available only when a storage medium is inserted.

8.3. Install additional fonts (fonts)

By default, the Berghof controllers have installed the fonts of the DejaVu family. However, if you want to use other fonts (such as Windows fonts) for web or target visualizations, you can load any Truetype fonts (*.ttf) into the /flash/plc/applications/fonts/ folder and then use them.
You must create this folder once if you have not yet used your own fonts. It is recommended to enable the FTP server of the controller and then load the fonts via FTP to the controller.

⚠ CAUTION

Due to the system, the file extension of the fonts must always be lowercase, otherwise the file will not be recognized by the system and may need to be renamed. E.g. if the Arial Unicode font is named "ARIALUNI.ttf", "ARIALUNI.TTF" or "ARIALUNI.Ttf" will not be recognized by the controller.

⚠ CAUTION

You can use either the preinstalled or the additional fonts. If you want to use your own fonts and the preinstalled DejaVu fonts at the same time, you must first download the corresponding DejaVu fonts from the "/var/lib/fonts" directory or via the web interface and then, in addition to your own fonts, again upload them to the "/flash/plc/applications/fonts/".

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

161

> ### ℹ **NOTE**
>
> **Many fonts which are known from the everyday use of PCs are licensed. Arial, for example, is a Microsoft font which is licensed via Windows.**
>
> When using alternative fonts, please make sure that they are not licensed and are freely available.

8.4. Update firmware or configuration settings "offline" using a USB drive

Apart from the web interface, you can also configure or update your controller using a specially prepared USB drive. This is especially useful if you need to update or configure many controllers at the same time, or you do not have access to the web interface of the controller.

To perform such a configuration change or a firmware update, you need a USB drive that has been formatted with FAT32. We recommend using USB drives with a size up to 32GB.

You must download the file "usbupdate-xxx_x.x.x.zip" from our closed download area. Unzip this file into the base directory of the USB drive, so that there is now a single folder with the name "usbupdate-mx6" or "usbupdate-pi4". This folder contains a predefined folder structure with additional subfolders and various "*.ini" files.

Many settings can be made via these "*.ini" files. It should be noted that an incorrectly edited INI file may mean that the update has no effect, that individual configurations are not accepted, or that in the worst case the controller is configured incorrectly, which may result in the controller not functioning anymore. Information on the structure of the INI files can be found here: https://en.wikipedia.org/wiki/INI_file

8.4.1. Edit Usbupdate.ini

The file "usbupdate.ini" in the directory "usbupdate-mx6" or "usbupdate-pi4" on the USB drive is the central configuration file. In this file, it is configured what actions are performed when booting the USB drive. If this file does not exist, the file "usbupdate_TEMPLATE-ET.ini" or "usbupdate_TEMPLATE-PLC.ini" must be renamed to "usbupdate.ini" and then edited depending on the controller type. The usbupdate.ini is divided into five sections and each of these sections has at least one pair of keys and value.
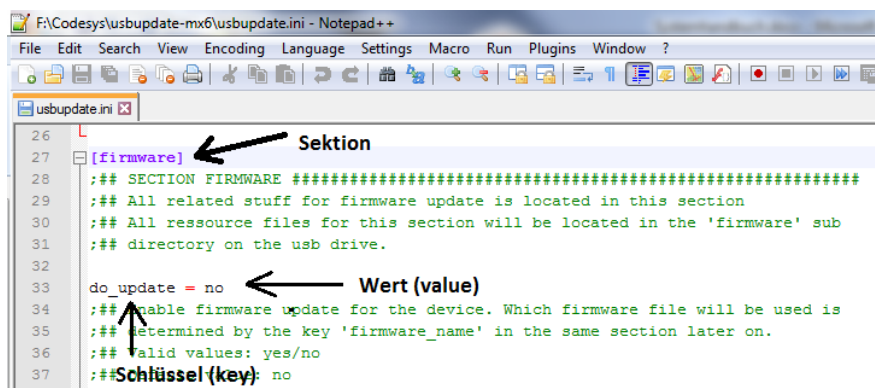
Picture 68: Update Settings

All entries in the usbupdate.ini have a short description of the function in English. Most values are of the "boolean" type and can therefore assume the values "yes" or "no".

The following five sections exist:
→ [firmware]: Settings for firmware upgrades or downgrades.
→ [webtheme]: Settings for exchanging the logo visible in the web interface.
→ [splashscreen]: Settings for exchanging the boot logo of a display control.
→ [sysconfig]: Settings for changing the system configuration.
→ [plcapp]: Settings for executing an application update or a copy process.

For each of these sections, there exists a folder of the same name on the USB drive. The files required for the respective execution are then stored in these folders.
Since the usbupdate.ini is just a text file with special formatting, it can be edited with any text editor. However, it is recommended to take an editor which masters the Ini format and format the text correctly and also saved in color, such as the free software Notepad++.

Each section and every key/value pair can be used independently. So, you can create a USB update that changes only a single setting or later copy files to the controller without making an application update.
The user can choose here freely and if necessary, create different USB memories for different occasions. Missing or commented out entries are simply ignored.
The possible entries of all sections will be briefly introduced in the following chapters.

8.4.2. USB update: Section [firmware]

The firmware section consists of the following keys:

Key	Possible values	Description
do_update	yes / no	Should a firmware update be carried out?
firmware_name	File name (e.g. firmware_mx6-plc_1.5.0.tgz)	Which firmware should be installed? The file must be in the subfolder "firmware".

The latest firmware can be downloaded from the closed download area. Please read the chapter 5.2 of this manual before upgrading the firmware.

8.4.3. USB update: Section [webtheme]

The webtheme section consists of the following key:

Key	Possible values	Description
do_update_webtheme	yes / no	Should the logo of the web interface be changed?

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

163

The logo must be a GIF image file named "logo.gif" and located in the "webtheme" subfolder. The image is scaled in the web interface by the browser, but it is still recommended to create the logo directly in the appropriate resolution for the purpose.

8.4.4. USB update: Section [splashscreen]

The splash screen section consists of the following key:

Key	Possible values	Description
do_update_splashscreen	yes / no	Should the splash screen (start screen) of the controller be replaced?

The splash screen must be a PNG image file named splash.png and located in the "splashscreen" subfolder. This setting is only relevant for display controls and E-terminals. If this option is activated on a controller without a display, the step is skipped. The splash screen is not scaled by the controller, so create the image file in the resolution required for the controller.

8.4.5. USB update: Section [license]

The License section consists of the following key:

Key	Possible values	Description
do_update_licenses	yes / no	Should the licenses of the controller be updated? The current licenses on the controller are overwritten with the licenses from the file "BGHlicense.lic" in the folder "license".

This function is rarely needed by end users, since normally all relevant licenses are loaded during the production of the controller. If you subsequently require an additional license (for example for Modbus/TCP), please note that when ordering the additional license, you specify which licenses are already on the controller, as the existing licenses are overwritten.

8.4.6. USB update: Section [sysconfig]

The Sysconfig section consists of the following keys:

Key	Possible values	Description
do_reset_syscfg_to_factory_defaults	yes / no	Should the configuration on the controller be reset to the delivery state?
do_sysconfig_from_file	yes / no	Do you want to update the configuration on the controller with the settings on the USB storage medium?
replace_config_file_instead_of_merge	yes / no	Combines the options "Reset to factory settings" and "Update settings from USB storage medium".

Many settings of the controller can be changed via an external file. The configuration settings to be updated are defined in the "configuration.ini" file in the "sysconfig" folder. The individual options are described in more detail in chapter 8.4.8.

8.4.7. USB update: Section [plcapp]

The Plcapp section consists of the following keys:

Key	Possible values	Description
do_clean_plcfolder	yes / no	Should the contents of the application folder on the controller (/flash/plc/applications/) be completely deleted?
do_update_plcapp	yes / no	Should an update of the control programs be carried out? All files contained in the update archive are written to the controller. Existing files with the same name will be overwritten in this case.
plcapp_name	File name (e.g. plcapplication.tgz)	Name of the file used for the "do_update_plcapp" action. This file must be located in the "application" folder and should have been previously downloaded from a controller via the web interface. Further information can be found in the chapter 4.3.4 (action "Download folder from PLC").
do_copy_plcdata	yes / no	Should all files and folders contained in the "application/data" folder be copied to the controller? The files are copied to the controller in the folder "/home/plc/applications/".

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

165

8.4.8. USB update: Change the settings of the controller via the file "configuration.ini"

All settings of the controller that can be set via the web interface can be changed automatically via a USB update. The basis is the file "configuration.ini" in the folder "sysconfig".
In this file, as in the "usbudate.ini", different sections are stored with key/value pairs.
Only some default sections and keys are presented in this document. For more information about available sections and keys, please contact technical support (support-controls@berghof.com).

Section [network]

Key	Possible values	Description
eth0.mode	"static" / "dhcp" / "inactive"	Mode of the interface.
eth0.ip	IP Address (e.g. 169.255.254.31)	IP Address of the controller. Is ignored in the "dhcp" and "inactive" mode.
eth0.netmask	Netmask (e.g. 255.255.0.0)	Netmask of the interface.
default_gateway	IP Address (e.g. 169.255.1.1)	Gateway address used to create the routing table. Is required if the controller needs access to additional IP networks (such as the Internet).

Naturally, the second network interface of the controller can also be configured using the appropriate eth1.xxx key. Same is possible with the virtual interfaces eth0:1 and eth1:1.

Section [ftp]

Key	Possible values	Description
enabled	"0" / "1"	With "1", the FTP server is activated, with "0" it is deactivated (default setting).

Section [vnc]

Key	Possible values	Description
screen.size	"480x272" / "640x480" / "800x480" / "1280x800" / "1366x768" / "1920x1080"	Resolution for the VNC server.
screen.depth	"16" / "32"	Color depth of the VNC visualization.

The VNC settings are only relevant for Berghof controllers without display. Controllers with display have fixed their resolution permanently; in this case you can remove the entry from the configuration.ini.

> **i NOTE**
>
> If you already have configurd your controller over the webinterface you can get actual configuration file from the PLC to use in your USB Update. Connect to the controller via FTP using the "ftpadmin" user. Leave the "ftpupload" folder and navigate to the folder "etc". Look and download the file "settings.ini". In this file you can see the all the section and key entries from the settings that were performed over the webinterface. You can use these entries for your "configuration.ini"

8.4.9. Perform USB update

Prepare your "usbupdate-mx6" or "usbupdate-pi4" folder and copy it onto the root directory of your USB drive.

1. Power off your controller and plug in the USB drive into a free USB port
2. Power on you controller and watch the state LED
3. When the USB Update is properly recognized and started the state LED will start blinking with an orange light, if you are using a controller with a display the update progess log will be shown on screen.
4. The USB Update will run through its configured settings and update the controller.
   Depending on the settings the USB Update might restart the controller several times.
   As long as the state LED is blinking DO NOT unplug the USB drive or power off the controller.
5. The USB Update is finished when the state LED is either constantly red (stopped) or green (run).
6. After the USB Update is done unplug the USB drive

8.4.10. USB update: Troubleshooting

If a USB update is not performed or fails, there can be several reasons for this. In any case, check the log file that creates the USB update on the USB memory. This file is in the "usbupdate-mx6" folder. The name of the file starts with the part number of the controller and has the file extension ".log".
Other common sources of error are:

→ Check if the USB drive is formatted in FAT32 file system.
→ Check if the USB drive has a valid partition table
→ Check if the folder "usbupdate-mx6" or "usbupdate-pi4" is in the root directory of the USB drive and has not been renamed.
→ A differentiation is made between small and capital letters. The files and folders within the "usbupdate-mx6" or "usbupdate-pi4" folder must have the file and folder names of the delivery state.
→ Check if all keys have valid values.
→ Check if a section or a key with ";" or "#" was commented out.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

167

9. Annex

9.1. Environmental Protection

9.1.1. Emission

When used correctly, our modules do not produce any harmful emissions.

9.1.2. Disposal

At the end of their service life, modules may be returned to the manufacturer against payment of an all-inclusive charge to cover costs. The manufacturer will then arrange for the modules to be recycled.

9.2. Maintenance/Upkeep

⚠ WARNING

Do not insert, apply, detach or touch connections while in operation – risk of destruction or malfunction.

Disconnect all incoming power supplies before working on our modules; this also applies to connected peripheral equipment such as externally powered sensors, programming devices, etc. All ventilation openings must always be kept free of any obstruction.

→ The modules are maintenance-free when used correctly.
→ Clean only with a dry, non-fluffing cloth.
→ Do not use detergents!

9.3. Repairs / Customer service

⚠ WARNING

Repair work may only be carried out by the manufacturer or its authorised service engineers.
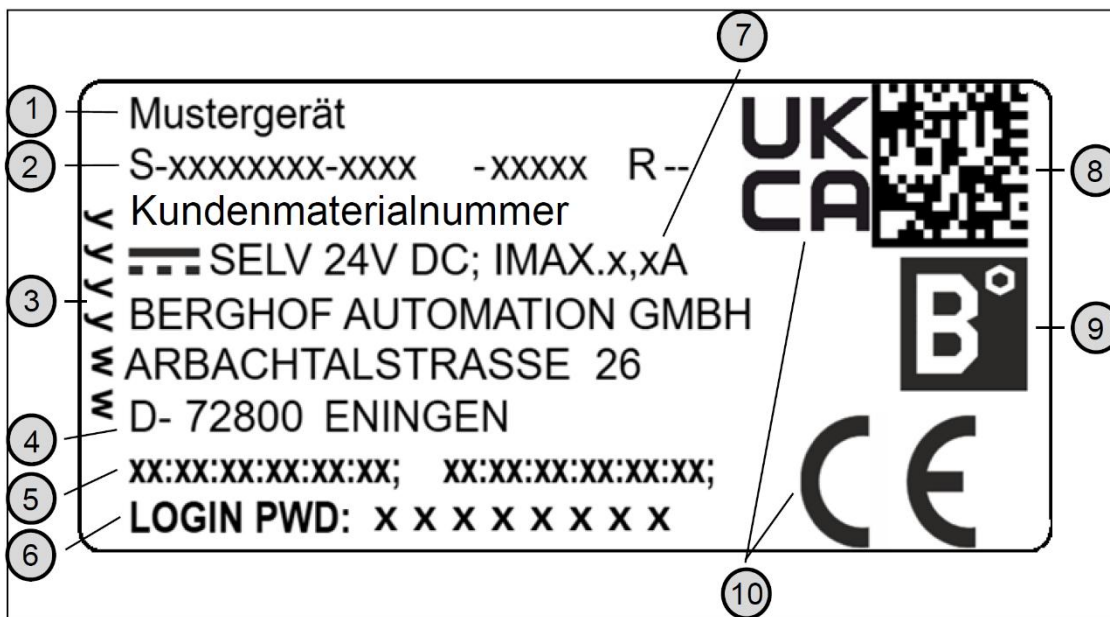
9.3.1. Warranty

Sold under statutory warranty conditions. Warranty lapses in the event of unauthorised attempts to repair the equipment and/or product, or in the event of any other form of intervention.

9.4. Type plate

Type plate descriptions (example)

No.	Designation	No.	Designation
1	Product description	7	Supply voltage and maximum current
2	Order number / serial number / version identification	8	QR code (identification no.)
3	Production date (year/week)	9	Manufacturer's logo
4	Manufacturer (manufacturer's address)	10	Conformity marking
5	MAC addresses of the unit		
6	Default password		

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

169

9.5. Addresses and Bibliography

9.5.1. Addresses

→ CiA

CAN in Automation; international manufacturers and users organisation for CAN users in the field of automation:
CAN in Automation e.V. (CiA)
Am Weichselgarten 26
D-91058 Erlangen / Germany
headquarters@can-cia.de
www.can-cia.de

→ ETG

EtherCAT Technology Group
ETG Headquarters
Ostendstraße 196
D-90482 Nuremberg / Germany
info@ethercat.org
www.ethercat.org

→ DIN-EN Standards

Beuth Verlag GmbH, 10772 Berlin
or
VDE-Verlag GmbH, 10625 Berlin

→ IEC Standards

VDE Verlag GmbH, 10625 Berlin
or
Internet search: www.iec.ch

9.5.2. Standards/Bibliography

Standard	Label
IEC61131-1 / EN61131-1	Programmable controllers Part 1: General information
IEC61131-2 / EN61131-2	Programmable controllers Part 2: Equipment requirements and tests
IEC61131-3 / EN61131-3	Programmable controllers Part 3: Programming languages
IEC61131-4 / EN61131Bl1	Programmable logic controllers Supplementary Sheet 1: User guidelines
IEC61000-6-4 / EN61000-6-4	German EMC Standard: Emitted interference
IEC61000-6-2 / EN61000-6-2	German EMC Standard: Noise immunity
ISO/DIS 11898	Draft International Standard: Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication
EN 61326-1	Electrical equipment for measurement, control and laboratory use - EMC requirements Part 1: General requirements
Bibliography	A variety of specialist publications on the CANbus is available from specialist bookshops or can be obtained through the CiA users' organisation.

Notice: Our Technical Support team will be glad to provide other literature references on request.

Berghof Automation GmbH | Arbachtalstrasse 26 | 72800 Eningen | www.berghof-automation.com
Berghof_Control_Systemmanual_1.5_EN.docx

171